aws

Guida per gli sviluppatori

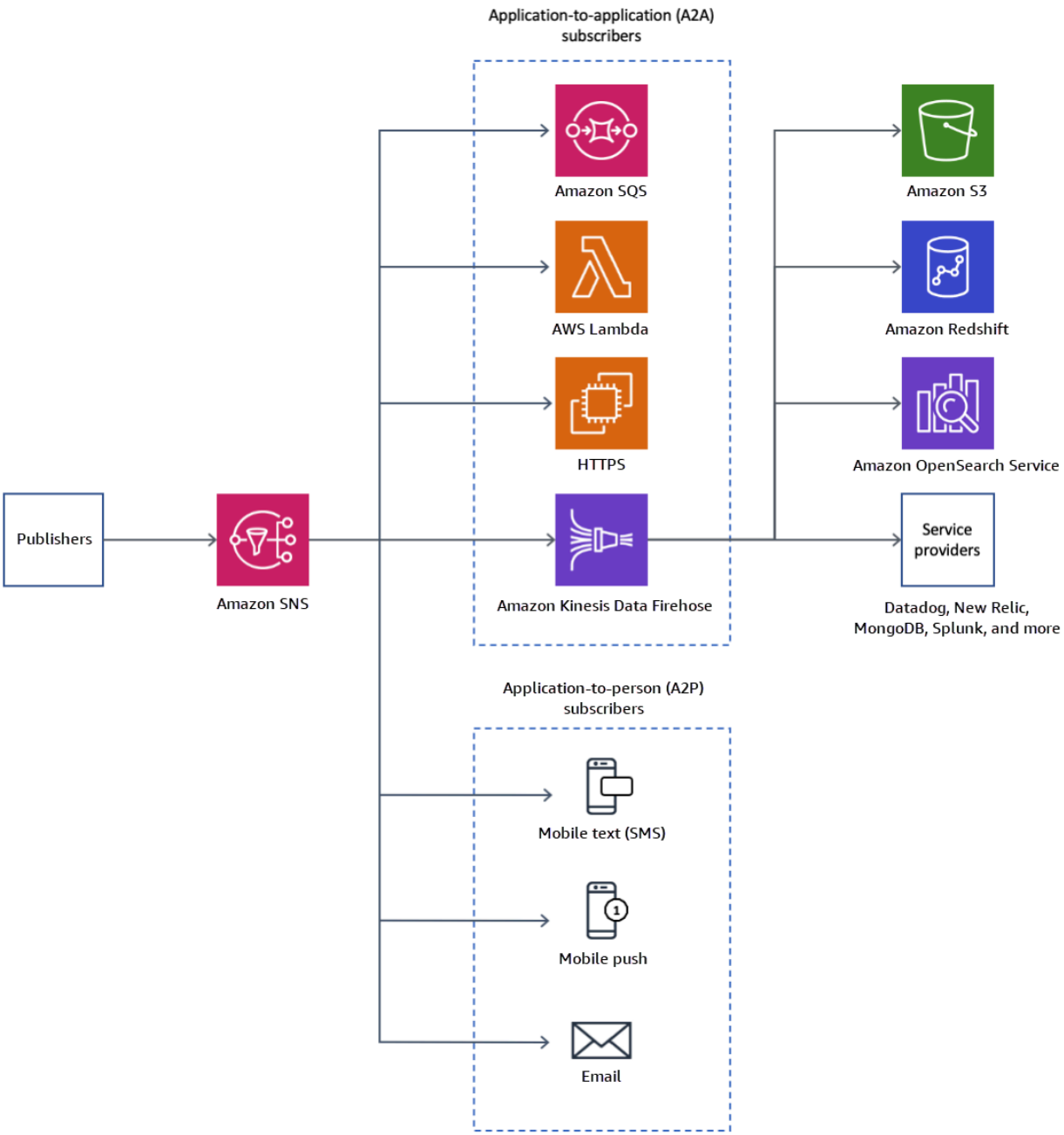
Amazon Simple Notification Service

Amazon Simple Notification Service: Guida per gli sviluppatori

Table of Contents

Che cos'è Amazon SNS?

Amazon Simple Notification Service (Amazon SNS) è un servizio gestito che offre la consegna dei messaggi dagli editori agli abbonati (noti anche come Produttori e Consumer). Gli editori comunicano in modo asincrono con gli abbonati creando e inviando messaggi a un argomento, che rappresenta un punto di accesso logico e un canale di comunicazione. I clienti possono abbonarsi all'argomento SNS e ricevere messaggi pubblicati utilizzando un tipo di endpoint supportato, come Amazon Data Firehose, Amazon SQS, HTTP, e-mail AWS Lambda, notifiche push mobili e messaggi di testo mobili (SMS).



Argomenti

- Caratteristiche e funzionalità
- Servizi correlati
- Accedere ad Amazon SNS

- Prezzi per Amazon SNS
- Scenari comuni di Amazon SNS
- Utilizzo di Amazon SNS con un SDK AWS

Caratteristiche e funzionalità

Amazon SNS offre le seguenti caratteristiche e funzionalità:

- Un messaggio pplication-to-application

  Una pplication-to-application messaggistica supporta abbonati come flussi di distribuzione Amazon Data Firehose, funzioni Lambda, code Amazon SQS, endpoint HTTP/S ed Event Fork Pipelines. AWS Per ulteriori informazioni, consulta Messaggistica applicazione-applicazione (A2A).

- pplication-to-person Una notifica

  pplication-to-person Le notifiche A forniscono notifiche utente agli abbonati come applicazioni mobili, numeri di cellulare e indirizzi e-mail. Per ulteriori informazioni, consulta Messaggistica da applicazione a persona (A2P).

- Argomenti standard e FIFO

  Utilizzare un argomento FIFO per garantire una precisa classificazione dei messaggi, per definirne i gruppi e per impedirne la duplicazione. È possibile utilizzare le code FIFO e standard per effettuare la sottoscrizione a un argomento FIFO. Per ulteriori informazioni, consulta Ordinamento e deduplicazione dei messaggi (argomenti FIFO).

  Utilizzare un argomento standard quando l'ordine di invio dei messaggi e la loro possibile duplicazione non risulta determinante. Tutti i protocolli di recapito supportati possono sottoscrivere un argomento standard.

- Durabilità dei messaggi

  Amazon SNS utilizza una serie di strategie che cooperano per garantire la durabilità dei messaggi:

  - I messaggi pubblicati vengono archiviati su più server e data center geograficamente separati.
  - Se un endpoint sottoscritto non è disponibile, Amazon SNS esegue una policy di nuovi tentativi di consegna.
  - Per conservare i messaggi che non vengono recapitati prima del termine della policy di nuovi tentativi di consegna, è possibile creare una coda DLQ.

- Archiviazione, riproduzione e analisi dei messaggi

  Puoi archiviare i messaggi con Amazon SNS in diversi modi, tra cui sottoscrivendo i flussi di distribuzione Firehose agli argomenti SNS, che ti consente di inviare notifiche a endpoint di analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift e altro ancora. Inoltre, gli argomenti FIFO di Amazon SNS supportano l'archiviazione e riproduzione dei messaggi come archivio di messaggi locale senza codice che consente ai proprietari degli argomenti di archiviare i messaggi all'interno del proprio argomento. Gli abbonati agli argomenti possono quindi recuperare (o riprodurre) i messaggi archiviati su un endpoint sottoscritto. Per ulteriori informazioni, consulta Archiviazione e riproduzione dei messaggi per argomenti FIFO.

- Attributi di messaggio

  Gli attributi del messaggio consentono di fornire metadati relativi a un messaggio. the section called “Attributi di messaggio”.

- Filtro dei messaggi

  Per impostazione predefinita, ogni sottoscrittore di un argomento riceve tutti i messaggi pubblicati nell'argomento. Per ricevere solo una sottocategoria di messaggi, un abbonato deve assegnare una policy di filtro alla sottoscrizione all'argomento. Un sottoscrittore può anche definire l'ambito della policy di filtro per abilitare il filtraggio basato sul payload o sugli attributi. Il valore predefinito per l'ambito della policy di filtro è `MessageAttributes`. Quando gli attributi del messaggio in arrivo corrispondono agli attributi della policy di filtro, il messaggio viene recapitato all'endpoint sottoscritto. In caso contrario, il messaggio viene filtrato. Quando l'ambito della policy di filtro è `MessageBody`, gli attributi della policy di filtro vengono confrontati con il payload. Per ulteriori informazioni, consulta Filtro dei messaggi.

- Sicurezza dei messaggi

  La crittografia lato server protegge il contenuto dei messaggi archiviati negli argomenti di Amazon SNS, utilizzando le chiavi di crittografia fornite da. AWS KMS Per ulteriori informazioni, consulta the section called “Crittografia a riposo”.

  Si può anche stabilire una connessione privata tra Amazon SNS e il cloud privato virtuale (VPC). Per ulteriori informazioni, consulta the section called “Riservatezza del traffico Internet”.

Servizi correlati

È possibile utilizzare i seguenti servizi con Amazon SNS:

- Amazon SQS offre una coda ospitata internamente sicura, durevole e disponibile che consente di integrare e separare i componenti e i sistemi software distribuiti. Amazon SQS è collegato ad Amazon SNS nei seguenti modi:
  - Amazon SNS offre code DLQ gestite da Amazon SQS per i messaggi non recapitabili.
  - È possibile effettuare la sottoscrizione di una coda Amazon SQS a un argomento SNS.
  - È possibile effettuare la sottoscrizione di una coda FIFO o di una coda standard di Amazon SQS a un argomento FIFO di Amazon SNS SNS. Solo le code FIFO di Amazon SQS garantiscono che i messaggi vengano ricevuti in ordine e senza duplicati.
- AWS Lambda permette di creare applicazioni che rispondono rapidamente alle nuove informazioni. Eseguire il codice dell'applicazione in funzioni Lambda su un'infrastruttura di calcolo altamente disponibile. Per ulteriori informazioni, consulta la Guida per gli sviluppatori di AWS Lambda. È possibile sottoscrivere una funzione Lambda a un argomento SNS.
- AWS Identity and Access Management (IAM) ti aiuta a controllare in modo sicuro l'accesso alle AWS risorse per i tuoi utenti. Utilizzare IAM per controllare chi può utilizzare i tuoi argomenti Amazon SNS (autenticazione), quali argomenti può utilizzare e in che modo (autorizzazione). Per ulteriori informazioni, consulta Utilizzo di policy basate su identità con Amazon SNS.
- AWS CloudFormationti consente di modellare e configurare AWS le tue risorse. Crea un modello che descriva le AWS risorse che desideri, inclusi gli argomenti e gli abbonamenti di Amazon SNS. AWS CloudFormation si occupa del provisioning e della configurazione di tali risorse per te. Per ulteriori informazioni, consulta la Guida per l'utente AWS CloudFormation.

Accedere ad Amazon SNS

Puoi configurare e gestire argomenti e abbonamenti SNS utilizzando la console Amazon SNS, gli strumenti a riga di comando o gli SDK. AWS

- La Console Amazon SNS fornisce una comoda interfaccia utente per la creazione di argomenti e abbonamenti, l'invio e la ricezione di messaggi e il monitoraggio di eventi e registri.
- Il AWS Command Line Interface (AWS CLI) ti dà accesso diretto all'API Amazon SNS per casi d'uso avanzati di configurazione e automazione. Per ulteriori informazioni, consultare Utilizzo di Amazon SNS con AWS CLI.
- AWS fornisce SDK in varie lingue. Per ulteriori informazioni, consulta SDK e kit di strumenti.

Prezzi per Amazon SNS

Amazon SNS non ha costi iniziali. I pagamenti vengono effettuati in base al numero di messaggi pubblicati, al numero di notifiche inviate e a eventuali chiamate API aggiuntive per la gestione di argomenti e abbonamenti. I prezzi di spedizione variano in base al tipo di endpoint. È possibile iniziare gratuitamente con il livello gratuito di Amazon SNS.

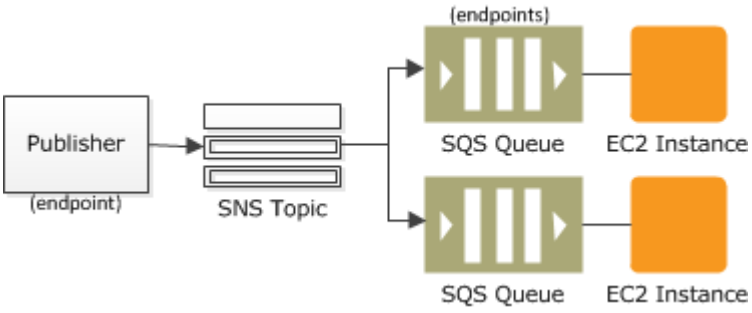
Per ulteriori informazioni, consultare Prezzi di Amazon SNS.

Scenari comuni di Amazon SNS

Integrazione di applicazioni

Lo scenario Fanout si verifica quando un messaggio pubblicato su un argomento SNS viene replicato e inviato a più endpoint, come flussi di distribuzione Firehose, code Amazon SQS, endpoint HTTP (S) e funzioni Lambda. Ciò consente l'elaborazione asincrona parallela.

Per esempio, si potrebbe sviluppare un'applicazione che invia un messaggio a un argomento SNS ogni volta che viene effettuato un ordine per un prodotto. Quindi, le code SQS che sono abbonate a quell'argomento SNS riceveranno notifiche identiche per il nuovo ordine. Un'istanza del server Amazon Elastic Compute Cloud (Amazon EC2) collegata a una delle code SQS può gestire l'elaborazione o l'evasione dell'ordine. È possibile allegare un'altra istanza del server Amazon EC2 a un data warehouse per l'analisi di tutti gli ordini ricevuti.



Un altro modo per utilizzare il "fan-out" è replicare i dati inviati al tuo ambiente di produzione con il tuo ambiente di prova. Parlando dell'esempio precedente in modo più approfondito, è possibile sottoscrivere un'altra coda SQS allo stesso argomento SNS per i nuovi ordini in entrata. Quindi, collegando questa nuova coda SQS al proprio ambiente di prova, è possibile continuare a migliorare e testare la propria applicazione utilizzando i dati ricevuti dall'ambiente di produzione.

> **Important**
>
> Assicurati di rispettare la privacy e la sicurezza dei dati prima di inviare i dati di produzione all'ambiente di test.

Per ulteriori informazioni, consulta le seguenti risorse:

- Flussi di distribuzione da Fanout a Firehose
- Funzioni da Fanout a Lambda
- Fan-out a code Amazon SQS
- Fanout agli endpoint HTTP(S)
- Elaborazione basata sugli eventi con Amazon SNS AWS e servizi di elaborazione, storage, database e rete

Avvisi dall' applicazione

Gli avvisi di sistema e dell' applicazione sono notifiche, attivate da soglie predefinite. Amazon SNS può inviare tali notifiche a determinati utenti tramite SMS ed e-mail. Ad esempio, puoi ricevere una notifica immediata quando si verifica un evento, ad esempio una modifica specifica al tuo gruppo Amazon EC2 Auto Scaling, un nuovo file caricato in un bucket Amazon S3 o una soglia metrica superata in Amazon. CloudWatch Per ulteriori informazioni, consulta Configurazione delle notifiche di Amazon SNS nella Amazon CloudWatch User Guide.

Notifiche all'utente

Amazon SNS può inviare messaggi e-mail push e messaggi di testo (messaggi SMS) a singoli o gruppi. Ad esempio, è possibile inviare conferme di ordine e-commerce come notifiche utente. Per avere ulteriori informazioni su come usare Amazon SNS per inviare messaggi SMS, consultare Messaggi di testo per dispositivi mobili (SMS).

Notifiche push per dispositivi mobili

Le notifiche push per dispositivi mobili ti permettono di inviare messaggi di notifica direttamente alle app su dispositivi mobili. Per esempio, puoi utilizzare Amazon SNS per inviare notifiche di aggiornamento a un'app. Il messaggio di notifica può includere un collegamento per eseguire il

download e installare l'aggiornamento. Per avere ulteriori informazioni su come usare Amazon SNS per inviare messaggi SMS, consultare Notifiche push per dispositivi mobili.

Utilizzo di Amazon SNS con un SDK AWS

AWS I kit di sviluppo software (SDK) sono disponibili per molti linguaggi di programmazione più diffusi. Ogni SDK fornisce un'API, esempi di codice, e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Documentazione sugli SDK	Esempi di codice
AWS SDK for C++	AWS SDK for C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK for Go	AWS SDK for Go esempi di codice
AWS SDK for Java	AWS SDK for Java esempi di codice
AWS SDK for JavaScript	AWS SDK for JavaScript esempi di codice
SDK AWS for Kotlin	SDK AWS for Kotlin esempi di codice
AWS SDK for .NET	AWS SDK for .NET esempi di codice
AWS SDK for PHP	AWS SDK for PHP esempi di codice
AWS Tools for PowerShell	Strumenti per esempi di PowerShell codice
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) esempi di codice
AWS SDK for Ruby	AWS SDK for Ruby esempi di codice
AWS SDK for Rust	AWS SDK for Rust esempi di codice
SDK AWS per SAP ABAP	SDK AWS per SAP ABAP esempi di codice
SDK AWS per Swift	SDK AWS per Swift esempi di codice

Per esempi specifici di Amazon SNS, consultare Esempi di codice per Amazon SNS che utilizzano SDK AWS.

> **Esempio di disponibilità**
>
> Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link Provide feedback (Fornisci un feedback) nella parte inferiore di questa pagina.

Sorgenti e destinazioni degli eventi Amazon SNS

Amazon SNS può ricevere notifiche basate su eventi da molte risorse AWS e notifiche di fan out alle destinazioni application-to-application (A2A) e application-to-person (A2P). Questa sezione elenca le origini e le destinazioni degli eventi supportate e fornisce collegamenti per ulteriori informazioni.

Argomenti

Origini eventi Amazon SNS

In questa pagina sono elencati i servizi AWS che possono pubblicare eventi su argomenti Amazon SNS, raggruppati in base alle categorie AWS di prodotti.

> **Note**
>
> Amazon SNS ha introdotto Argomenti FIFO nel mese di ottobre 2020. Attualmente, la maggior parte dei servizi AWS supportano l'invio di eventi solo ad argomenti standard.

Servizi di analisi

AWSServizio	Vantaggio dell'utilizzo di con Amazon SNS
Amazon Athena - consente di analizzare i dati in Amazon S3 utilizzando SQL standard.	Ricevi notifiche quando i limiti di controllo vengono superati. Per ulteriori informazioni, consulta Impostazione dei limiti per il controllo dell'utilizzo nella Guida per l'utente di Amazon Athena.
AWS Data Pipeline – Automatizza il movimento e la trasformazione dei dati.	Ricevere notifiche sullo stato dei component i della pipeline. Per ulteriori informazioni, consulta la sezione SnsAlarm nella Guida per gli sviluppatori di AWS Data Pipeline.

AWSServizio	Vantaggio dell'utilizzo di con Amazon SNS
Amazon Redshift - Il servizio gestisce tutte le attività di configurazione, esecuzione e dimensionamento di un data warehouse.	Ricevi notifiche degli eventi Amazon Redshift. Per ulteriori informazioni, consulta Notifiche di eventi Amazon Redshift nella Guida alla gestione di Amazon Redshift.

Integrazione servizi applicazioni

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
Amazon EventBridge: fornisce un flusso di dati in tempo reale dalle tue applicazioni, applicazioni software-as-a-service (SaaS) e AWS servizi e indirizza tali dati verso destinazioni, tra cui Amazon SNS. EventBridge in precedenza si chiamava Events. CloudWatch	Ricevi notifiche di eventi. EventBridge Per ulteriori informazioni, consulta EventBridgegli obiettivi di Amazon nella Amazon EventBridge User Guide.
AWS Step Functions - Consente di combinare funzioni AWS Lambda e altri servizi AWS per creare applicazioni business-critical.	Ricevere la notifica degli eventi Step Functions . Per ulteriori informazioni, consulta Richiama Amazon SNS con Step Functions nella Guida per gli sviluppatori di AWS Step Functions.

Servizi di gestione di costi e fatturazione

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS Billing and Cost Management - fornisce funzioni che consentono di monitorare i costi e pagare la fattura.	Ricevi notifiche di budget, notifiche di modifica del prezzo e avvisi di anomalia. Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS Billing: • Creazione di un argomento Amazon SNS per le notifiche dei budget

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
	• Configurazione delle notifiche • Rilevamento di spese insolite con il rilevamento di anomalie dei costi AWS

Servizi applicativi aziendali

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon Chime - Consente di incontrare, chattare e effettuare chiamate aziendali all'interno e all'esterno dell'organizzazione.	Ricevere importanti notifiche degli eventi di riunione. Per ulteriori informazioni, consulta la notifica eventi Amazon Chime SDK nella Guida per sviluppatori di Amazon Chime.

Servizi di elaborazione

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon EC2 Auto Scaling - Consente di avere il numero corretto di istanze Amazon Elastic Compute Cloud (Amazon EC2) disponibili per gestire il carico dell'applicazione.	Ricevi notifiche quando Auto Scaling avvia o interrompe le istanze Amazon EC2 nel gruppo Auto Scaling. Per ulteriori informazioni, consulta l'argomento relativo alla ricezione di notifiche Amazon SNS quando il gruppo Auto Scaling viene ridimensionato nella Guida per l'utente di Amazon EC2 Auto Scaling.
EC2 Image Builder: aiuta ad automatizzare la creazione, la gestione e l'implementazione di immagini server personalizzate up-to-date, sicure e preinstallate con software e impostazioni per soddisfare specifici standard IT.	Ricevi notifiche quando le build sono completate. Per ulteriori informazioni, consulta Monitoraggio delle immagini server più recenti nelle pipeline di EC2 Image Builder sul Compute blog AWS.

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS Elastic Beanstalk – Gestisce automatic amente i dettagli di provisioning della capacità, bilanciamento del carico, dimensionamento e monitoraggio dello stato dell'applicazione.	Ricevi notifiche di eventi importanti che interessano la tua applicazione. Per ulteriori informazioni, consulta Notifiche dell'ambiente Elastic Beanstalk con Amazon SNS nella Guida per gli sviluppatori AWS Elastic Beanstalk.
AWS Lambda - Consente di eseguire il codice senza effettuare il provisioning dei server o senza gestirli.	Ricevere i dati di output della funzione impostando un argomento SNS come una coda di lettere non recapitate Lambda o una destinazione Lambda. Per ulteriori informazioni, consulta Chiamata asincrona nella Guida per gli sviluppatori AWS Lambda.
Amazon Lightsail — Aiuta gli sviluppatori a iniziare a utilizzare AWS per creare siti web o applicazioni web.	Riceve una notifica quando un parametro per una delle istanze, dei database o dei sistemi di bilanciamento del carico attraversa una soglia specificata. Per ulteriori informazioni, consulta Aggiunta di contatti di notifica in Amazon Lightsail nella Guida per gli sviluppat ori di Amazon Lightsail.

Servizi di container

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon EKS Distribuzione - Consente di creare cluster affidabili e sicuri ovunque vengano distribuite le applicazioni.	Tieni traccia degli aggiornamenti e delle patch di sicurezza per i cluster creati con Amazon EKS Distro. Per ulteriori informazioni, consulta Presentazione di Amazon EKS Distro - una distribuzione Kubernetes open source utilizzata da Amazon EKS.

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon Elastic Container Service (Amazon ECS) - Consente di eseguire, arrestare e gestire i contenitori in un cluster.	Ricevi notifiche quando è disponibile un nuovo AMI ottimizzato per Amazon ECS. Per ulteriori informazioni, consulta la sezione relativa alla sottoscrizione alle notifiche per l'aggiornamento di AMI Linux ottimizzate per Amazon ECS nella Guida per gli sviluppatori di Amazon Elastic Container Service.

Servizi di coinvolgimento dei clienti

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon Connect - Consente di configurare un contact center cloud omnicanale per interagire con i clienti.	Ricevi avvisi e convalide. Per ulteriori informazioni, consulta La potenza di AWS con Amazon Connect nella Guida per l'amministratore di Amazon Connect.
Amazon Pinpoint - Consente di coinvolgere i clienti inviando loro e-mail, SMS, messaggi vocali e notifiche push.	Configura SMS bidirezionali, consentendo così di ricevere messaggi dai clienti. Per ulteriori informazioni, consulta Uso della messaggistica SMS bidirezionale in Amazon Pinpointnella Guida per l'utente di Amazon Pinpoint.
Amazon Simple Email Service (Amazon SES) - Offre un metodo conveniente per inviare e ricevere e-mail usando domini e indirizzi e-mail personali.	Ricevi notifiche di mancati recapiti (bounce), reclami e consegne. Per maggiori informazioni, consulta Configurare le notifiche Amazon SNS per Amazon SES nella Guida per sviluppatori di Amazon Simple EmailService.

Servizi di database

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS Database Migration Service - Consente di eseguire la migrazione dei dati dai database locali nel cloud AWS	Ricezione di notifiche quando si verificano AWS DMS; ad esempio, quando un'istanza di replica viene creata o eliminata. Per ulteriori informazioni, vedi Utilizzo degli eventi e delle notifiche AWS Database Migration Service nella AWS Database Migration ServiceGuida per l'utente.
Amazon DynamoDB - Combina prestazioni elevate e prevedibili con una scalabilità ottimale nel servizio database NoSQL interamente gestito.	Ricevi notifiche quando si verificano eventi di manutenzione. Per ulteriori informazioni, consulta Personalizzazione delle impostazioni del cluster DAX nella Guida per gli sviluppatori di Amazon DynamoDB.
Amazon ElastiCache: fornisce una cache in memoria ad alte prestazioni, ridimensionabile ed economica, eliminando al contempo la complessità associata all'implementazione e alla gestione di un ambiente di cache distribui to.	Ricevi notifiche quando si verificano eventi significativi. Per ulteriori informazioni, consulta Notifiche di eventi e Amazon SNS nella Guida per l'utente di Amazon ElastiCache for Memcached.
Amazon Neptune - Consente di creare ed eseguire applicazioni che funzionano con set di dati altamente connessi.	Ricevi notifiche quando si verifica un evento Neptune. Per ulteriori informazioni, consulta la sezione relativa all'utilizzo delle notifiche eventi Neptune nella Guida per l'utente Neptune.
Amazon Redshift - Gestisce tutte le attività di configurazione, esecuzione e dimensionamento di un data warehouse.	Ricevi notifiche degli eventi Amazon Redshift. Per ulteriori informazioni, consulta Notifiche di eventi Amazon Redshift nella Guida alla gestione di Amazon Redshift.
Amazon Relational Database Service – Semplifica la configurazione, l'uso e il	Ricevi notifiche degli eventi RDS di Amazon. Per ulteriori informazioni, consulta la sezione

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
dimensionamento di un database relazionale nel cloud AWS.	Utilizzo della notifica eventi Amazon RDS nella Guida per l'utente di Amazon RDS.

Servizi e strumenti per sviluppatori

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS CodeBuild - Compila il codice sorgente, esegue unit test e prepara artefatti pronti per essere distribuiti.	Ricevi notifiche quando le build hanno esito positivo, non riescono o si spostano da una fase di compilazione all'altra. Per ulteriori informazioni, consulta l'esempio di compilazione delle notifiche CodeBuild nella Guida per l'AWS CodeBuildutente.
AWS CodeCommit - Fornisce il controllo delle versioni per archiviare e gestire in modo privato le risorse nel cloud.	Ricevi notifiche sugli eventi CodeCommit del repository. Per ulteriori informazioni, consulta Esempio: creazione di un attivatore AWS CodeCommit per un argomento Amazon SNS nella Guida per l'utente di AWS CodeCommit.
AWS CodeDeploy — Automatizza la distribuzione dell'applicazione a istanze Amazon EC2, istanza locali, funzioni Lambda serverless o servizi Amazon ECS.	Ricevi notifiche per CodeDeploy implement azioni o eventi di istanza. Per ulteriori informazioni, consulta Creare un trigger per un CodeDeploy evento nella Guida per l'AWS CodeDeployutente.
Amazon CodeGuru: raccoglie i dati sulle prestazioni di runtime dalle tue applicazioni live e fornisce consigli che possono aiutarti a ottimizzare le prestazioni delle tue applicazioni.	Ricevi notifiche quando si verificano anomalie. Per ulteriori informazioni, consulta Lavorare con i report sulle anomalie e sui consigli nella Amazon CodeGuru User Guide.
AWS CodePipeline - Automatizza i passaggi necessari per rilasciare continuamente le modifiche software.	Ricevere notifiche sulle azioni di approvazione. Per ulteriori informazioni, consulta Gestire le azioni di approvazione CodePipeline nella Guida per l'AWS CodePipelineutente.

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS CodeStar: creazione, gestione e utilizzo di progetti di sviluppo software su AWS.	Ricevi notifiche sugli eventi che si verificano nelle risorse utilizzate. Per ulteriori informazioni, consulta la sezione relativa alla configurazione degli argomenti Amazon SNS per le notifiche nella Guida per l'utente della console degli strumenti per sviluppatori.

Servizi web e mobili front-end

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon Pinpoint - Consente di coinvolgere i clienti inviando loro e-mail, SMS, messaggi vocali e notifiche push.	Configura SMS bidirezionali, consentendo così di ricevere messaggi dai clienti. Per ulteriori informazioni, consulta Uso della messaggistica SMS bidirezionale in Amazon Pinpoint nella Guida per l'utente di Amazon Pinpoint.

Servizi per lo sviluppo dei giochi

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon GameLift: fornisce soluzioni per l'hosting di server di gioco multiplayer basati su sessioni nel cloud, incluso un servizio completamente gestito per la distribuzione, il funzionamento e il ridimensionamento dei server di gioco.	Ricevi notifiche di eventi di matchmaking e coda. Per ulteriori informazioni, consulta le pagine seguenti: • Per le notifiche di matchmaking, consulta Configurare la notifica FlexMatch degli eventi nella Amazon GameLift FlexMatch Developer Guide. • Per le notifiche sulla coda, consulta Configurare la notifica degli eventi per il

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
	posizionamento delle sessioni di gioco nella Amazon GameLift Developer Guide.

Servizi Internet of Things

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS IoT Core - Fornisce i servizi cloud che connettono i dispositivi IoT ad altri dispositivi eAWSServizi cloud.	Ricezione di notifiche di eventi AWS IoT Core. Per ulteriori informazioni, consulta Creazione di una regola Amazon SNS nella AWS IoTGuida per gli sviluppatori.
AWS IoT Device Defender – Esegue l'auditing della configurazione dei dispositivi, monitorar e i dispositivi connessi per rilevare eventuali comportamenti anomali e mitigare i rischi di sicurezza.	Ricevi allarmi quando un dispositivo viola un comportamento. Per ulteriori informazi oni, consulta Utilizzo di rilevamento AWS IoT Device Defender nella AWS IoTGuida per gli sviluppatori.
AWS IoT Events – Consente di monitorare le apparecchiature o i parchi di dispositivi e individuare errori o modifiche di funzionam ento e attivare le relative operazioni quando tali eventi si verificano.	Ricezione di notifiche di eventi AWS IoT Events. Per ulteriori informazioni, consultare Amazon Simple Notification Service nella AWS IoT EventsGuida per gli sviluppatori
AWS IoT Greengrass – Estende AWS solo ai dispositivi fisici in modo che possano interveni re a livello locale sui dati che generano, continuando a utilizzare il cloud per la gestione, l'analisi e lo storage duraturo.	Ricezione di notifiche di eventi AWS IoT Greengrass. Per ulteriori informazioni, consulta la sezione relativa al connettore SNS nella Guida per lo sviluppatore di AWS IoT Greengrass Version 1.

Servizi Machine Learning

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon CodeGuru: raccoglie i dati sulle prestazioni di runtime dalle tue applicazioni live e fornisce consigli che possono aiutarti a ottimizzare le prestazioni delle tue applicazioni.	Ricevi notifiche quando si verificano anomalie. Per ulteriori informazioni, consulta Lavorare con i report sulle anomalie e sui consigli nella Amazon CodeGuru User Guide.
Amazon DevOps Guru: genera informazioni operative utilizzando l'apprendimento automatico per aiutarti a migliorare le prestazioni delle tue applicazioni operative.	Proseguire approfondimenti e conferme. Per ulteriori informazioni, consulta Offri approfondimenti operativi basati su ML ai tuoi team di chiamata tramite PagerDuty Amazon DevOps Guru sul blog AWSManagement & Governance.
Amazon Lookout for Metrics - Individua le anomalie nei dati, ne determina le cause principali e consente di intervenire rapidamente.	Ricevi notifiche di anomalie. Per ulteriori informazioni, consulta Utilizzo di Amazon SNS con Lookout for Metrics nella Guida per gli sviluppatori di Amazon Lookout for Metrics.
Amazon Rekognition - Consente di aggiungere funzionalità di analisi di immagini e video alle applicazioni	Ricevi notifiche dei risultati delle richieste. Per ulteriori informazioni, consulta Riferimento: Notificazione dei risultati dell'analisi video nella Guida per gli sviluppatori di Amazon Rekognition.
Amazon SageMaker: consente ai data scientist e agli sviluppatori di creare e addestrare modelli di machine learning e quindi di distribuirli direttamente in un ambiente ospitato pronto per la produzione.	Ricevi notifiche quando un oggetto dati viene etichettato. Per ulteriori informazioni, consulta Creating a streaming labeling job nella Amazon SageMaker Developer Guide.

Servizi di governance e gestione

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
AWS Chatbot— Consente ai DevOps team di sviluppo software di utilizzare le chat room di Amazon Chime e Slack per monitorare e rispondere agli eventi operativi nel cloud. AWS	Distribuisci notifiche alle chat room. Per ulteriori informazioni, consulta Configurazione della AWS Chatbot nella Guida per l'amministratore di AWS Chatbot.
AWS CloudFormation - Consente di creare ed effettuare il provisioning delle distribuz ioni dell'infrastruttura AWS in modo ripetuto e prevedibile.	Ricevi notifiche quando gli stack vengono creati e aggiornati. Per ulteriori informazioni, consulta Impostazione di Opzioni di stack AWS CloudFormation nella Guida per l'utente di AWS CloudFormation.
AWS CloudTrail — Fornisce la cronologia degli eventi delle attività del Account AWS.	Ricevi notifiche quando vengono CloudTrai l pubblicati nuovi file di log nel tuo bucket Amazon S3. Per ulteriori informazioni, consulta Configurazione delle notifiche di Amazon SNS nella Guida CloudTrail per AWS CloudTrai l'utente.
Amazon CloudWatch: monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale.	Ricevi notifiche quando gli allarmi cambiano stato. Per ulteriori informazioni, consulta Using Amazon CloudWatch alarms nella Amazon CloudWatch User Guide.
AWS Config - Fornisce una panoramica dettagliata della configurazione delle risorse AWS nel tuo account Account AWS.	Ricevi notifiche quando le risorse vengono aggiornate o quando AWS Config valuta le regole personalizzate o gestite rispetto alle risorse. Per ulteriori informazioni, consulta Notifiche che AWS Config invia a un argomento SNS ed Esempio di notifiche di modifica degli elementi di configurazione nella AWS ConfigGuida per gli sviluppatori.

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
AWS Control Tower – Consente di configura re e gestire un ambiente AWS con più account sicuro e conforme.	Utilizza gli avvisi per aiutarti a prevenire la deriva all'interno della tua landing zone e ricevere notifiche di conformità. Per ulteriori informazioni, consulta Controllo degli avvisi tramite Amazon Simple Notification nella Guida per l'utente di AWS Control Tower.
AWS License Manager - Consente di gestire le licenze software dei fornitori a tra AWS e gli ambienti locali on-premise.	Ricevere notifiche e avvisi di License Manager. Per ulteriori informazioni, consulta Impostazioni in License Manager nella Guida per l'utente di License Manager e Creazione di ServiceNow incidenti per AWS License Manager le notifiche sul blog AWS Management & Governance.
AWS Service Catalog - Consente agli amministratori IT di creare e gestire portafogli di prodotti approvati e distribuirli a utenti finali consentendo loro di accedere ai prodotti che desiderano in un portale personalizzato.	Ricevi notifiche sugli eventi dello stack. Per ulteriori informazioni, consulta le limitazioni per le notifiche di AWS Service Catalog nella guida per l'amministratore di Service Catalog.
AWS Systems Manager - Consente di visualizz are e controllare l'infrastruttura su AWS.	Ricevi notifiche sullo stato dei comandi. Per ulteriori informazioni, consulta Monitoraggio delle modifiche di stato di Systems Manager utilizzando le notifiche Amazon SNS nella AWS Systems ManagerGuida per l'utente di.

Servizi multimediali

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon Elastic Transcoder - Consente di convertire i file multimediali memorizzati in Amazon S3 in file multimediali nei formati	Ricevi notifiche quando i processi cambiano lo stato. Per ulteriori informazioni, consulta Noticole sullo stato di un processo nella Guida per sviluppatori di Amazon Elastic Transcoder.

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
richiesti dai dispositivi di riproduzione dei consumatori.	

Servizi di trasferimento e migrazione

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS Application Discovery Service – Aiuta a pianificare la migrazione al cloud AWS raccogliendo dati di utilizzo e configurazione relativi ai server locali.	Ricevi notifiche degli eventi tramite AWS CloudTrail. Per ulteriori informazioni, consulta Registrazione delle chiamate API dell'Application Discovery Service con AWS CloudTrail nella Guida per l'utente dell'Application Discovery Service.
AWS Database Migration Service - Cnsente di eseguire la migrazione dei dati dai database locali nel cloud AWS.	Ricezione di notifiche quando si verificano AWS DMS; ad esempio, quando un'istanza di replica viene creata o eliminata. Per ulteriori informazioni, vedi Utilizzo degli eventi e delle notifiche AWS Database Migration Service nella AWS Database Migration ServiceGuida per l'utente.
AWS Snowball— Utilizza dispositivi di archiviazione fisici per trasferire grandi quantità di dati tra Amazon S3 e la posizione di archiviazione dei dati in sede a velocità elevate. faster-than-internet	Ricevi notifiche per i lavori Snowball. Per ulteriori informazioni, consulta gli argomenti seguenti: • Notifiche Snowball nella Guida per l'utente di AWS Snowball • Passaggio 5: Scegli le tue preferenze di notifica nella AWSGuida per sviluppatori di Snowball Edge •

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
	Passaggio 5: Scegli le tue preferenze di notifica nella AWSGuida dell'utente di Snowcone

Reti e servizi di distribuzione di contenuti

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon API Gateway: consente di creare e distribuire REST e WebSocket API personali zzate su qualsiasi scala.	Ricevere messaggi inviati a un endpoint API Gateway. Per ulteriori informazioni, consulta Tutorial: creazione di un'API REST API Gateway con integrazione AWS nella Guida per sviluppatori di API Gateway.
Amazon CloudFront: velocizza la distribuz ione di contenuti Web statici e dinamici, come .html, .css, .php, immagini e file multimediali.	Ricevi notifiche quando si verificano allarmi basati su metriche specifiche. CloudFront Per ulteriori informazioni, consulta la sezione Impostazione degli allarmi per ricevere notifiche nell'Amazon CloudFront Developer Guide.
AWS Direct Connect — Collega la rete interna a una località AWS Direct Connect tramite un cavo Ethernet standard in fibra ottica.	Ricevi notifiche quando allarmi che monitoran o lo stato di un stato AWS Direct Connect di modifica della connessione. Per ulteriori informazioni, consulta Creazione di CloudWatc h allarmi per monitorare le AWS Direct Connect connessioni nella Guida per l'AWS Direct Connectutente.
Elastic Load Balancing — Distribuisce automaticamente il traffico in ingresso su più destinazioni, ad esempio su istanze Amazon EC2, contenitori e indirizzi IP, in più o più zone di disponibilità.	Ricevi notifiche degli allarmi creati per gli eventi di bilanciamento del carico. Per ulteriori informazioni, consulta Creare CloudWatch allarmi per il sistema di bilanciamento del carico nella Guida utente di Classic Load Balancers.

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon Route 53: fornisce la registrazione del dominio, il routing DNS e il controllo dell'integrità.	Ricevere una notifica quando un controllo dello stato non è integro. Per ulteriori informazioni, consulta Come ricevere una notifica Amazon SNS quando un controllo dello stato non è integro (console) nella Guida per sviluppatori di Amazon Route 53.
Amazon Virtual Private Cloud (Amazon VPC) – Consente di avviare risorse AWS in una rete virtuale definita.	Ricevere notifiche per eventi specifici che si verificano nell'endpoint di interfaccia. Per ulteriori informazioni, consulta Come creare e gestire una notifica per un servizio endpoint nella Amazon VPC User Guide.

Servizi per sicurezza, identità e conformità

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS Directory Service – Offre diversi modi per utilizzare Microsoft Active Directory (AD) con altri servizi AWS.	Ricevere e-mail o messaggi di testo (SMS) quando lo stato della directory cambia. Per ulteriori informazioni, consulta Configurare le notifiche sullo stato della directory nella AWS Directory ServiceGuida di amministrazione.
Amazon GuardDuty: fornisce un monitoraggio continuo della sicurezza per aiutare a identific are attività impreviste e potenzialmente non autorizzate o dannose nel tuo AWS ambiente.	Ricevere notifiche sui tipi di risultati rilasciati recentemente, aggiornamenti ai tipi di risultati esistenti e altre modifiche alle funzionalità. Per ulteriori informazioni, consulta l'argomen to Abbonamento agli GuardDuty annunci SNS nella Amazon User Guide. GuardDuty
Amazon Inspector - Esegue il test dell'acce ssibilità di rete delle istanze Amazon EC2 e dello stato di sicurezza delle applicazioni che vengono eseguite sulle istanze.	Ricevi notifiche per gli eventi Amazon Inspector . Per ulteriori informazioni, consulta Impostazi one di un argomento SNS per le notifiche di

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
	Amazon Inspector nella Guida dell'utente di Amazon Inspector.
AWS Security Hub: automatizza i controlli di sicurezza di AWSI e centralizza gli avvisi di sicurezza.	Ricevi notifiche su annunci AWS Security Hub, comprese le notifiche su controlli AWS Security Hub o standard che sono stati aggiunti, modificati o ritirati. Per ulteriori informazioni, consulta Iscrizione agli annunci AWS Security Hub con Amazon SNS.

Servizi serverless

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
Amazon DynamoDB – Combina prestazioni elevate e prevedibili con una scalabilità ottimale nel servizio database NoSQL interamente gestito.	Ricevi notifiche quando si verificano eventi di manutenzione. Per ulteriori informazioni, consulta Personalizzazione delle impostazioni del cluster DAX nella Guida per gli sviluppatori di Amazon DynamoDB.
Amazon EventBridge: fornisce un flusso di dati in tempo reale dalle tue applicazioni, applicazioni software-as-a-service (SaaS) e AWS servizi e indirizza tali dati verso destinazioni, tra cui Amazon SNS. EventBridge in precedenza si chiamava Events. CloudWatch	Ricevi notifiche di eventi. EventBridge Per ulteriori informazioni, consulta EventBridgegli obiettivi di Amazon nella Amazon EventBridge User Guide.
AWS Lambda - Consente di eseguire il codice senza effettuare il provisioning dei server o senza gestirli.	Ricevere i dati di output della funzione impostando un argomento SNS come una coda di lettere non recapitate Lambda o una destinazione Lambda. Per ulteriori informazioni, consulta Chiamata asincrona nella AWS LambdaGuida per gli sviluppatori.

Servizi di storage

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
AWS Backup – Consente di centralizzare e automatizzare il backup dei dati in tutti i servizi AWS nel cloud e in locale.	Ricezione di notifiche di eventi AWS Backup. Per ulteriori informazioni, consulta Utilizzo di Amazon SNS per monitorare eventi AWS Backup nella AWS BackupGuida per gli sviluppatori.
Amazon Elastic File System - Consente di archiviare i file per le istanze Amazon EC2.	Ricevi notifiche degli avvisi che hai creato per gli eventi EFS Amazon. Per ulteriori informazioni, consulta Strumenti di monitoraggio automatizzato nella Guida per l'utente di Amazon Elastic File System.
Amazon S3 Glacier - Consente di archiviare i dati utilizzati raramente.	Impostare la configurazione delle notifiche per un vault di modo che un messaggio venga inviato a un argomento di SNS al completamento del processo. Per ulteriori informazioni, consulta Configurazione delle notifiche di vault in Amazon S3 Glacier nella Guida per gli sviluppatori Amazon S3 Glacier.
Amazon Simple Storage Service (Amazon S3) — Fornisce l'archiviazione degli oggetti.	Ricevi notifiche quando si verificano modifiche a un bucket Amazon S3 o nel raro caso in cui gli oggetti non vengono replicati nella regione di destinazione. Per ulteriori informazioni, consulta Spiegazione passo per passo: come configurare un bucket per le notifiche (argomento SNS o coda SQS) e Monitoraggio dell'avanzamento con i parametri di replica e le notifiche di eventi Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.
AWS Snowball— Utilizza dispositivi di archiviazione fisici per trasferire grandi quantità di dati tra Amazon S3 e la posizione di archiviazione	Ricevi notifiche per i lavori Snowball. Per ulteriori informazioni, consulta gli argomenti seguenti:

AWSServizio	Vantaggio dell'utilizzo con Amazon SNS
dei dati in sede a velocità elevate. faster-than-internet	• Notifiche Snowball nella Guida per l'utente di AWS Snowball • Passaggio 5: Scegli le tue preferenze di notifica nella AWSGuida per sviluppatori di Snowball Edge • Passaggio 5: Scegli le tue preferenze di notifica nella AWSGuida dell'utente di Snowcone

Origini eventi aggiuntivi

Origine	Vantaggio dell'utilizzo con Amazon SNS
Aggiornamenti giornalieri delle funzionalità AWS	Ricevi tempestivamente informazioni dettagliate su release e aggiornamenti AWS tramite un argomento Amazon SNS. Queste versioni includono endpoint Amazon VPC Regioni AWSServizi AWS, Servizi AWS integrati con AWS Service Quotas, tipi di istanze Amazon EC2, tipi di istanze Amazon, tipi di istanze Amazon SageMaker Nimble Studio, versioni del motore di database Amazon RDS e versioni di Amazon MSK Apache Kafka. Per ulteriori informazioni, consultare Subscribe to AWS Daily Feature Updates via Amazon SNS nel Blog AWS News.
Intervalli di indirizzi IP AWS	Ricevi notifiche di modifiche a intervalli IP AWS tramite un argomento Amazon SNS. Per ulteriori informazioni, consultare Notifiche degli intervalli di indirizzi IP di AWS nei Riferimenti generali di Amazon Web Services e Subscribe

Origine	Vantaggio dell'utilizzo con Amazon SNS
	to AWS Public IP Address Changes via Amazon SNS (Abbonamento alle modifiche degli indirizzi IP AWS mediante Amazon SNS) nel Blog AWS News.

Per ulteriori informazioni sull'elaborazione basata sugli eventi, consulta i seguenti argomenti:

- Cos'è un'architettura basata sugli eventi?
- Elaborazione basata sugli eventi con Amazon SNS eServizi AWS di elaborazione, storage, database e rete sul AWSCompute Blog
- Vantaggi delle architetture basate sugli eventi con Event Fork Pipelines AWS sul AWSCompute Blog

Destinazioni eventi di Amazon SNS

Questa pagina elenca tutte le destinazioni che possono ricevere informazioni sugli eventi, raggruppate per messaggi application-to-application (A2A) e notifiche application-to-person (A2P).

> **Note**
>
> Amazon SNS ha introdotto Argomenti FIFO nel mese di ottobre 2020. Attualmente, la maggior parte AWS servizi supportano la ricezione di eventi solo da argomenti standard SNS. Amazon SQS supporta la ricezione di eventi sia dagli argomenti standard SNS che FIFO.

A2A destinazioni

Destinazione di evento	Vantaggi dell'utilizzo di Amazon SNS
Amazon Data Firehose	Distribuire eventi ai flussi di distribuzione per scopi di archiviazione e analisi. Tramite i flussi di distribuzione, puoi distribuire eventi a AWS destinazioni come Amazon Simple Storage Service (Amazon S3), Amazon Redshift e

Destinazione di evento	Vantaggi dell'utilizzo di Amazon SNS
	OpenSearch Amazon Service OpenSearch (Service) o a destinazioni di terze parti come Datadog, New Relic, MongoDB e Splunk. Per ulteriori informazioni, consulta Flussi di distribuzione da Fanout a Firehose.
AWS Lambda	Distribuire eventi alle funzioni per attivare l'esecuzione della logica aziendale personalizzata. Per ulteriori informazioni, consulta Funzioni da Fanout a Lambda.
Amazon SQS	Distribuire gli eventi alle code per scopi di integrazione delle applicazioni. Per ulteriori informazioni, consulta Fan-out a code Amazon SQS.
AWS Pipeline Event Fork	Distribuisci eventi per il backup e lo storage degli eventi, la ricerca e l'analisi degli eventi o le pipeline di ripetizione degli eventi. Per ulteriori informazioni, consulta Da Fanout a AWS Pipeline Even Fork.
HTTP/S	Distribuire eventi a webhook esterni. Per ulteriori informazioni, consulta Fanout agli endpoint HTTP(S).

Destinazioni A2P

Destinazione di evento	Vantaggi dell'utilizzo di Amazon SNS
SMS	Consegnare eventi ai telefoni cellulari come messaggi di testo. Per ulteriori informazioni, consulta Messaggi di testo per dispositivi mobili (SMS).

Destinazione di evento	Vantaggi dell'utilizzo di Amazon SNS
E-mail	Consegna degli eventi nelle caselle di posta in arrivo come messaggi di posta. Per ulteriori informazioni, consulta Notifiche e-mail.
Endpoint platform	Distribuisci eventi ai telefoni cellulari come notifiche push native. Per ulteriori informazioni, consulta Notifiche push per dispositivi mobili.
AWS Chatbot	Consegna eventi alle chat room Amazon Chime o ai canali Slack. Per ulteriori informazioni, consulta le pagine seguenti nella AWS ChatbotGuida per l'amministratore: • Configurazione AWS Chatbot con Amazon Chime • Configurazione AWS Chatbot con Slack • Uso di AWS Chatbot con altri servizi AWS
PagerDuty	Fornisci informazioni operative ai team on-call. Per ulteriori informazioni, consulta Offri approfondimenti operativi basati su ML ai tuoi team di chiamata tramite PagerDuty Amazon DevOps Guru sul blog AWSManagement & Governance.

> **Note**
>
> È possibile fornire sia nativi AWS eventi che eventi personalizzati per le app chat:
>
> - Nativo AWS eventi – È possibile utilizzare AWS Chatbot per inviare nativo AWS eventi, attraverso gli argomenti di Amazon SNS, ad Amazon Chime e Slack. Il set di AWS eventi nativi supportato include eventi diAWS Billing and Cost Management,AWS Health, AWS

CloudFormation CloudWatch, Amazon e altro ancora. Per ulteriori informazioni, consulta Utilizzo di AWS Chatbot con altri servizi nella AWS Chatbot Guida per l'amministratore.

- Eventi personalizzati – Puoi anche inviare i tuoi eventi personalizzati, tramite gli argomenti di Amazon SNS, ad Amazon Chime, Slack e Microsoft Teams. A tale scopo, è possibile pubblicare eventi personalizzati in un argomento SNS, che fornisce gli eventi a una funzione Lambda sottoscritta. La funzione Lambda utilizza quindi il webhook dell'app di chat per consegnare gli eventi ai destinatari. Per ulteriori informazioni, consulta Come faccio a utilizzare i webhook per pubblicare messaggi Amazon SNS su Amazon Chime, Slack o Microsoft Teams?

Configurazione degli accessi per Amazon SNS

Prima di utilizzare Amazon SNS per la prima volta, devi completare la procedura seguente.

Argomenti

Fase 1: Creare un utente Account AWS e un utente IAM

Per accedere a qualsiasi AWS servizio, devi prima creare un Account AWS. Puoi utilizzare il tuo Account AWS per visualizzare i report sulle attività e sull'utilizzo e per gestire l'autenticazione e l'accesso.

Iscriviti per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Quando ti iscrivi a Account AWS, viene creato un utente Account AWS root. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come procedura consigliata in materia di sicurezza, assegnate l'accesso amministrativo a un utente e utilizzate solo l'utente root per eseguire attività che richiedono l'accesso da parte dell'utente root.

AWS ti invia un'e-mail di conferma dopo il completamento della procedura di registrazione. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo https://aws.amazon.com/ e selezionando Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi il tuo utente Account AWS root AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzarlo per le attività quotidiane.

Proteggi il tuo utente Account AWS root

1. Accedi AWS Management Consolecome proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Signing in as the root user della Guida per l'utente di Accedi ad AWS .
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per istruzioni, consulta Abilitare un dispositivo MFA virtuale per l'utente Account AWS root (console) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

   Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center .
2. In IAM Identity Center, concedi l'accesso amministrativo a un utente.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con le impostazioni predefinite IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accedi come utente con accesso amministrativo

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta AWS Accedere al portale di accesso nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso ad altri utenti

1. In IAM Identity Center, crea un set di autorizzazioni che segua la migliore pratica di applicazione delle autorizzazioni con privilegi minimi.

   Per istruzioni, consulta Creare un set di autorizzazioni nella Guida per l'utente.AWS IAM Identity Center
2. Assegna gli utenti a un gruppo, quindi assegna l'accesso Single Sign-On al gruppo.

   Per istruzioni, consulta Aggiungere gruppi nella Guida per l'utente.AWS IAM Identity Center

Passaggi successivi

Ora che sei pronto a utilizzare Amazon SNS, inizia creando un argomento, una sottoscrizione per l'argomento, pubblicando un messaggio nell'argomento ed eliminando la sottoscrizione e l'argomento.

Nozioni di base su Amazon SNS

Questa sezione consente di acquisire familiarità con Amazon SNS mostrando come gestire argomenti, sottoscrizioni e messaggi utilizzando la console Amazon SNS.

Argomenti

Prerequisiti

Prima di iniziare, completa i passaggi descritti in Configurazione degli accessi per Amazon SNS.

Fase 1: creazione di un argomento

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).
3. Nella pagina Topics (Argomenti), seleziona Create new topic (Crea nuovo argomento).
4. Per impostazione predefinita, la console crea un argomento FIFO. Scegliere Standard.
5. Nella sezione Dettagli, inserisci un nome per l'argomento, ad esempio `MyTopic`.
6. Vai in fondo al modulo e scegli Creare un argomento.

   La console apre la pagina nuovi argomenti Dettagli.

Fase 2: Creazione di una sottoscrizione per un argomento

1. Nel pannello di navigazione sinistro scegli Sottoscrizioni.
2. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.

3. Sulla pagina Creazione di una sottoscrizione, scegliere il campo ARN argomento per visualizzare l'elenco degli argomenti presenti nel Account AWS.
4. Scegliere laargomento creato nei passaggi precedenti.
5. Per Protocol, scegliere Email.
6. In Endpoint immetti l'indirizzo e-mail utilizzabile per ricevere le notifiche.
7. Scegli Create Subscription (Crea sottoscrizione).

   La console apre la nuova pagina di sottoscrizione Dettagli.
8. Controlla la tua casella di posta elettronica e scegli Conferma della sottoscrizione nella mail da AWS Notificazioni. L'ID mittente è in genere "no-reply@sns.amazonaws.com".
9. Amazon SNS apre il browser Web e visualizza una conferma dell'abbonamento con il tuo ID abbonamento.

Fase 3: pubblicazione di un messaggio nell'argomento

1. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).
2. Nella pagina Topics (Argomenti), scegliere l'argomento creato in precedenza quindi selezionare Publish message (Pubblica messaggio).

   La console apre la pagina Pubblica il messaggio nell'argomento.
3. (Facoltativo) Nella sezione Message details (Dettagli messaggio), immettere il Subject (Oggetto), ad esempio:

   ```
   Hello from Amazon SNS!
   ```
4. Nella sezione Corpo del messaggio, scegliere Payload identico per tutti i protocolli di consegna, quindi immettere un corpo del messaggio, ad esempio:

   ```
   Publishing a message to an SNS topic.
   ```
5. Seleziona Publish message (Pubblica messaggio).

   Il messaggio viene pubblicato nell'argomento e la console apre la pagina Dettagli.
6. Controlla la tua casella di posta elettronica e verifica di aver ricevuto un'e-mail da Amazon SNS con il messaggio pubblicato.

Fase 4: eliminazione della sottoscrizione e dell'argomento

1. Nel riquadro di navigazione, scegli Sottoscrizioni.
2. Nella pagina Sottoscrizione, scegli sottoscrizione confermata quindi scegli Elimina.

   > **Note**
   >
   > Non puoi eliminare una conferma in sospeso. Dopo 48 ore, Amazon SNS lo elimina automaticamente.

3. Nella finestra di dialogo Delete subscription (Elimina sottoscrizione), scegliere Delete (Elimina).

   La sottoscrizione viene eliminata.
4. Nel pannello di navigazione, scegliere Argomenti.
5. Nella pagina Topics (Argomenti), selezionare un argomento quindi scegliere Delete (Elimina).

   > **Important**
   >
   > Quando elimini un argomento, elimini anche tutte le relative sottoscrizioni.

6. Nella finestra di ***MyTopic***dialogo Elimina argomento, inserisci **`delete me`** e scegli Elimina.

   L'argomento viene eliminato.

Passaggi successivi

Dopo aver creato un argomento con una sottoscrizione e inviato messaggi all'argomento, potrebbe essere necessario provare a eseguire le operazioni seguenti:

- Esplora il AWS Centro sviluppatori.
- Ulteriori informazioni sulla protezione dei dati nella sezione Sicurezza.
- Abilitare la crittografia lato server per un argomento.
- Abilitare la crittografia lato server per un argomento con un coda Amazon Simple Queue Service (Amazon SQS) crittografata sottoscritto.
- Sottoscrivere AWS Pipeline Event Fork a un argomento.

Configurazione Amazon SNS

Utilizzo della Console Amazon SNS per creare e configurare gli argomenti e gli abbonamenti di Amazon SNS. Per ulteriori informazioni su Amazon SNS, consulta la Che cos'è Amazon SNS?.

Argomenti

Creare un argomento Amazon SNS

Un argomento Amazon SNS è un punto di accesso logico che funge da canale di comunicazione. Un argomento consente di raggruppare più endpoint (ad esempio Amazon SQS AWS Lambda, HTTP/S o un indirizzo e-mail).

Per trasmettere i messaggi di un sistema producer (ad esempio un sito Web di E-Commerce) e che opera con più servizi diversi dai quali tali messaggi sono richiesti (ad esempio i sistemi di pagamento ed evasione degli ordini), puoi creare un argomento per il sistema producer.

La prima attività di Amazon SNS, nonché la più comune, è la creazione di un argomento. Questa pagina mostra come utilizzare il AWS Management Console, il e il AWS SDK for Java AWS SDK for .NET per creare un argomento.

Durante la creazione, è possibile scegliere un tipo di argomento (standard o FIFO) e assegnare un nome all'argomento. Una volta creato un argomento, non è possibile modificare il tipo o il nome dell'argomento. Tutte le altre opzioni di configurazione sono facoltative durante la creazione dell'argomento ed è possibile modificarle in un secondo momento.

> **Important**
>
> Non aggiungere informazioni personali di identificazione (PII) o altre informazioni riservate o sensibili nei nomi dei argomenti. I nomi degli argomenti sono accessibili ad altri Amazon Web Services, inclusi CloudWatch i log. I nomi dei argomenti non sono destinati ad essere utilizzati per dati privati o sensibili.

Argomenti

Per creare un argomento utilizzando il AWS Management Console

1. Accedi alla console Amazon SNS.
2. Esegui una di queste operazioni:
   - Se non è mai stato creato alcun argomento in Account AWS precedenza, leggi la descrizione di Amazon SNS nella home page.
   - Se gli argomenti sono già stati creati sotto il tuo Account AWS nome, nel pannello di navigazione, scegli Argomenti.
3. Nella pagina Topics (Argomenti), seleziona Create new topic (Crea nuovo argomento).
4. Nella pagina Create topic (Crea argomento), nella sezione Details (Dettagli), eseguire queste operazioni:
   a. Per Type (Tipo), scegli un tipo di argomento (standard o FIFO).
   b. Immetti un nome per l'argomento. Per un Argomento FIFO, aggiungere .fifo alla fine del nome.
   c. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.

      > **Important**
      >
      > Quando si effettua la sottoscrizione di un endpoint e-mail, il numero combinato di caratteri del nome dell'argomento Amazon SNS mostrato e dell'indirizzo e-mail da cui viene eseguito l'invio (ad esempio, no-reply@sns.amazonaws.com) non deve essere superiore a 320 caratteri UTF-8. È possibile utilizzare uno strumento di codifica di terze parti per verificare la lunghezza dell'indirizzo da cui eseguire l'invio prima di configurare un nome da mostrare per l'argomento Amazon SNS.

   d. (Facoltativo) Per un argomento FIFO, puoi scegliere Deduplicazione dei messaggi basata sul contenuto per abilitare la deduplicazione predefinita dei messaggi. Per ulteriori informazioni, consulta Deduplicazione dei messaggi per argomenti FIFO.

5. (Facoltativo) Espandere la sezione Encryption (Crittografia) e procedere come segue. Per ulteriori informazioni, consulta Crittografia a riposo.

   a. Scegliere Enable encryption (Abilita crittografia).

   b. Specificate la AWS KMS chiave. Per ulteriori informazioni, consulta Termini chiave.

   Per ciascun tipo di KMS vengono visualizzati i valori Description (Descrizione), Account e KMS ARN (ARN KMS).

   > **Important**
   >
   > Se non sei il proprietario della KMS, oppure se effettui l'accesso con un account che non dispone delle autorizzazioni `kms:ListAliases` e `kms:DescribeKey`, non puoi visualizzare le informazioni sulla KMS sulla console Amazon SNS.
   > Chiedi al proprietario della KMS di concederti queste autorizzazioni. Per esempi e ulteriori informazioni consulta AWS KMS Autorizzazioni API: e Documentazione su operazioni e risorse nella AWS Key Management Service Guida per sviluppatori.

   - Il KMS AWS gestito per Amazon SNS (predefinito) alias/aws/sns è selezionato per impostazione predefinita.

     > **Note**
     >
     > Ricorda quanto segue:
     >
     > - La prima volta che usi AWS Management Console per specificare il KMS AWS gestito per Amazon SNS per un argomento AWS KMS , crea AWS il KMS gestito per Amazon SNS.
     > - In alternativa, la prima volta che utilizzi l'`Publish`azione su un argomento con SSE abilitato, AWS KMS crea il KMS AWS gestito per Amazon SNS.

   - Per utilizzare un KMS personalizzato dal tuo AWS account, scegli il campo chiave KMS, quindi scegli il KMS personalizzato dall'elenco.

> **Note**
>
> Per istruzioni sulla creazione di KMS personalizzate, consulta Creazione di chiavi nella Guida per gli sviluppatori di AWS Key Management Service

- Per utilizzare un ARN KMS personalizzato dal AWS tuo account o da AWS un altro account, inseriscilo nel campo della chiave KMS.

6. (Facoltativo) Per impostazione predefinita, solo il proprietario dell'argomento può pubblicare o sottoscrivere l'argomento. Per configurare autorizzazioni di accesso aggiuntive, espandi la sezione Access policy (Policy di accesso) . Per ulteriori informazioni, consulta Identity and Access Management in Amazon SNS e Esempi di casi per il controllo degli accessi Amazon SNS.

> **Note**
>
> Quando crei un argomento utilizzando la console, la policy predefinita utilizza la chiave di condizione `aws:SourceOwner`. Questa chiave è analoga a `aws:SourceAccount`.

7. (Facoltativo) Per configurare il modo in cui Amazon SNS riprova a consegnare i messaggi non recapitati, espandere la sezione Delivery retry policy (HTTP/S) (Policy nuovi tentativi di consegna [HTTP/S]). Per ulteriori informazioni, consulta Tentativi di consegna dei messaggi di Amazon SNS.
8. (Facoltativo) Per configurare il modo in cui Amazon SNS registra la consegna dei messaggi CloudWatch, espandi la sezione Registrazione dello stato di consegna. Per ulteriori informazioni, consulta Stato di consegna dei messaggi Amazon SNS.
9. (facoltativo) Per aggiungere tag di metadati all'argomento, espandere la sezione Tags (Tag), immettere una Key (Chiave) e un Value (Valore) (opzionale) e scegliere Add tag (Aggiungi tag). Per ulteriori informazioni, consulta Assegnazione di tag all'argomento Amazon SNS.
10. Scegliere Create topic (Crea argomento).

    L'argomento viene creato e la ***MyTopic***pagina viene visualizzata.

    Il nome dell'argomento, l'ARN, (opzionale) il nome visualizzato e l'ID AWS account del proprietario dell'argomento vengono visualizzati nella sezione Dettagli.
11. Copiare l'ARN dell'argomento negli appunti, ad esempio:

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Per creare un argomento utilizzando un SDK AWS

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

I seguenti esempi di codice mostrano come utilizzare. `CreateTopic`

.NET

AWS SDK for .NET

> **Note**
>
> C'è 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 argomento con un nome di specifico.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();
```

```
            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Crea un nuovo argomento con un nome e attributi FIFO e di deduplicazione specifici.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
```

```
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for C++ API Reference.

CLI

AWS CLI

Creazione di un argomento SNS

Nell'esempio `create-topic` seguente viene creato un argomento SNS denominato `my-topic`.

```
aws sns create-topic \
    --name my-topic
```

Output:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Per ulteriori informazioni, consulta Using the AWS Command Line Interface with Amazon SQS e Amazon SNS nella Command Line Interface AWS User Guide.

- Per i dettagli sull'API, consulta AWS CLI Command CreateTopicReference.

Go

SDK per Go V2

> **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.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
```

```
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for Go API Reference.

Java

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.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, CreateTopicconsulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- Per i dettagli sull'API, consulta CreateTopic AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **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 class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **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.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Per i dettagli sulle API, consulta il riferimento CreateTopicall'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
 is returned for testing purposes. "
        MESSAGE 'SNS topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
```

```
        MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Per i dettagli sulle API, CreateTopicconsulta AWS SDK for SAP ABAP API reference.

Iscrizione a un argomento Amazon SNS

Per ricevere i messaggi pubblicati in un argomento devi effettuare la sottoscrizione di un endpoint all'argomento specificato. Una volta effettuata l'iscrizione di un endpoint a un argomento, l'endpoint inizia a ricevere tutti i messaggi pubblicati nell'argomento associato.

> **Note**
>
> Gli endpoint HTTP, gli indirizzi e-mail e le risorse AWS in altri Account AWS richiedono la conferma dell'abbonamento prima che possano ricevere messaggi.

Per iscrivere un endpoint a un argomento Amazon SNS

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione sinistro scegli Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:

   a. Per argomento ARN, scegli l'Amazon Resource Name (ARN) di un argomento. Questo valore è l'ARN AWS generato quando hai creato l'argomento Amazon SNS, ad esempio `arn:aws:sns:us-east-2:123456789012:your_topic`.

   b. Per Protocollo, scegli un tipo di endpoint. I tipi di endpoint disponibili sono:

      - HTTP/HTTPS
      - Email/Email-JSON
      - Amazon Data Firehose
      - Amazon SQS

> **Note**
>
> Per sottoscrivere ad un argomento SNS FIFO scegli questa opzione.

- AWS Lambda
- Endpoint applicazione piattaforma
- SMS

c. Per Endpoint, inserisci il valore dell'endpoint, ad esempio un indirizzo e-mail o l'ARN di una coda Amazon SQS.

d. Solo endpoint Firehose: per il ruolo di sottoscrizione ARN, specifica l'ARN del ruolo IAM che hai creato per la scrittura nei flussi di distribuzione Firehose. Per ulteriori informazioni, consulta Prerequisiti per la sottoscrizione dei flussi di distribuzione di Firehose agli argomenti di Amazon SNS.

e. (Facoltativo) Per gli endpoint Firehose, Amazon SQS, HTTP/S, puoi anche abilitare il recapito di messaggi non elaborati. Per ulteriori informazioni, consulta Consegna di messaggi non elaborati Amazon SNS.

f. (Facoltativo) Per configurare un criterio di filtro, espandere la sezione policy di filtro della sottoscrizione. Per ulteriori informazioni, consulta Policy di filtro degli abbonamenti Amazon SNS.

g. (Facoltativo) Per abilitare il filtro basato sul payload, imposta `Filter Policy Scope` su `MessageBody`. Per ulteriori informazioni, consulta Ambito delle policy di filtro per le sottoscrizioni Amazon SNS.

h. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS.

i. Scegli Create Subscription (Crea sottoscrizione).

   La console crea la sottoscrizione e apre la pagina dettagli della sottoscrizione.

Eliminazione di un argomento e di un abbonamento Amazon SNS

Quando un argomento viene eliminato, le sottoscrizioni associate vengono eliminate in modo asincrono. Sebbene i clienti possano ancora accedere a questi abbonamenti, gli abbonamenti non sono più associati all'argomento, anche se si ricrea l'argomento utilizzando lo stesso nome.

Se un abbonato tenta di pubblicare un messaggio sull'argomento eliminato, l'editore riceverà un messaggio di errore che indica che l'argomento non esiste. Analogamente, qualsiasi tentativo di abbonamento all'argomento eliminato genererà anche un messaggio di errore.

Non è possibile eliminare un abbonamento in attesa di conferma. Dopo 48 ore, Amazon SNS elimina automaticamente gli abbonamenti non confermati.

Argomenti

Per eliminare un argomento o un abbonamento Amazon SNS utilizzando AWS Management Console

Per eliminare un argomento utilizzando il AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).
3. Nella pagina Topics (Argomenti), selezionare un argomento e scegliere Elimina.
4. Nella finestra di dialogo Delete topic (Elimina argomento), digitare `delete me`, quindi selezionare Delete (Elimina).

   La console elimina l'argomento.

Per eliminare un abbonamento utilizzando il AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione sinistro scegli Sottoscrizioni.
3. Nella pagina Abbonamenti, seleziona un abbonamento con lo stato Confermato, quindi scegli Elimina.
4. Nella finestra di dialogo Delete subscription (Elimina iscrizione), scegliere Delete (Elimina).

   La console elimina la sottoscrizione.

Per eliminare una sottoscrizione e un argomento tramite un AWS SDK

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

I seguenti esempi di codice mostrano come utilizzare. `DeleteTopic`

.NET

AWS SDK for .NET

> **Note**
>
> C'è 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 argomento in base all'ARN dell'argomento.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Eliminazione di un argomento SNS

Nell'esempio `delete-topic` seguente viene eliminato l'argomento SNS specificato.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta DeleteTopic AWS CLICommand Reference.

Go

SDK per Go V2

> **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.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
```

```
 }
 return err
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for Go API Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
```

```
  // }
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Per i dettagli sull'API, DeleteTopicconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Per i dettagli sull'API, consulta DeleteTopic AWSSDK for Python (Boto3) API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, DeleteTopicconsulta AWS SDK for SAP ABAP API reference.

Assegnazione di tag all'argomento Amazon SNS

Amazon SNS supporta l'assegnazione di tag agli argomenti Amazon SNS. Ciò può aiutare a monitorare e gestire i costi associati ai tuoi argomenti e fornisce una maggiore sicurezza nelle tue policy Identity and Access Management (IAM) AWS e consente di cercare o filtrare facilmente migliaia di argomenti. L'assegnazione di tag ti consente di gestire gli argomenti Amazon SNS utilizzando AWS Resource Groups. Per ulteriori informazioni sui Resource Groups, consulta la Guida per l'utente dei Resource Groups AWS.

Argomenti

- Assegnazione di tag per l'allocazione dei costi
- Assegnazione di tag per il controllo degli accessi
- Assegnazione di tag per la ricerca e il filtro delle risorse
- Configurare i tag dell'argomento Amazon SNS

Assegnazione di tag per l'allocazione dei costi

Per organizzare e identificare gli argomenti Amazon SNS per l'allocazione dei costi, puoi aggiungere tag che identificano lo scopo di un argomento. Questo è particolarmente utile in presenza di molti argomenti. Per organizzare le fatture AWS al fine di riflettere la struttura dei costi, puoi utilizzare i tag di allocazione dei costi. Per eseguire questa operazione, registrati per far sì che la fattura del tuo account AWS includa le chiavi e i valori di tag. Per ulteriori informazioni, consulta Impostazione di un report di allocazione dei costi mensili nella Guida per l'utente di gestione fatturazione e costi di AWS.

Ad esempio, puoi aggiungere tag che rappresentano il centro di costo e lo scopo degli argomenti Amazon SNS, come indicato di seguito:

Risorsa	Key (Chiave)	Value (Valore)
Argomento 1	Centro costi	43289
	Applicazione	Elaborazione di ordini
Argomento 2	Centro costi	43289
	Applicazione	Elaborazione dei pagamenti
Argomento 3	Centro costi	76585
	Applicazione	Archiviazione

Questo schema di assegnazione di tag consente di raggruppare due argomenti che eseguono processi correlati nello stesso centro di costo, mentre si esegue l'assegnazione di tag a un'attività non pertinente con un tag di allocazione dei costi diverso.

Assegnazione di tag per il controllo degli accessi

AWS Identity and Access Management supporta il controllo dell'accesso alle risorse in base ai tag. Dopo aver taggato le risorse, devi fornire informazioni sui tag delle risorse nell'elemento condizione di una policy IAM per gestire gli accessi basati su tag. Per informazioni su come assegnare i tag alle risorse utilizzando la Console Amazon SNS o il SDK AWS, consultare Configurazione dei tag.

È possibile limitare l'accesso per un'identità IAM. Ad esempio, è possibile limitare l'accesso `Publish` e `PublishBatch` a tutti gli argomenti Amazon SNS che includono un tag con la chiave

`environment` e il valore `production`, consentendo al contempo l'accesso a tutti gli altri argomenti Amazon SNS. Nell'esempio riportato di seguito, la policy limita la possibilità di pubblicare messaggi su argomenti con taggati con `production`, consentendo al contempo la pubblicazione di messaggi su argomenti taggati con `development`. Per ulteriori informazioni, consulta Controllo dell'accesso tramite tag nella Guida per l'utente di IAM.

> **Note**
>
> Impostare l'autorizzazione IAM per `Publish` imposta l'autorizzazione sia per `Publish` che per `PublishBatch`.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": [
 "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
}
```

Assegnazione di tag per la ricerca e il filtro delle risorse

Un account AWS può avere decine di migliaia di argomenti Amazon SNS (vedere Quote Amazon SNS per i dettagli). Assegnando tag ai tuoi argomenti, puoi semplificare il processo di ricerca o filtro degli argomenti.

Ad esempio, è possibile disporre di centinaia di argomenti associati all'ambiente di produzione. Anziché dover cercare manualmente questi argomenti, è possibile interrogare tutti gli argomenti con un determinato tag:

```
import com.amazonaws.services.resourcegroups.AWSResourceGroups;
import com.amazonaws.services.resourcegroups.AWSResourceGroupsClientBuilder;
import com.amazonaws.services.resourcegroups.model.QueryType;
import com.amazonaws.services.resourcegroups.model.ResourceQuery;
import com.amazonaws.services.resourcegroups.model.SearchResourcesRequest;
import com.amazonaws.services.resourcegroups.model.SearchResourcesResult;

public class Example {
    public static void main(String[] args) {
        // Query Amazon SNS Topics with tag "keyA" as "valueA"
        final String QUERY = "{\"ResourceTypeFilters\":[\"AWS::SNS::Topic\"],
\"TagFilters\":[{\"Key\":\"keyA\", \"Values\":[\"valueA\"]}]}";

        // Initialize ResourceGroup client
        AWSResourceGroups awsResourceGroups = AWSResourceGroupsClientBuilder
            .standard()
            .build();

        // Query all resources with certain tags from ResourceGroups
        SearchResourcesResult result = awsResourceGroups.searchResources(
            new SearchResourcesRequest().withResourceQuery(
                new ResourceQuery()
                .withType(QueryType.TAG_FILTERS_1_0)
                .withQuery(QUERY)
            ));
        System.out.println("SNS Topics with certain tags are " +
 result.getResourceIdentifiers());
    }
}
```

Configurare i tag dell'argomento Amazon SNS

Questa pagina mostra come utilizzare AWS Management Console, un AWS SDK e la AWS CLI per configurare i tag per un argomento di Amazon SNS.

> **Important**
>
> Non aggiungere Informazioni personali di identificazione (PII) o altre informazioni riservate o sensibili nei tag. I tag sono accessibili ad altri Amazon Web Services, inclusa la fatturazione. I tag non sono destinati ad essere utilizzati per dati privati o sensibili.

Argomenti

Elencare, aggiungere e rimuovere tag per un argomento di Amazon SNS utilizzando AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti), selezionare un argomento quindi scegliere Edit (Modifica).
4. Espandere la sezione Tag.

   Vengono elencati i tag aggiunti all'argomento.
5. Modificare i tag dell'argomento:

   - Per aggiungere un tag, scegliere Add tag (Aggiungi tag) e specificare Key (Chiave) e Value (Valore) (opzionale),
   - Per rimuovere un tag, scegliere Remove tag (Rimuovi tag) accanto a una coppia chiave-valore.
6. Seleziona Salvataggio delle modifiche.

Aggiunta di tag a un argomento utilizzando un SDK AWS

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

I seguenti esempi di codice mostrano come utilizzare. `TagResource`

CLI

AWS CLI

Aggiungere un tag a un argomento

Nell'esempio `tag-resource` seguente viene aggiunto un tag di metadati all'argomento Amazon SNS specificato.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta TagResource AWS CLICommand Reference.

Java

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.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.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
```

```
    }

    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Per i dettagli sull'API, TagResourceconsulta AWS SDK for Kotlin API reference.

Gestione dei tag con le operazioni dell’API Amazon SNS

Per gestire i tag utilizzando l’API Amazon SNS, utilizza le seguenti operazioni API:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

Azioni API che supportano ABAC

Di seguito è riportato un elenco di operazioni API che supportano il controllo di accesso basato su attributi (ABAC). Per maggiori dettagli su ABAC, vedi A cosa serve ABAC? AWS nella Guida per l'utente di IAM.

- `AddPermission`

- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

Ordinamento e deduplicazione dei messaggi (argomenti FIFO)

Puoi utilizzare gli argomenti FIFO di Amazon SNS (first in, first out) con Code FIFO di Amazon SQS per fornire un ordinamento rigoroso dei messaggi e la deduplicazione dei messaggi. Le funzionalità FIFO di ciascuno di questi servizi lavorano insieme per fungere da servizio completamente gestito per integrare applicazioni distribuite che richiedono coerenza dei dati in tempo quasi reale. La sottoscrizione di code standard di Amazon SQS agli argomenti FIFO di Amazon SNS fornisce il miglior ordine possibile e la consegna almeno una volta.

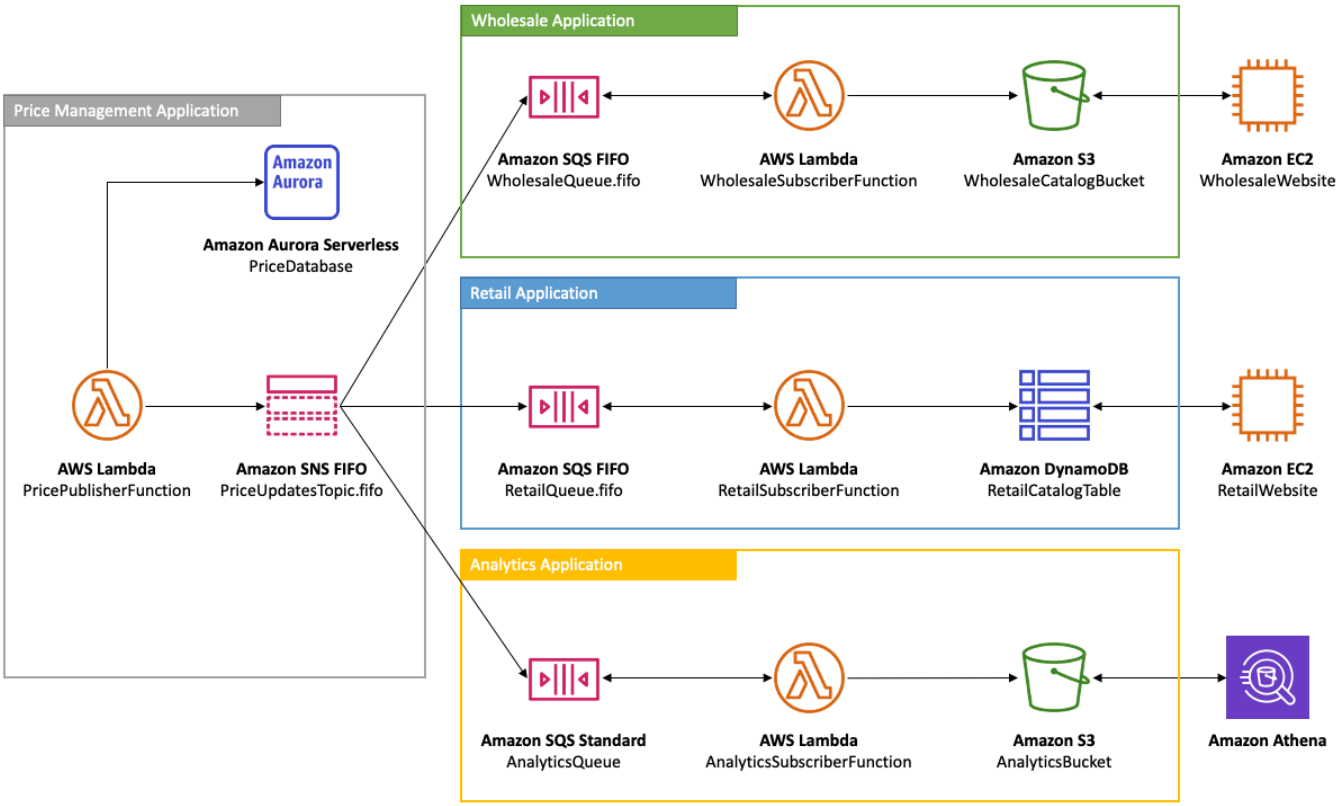
Argomenti

Esempio di caso d'uso di argomenti FIFO

L'esempio seguente descrive una piattaforma di e-commerce creata da un produttore di ricambi auto utilizzando gli argomenti FIFO di Amazon SNS e le code di Amazon SQS. La piattaforma è composta da quattro applicazioni serverless:

- I responsabili dell'inventario utilizzano un'applicazione di gestione dei prezzi per impostare il prezzo per ogni articolo in magazzino. In questa azienda, i prezzi dei prodotti possono cambiare in base alla fluttuazione del cambio di valuta, alla domanda del mercato e ai cambiamenti nella strategia di vendita. L'applicazione di gestione dei prezzi utilizza una funzione AWS Lambda che pubblica gli aggiornamenti dei prezzi in un argomento FIFO di Amazon SNS ogni volta che i prezzi cambiano.

- Un'applicazione all'ingrosso fornisce il backend per un sito web in cui carrozzerie auto e produttori di automobili possono acquistare parti auto della società in massa. Per ottenere le notifiche di modifica del prezzo, l'applicazione all'ingrosso sottoscrive la coda FIFO SQS all'argomento FIFO di Amazon SNS dell'applicazione di gestione dei prezzi.
- Un'applicazione di vendita al dettaglio fornisce il backend per un altro sito web in cui i proprietari di auto e gli appassionati di tuning auto possono acquistare singoli pezzi di auto per i loro veicoli. Per ottenere le notifiche di modifica del prezzo, l'applicazione di vendita al dettaglio sottoscrive anche la coda FIFO di Amazon SQS all'argomento FIFO di Amazon SNS dell'applicazione di gestione dei prezzi.
- Un'applicazione di analisi che aggrega gli aggiornamenti dei prezzi e li archivia in un bucket Amazon S3, consentendo ad Amazon Athena di eseguire query sul bucket per scopi di business intelligence (BI). Per ottenere le notifiche di modifica del prezzo, l'applicazione di analisi sottoscrive la coda standard Amazon SQS all'argomento FIFO di Amazon SNS dell'applicazione di gestione dei prezzi. A differenza delle altre applicazioni, quella di analisi non richiede che gli aggiornamenti dei prezzi siano ordinati rigorosamente.

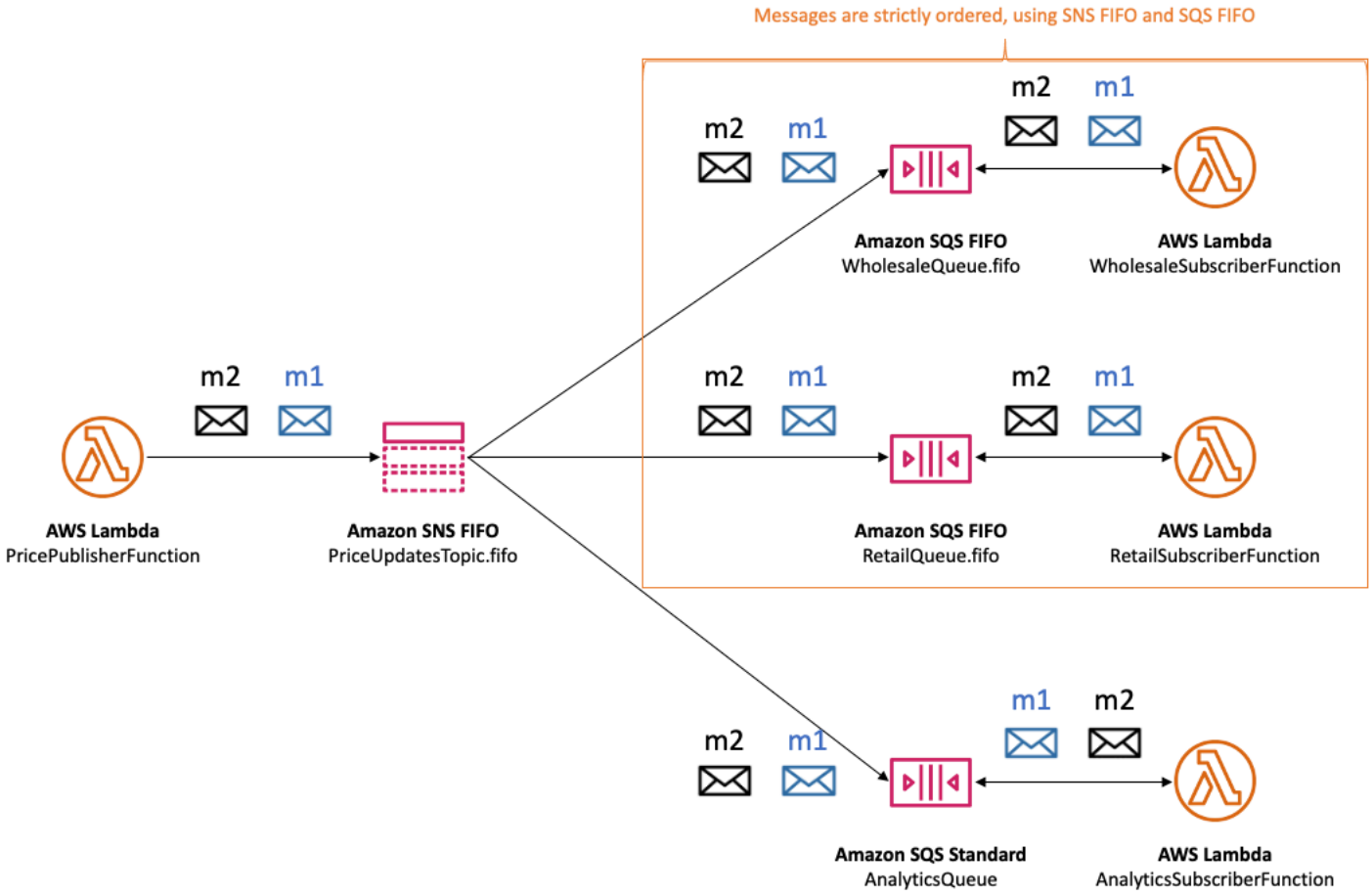


Affinché le applicazioni all'ingrosso e al dettaglio ricevano gli aggiornamenti dei prezzi nell'ordine corretto, l'applicazione di gestione dei prezzi deve utilizzare un sistema di distribuzione dei messaggi

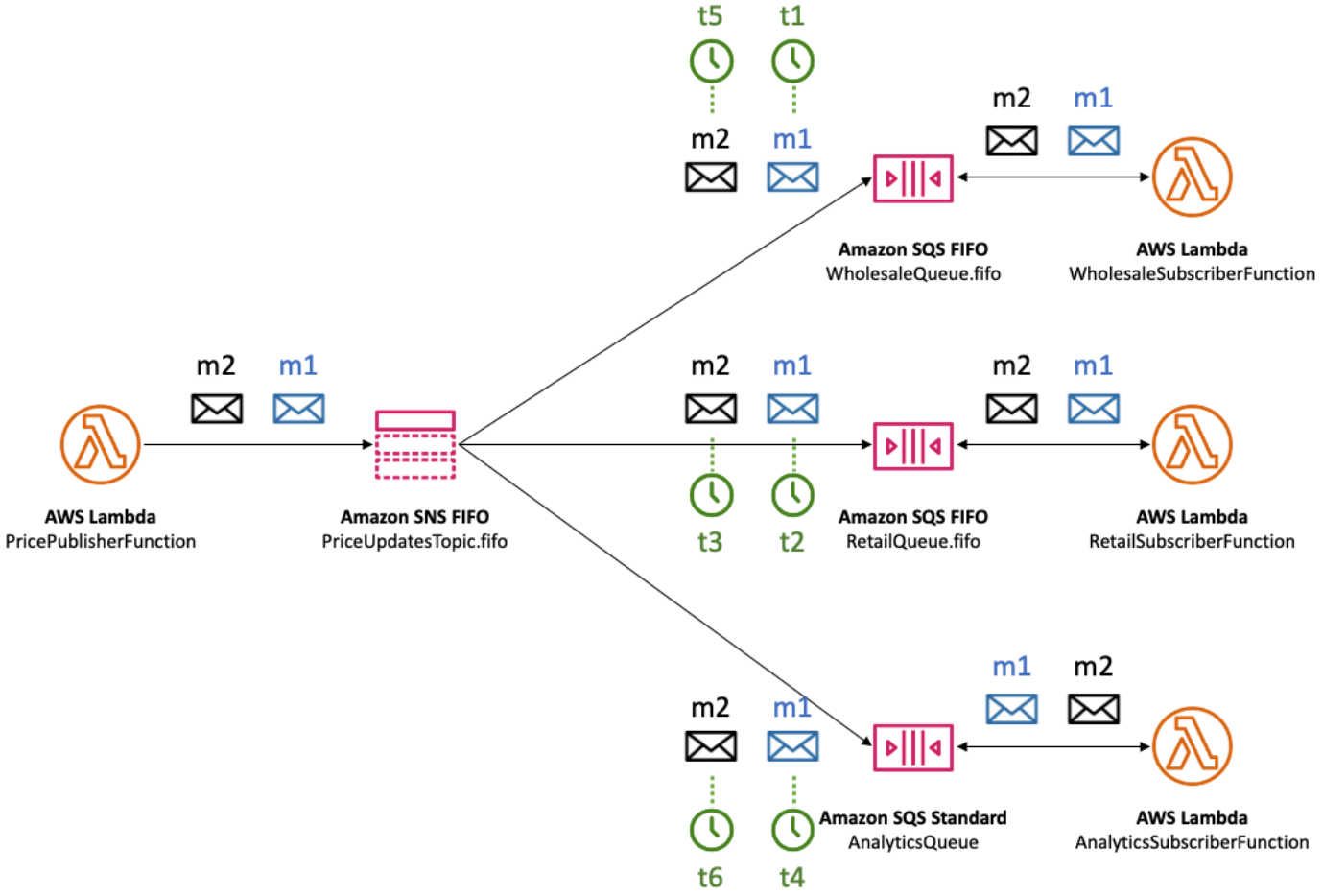
rigorosamente ordinato. L'utilizzo di argomenti FIFO di Amazon SNS e code FIFO di Amazon SQS consente l'elaborazione dei messaggi in ordine e senza duplicazione. Per ulteriori informazioni, consulta Dettagli dell'ordine dei messaggi per gli argomenti FIFO. Per gli snippet di codice che implementano questo caso d'uso, vedere Esempi di codice per argomenti FIFO.

Dettagli dell'ordine dei messaggi per gli argomenti FIFO

Un argomento FIFO di Amazon SNS invia sempre messaggi alle code di Amazon SQS sottoscritte nell'ordine esatto in cui i messaggi vengono pubblicati nell'argomento e una sola volta. Con una coda FIFO di Amazon SQS sottoscritta, l'utente della coda riceve i messaggi nell'ordine esatto in cui questi vengono consegnati alla coda e senza duplicati. Con una coda standard di SQS sottoscritta, tuttavia, l'utente della coda può ricevere messaggi non in ordine e più di una volta. Ciò consente un ulteriore disaccoppiamento degli abbonati dagli editori, offrendo agli abbonati una maggiore flessibilità in termini di consumo di messaggi e di ottimizzazione dei costi, come mostrato nel diagramma seguente, basato su Esempio di caso d'uso di argomenti FIFO.

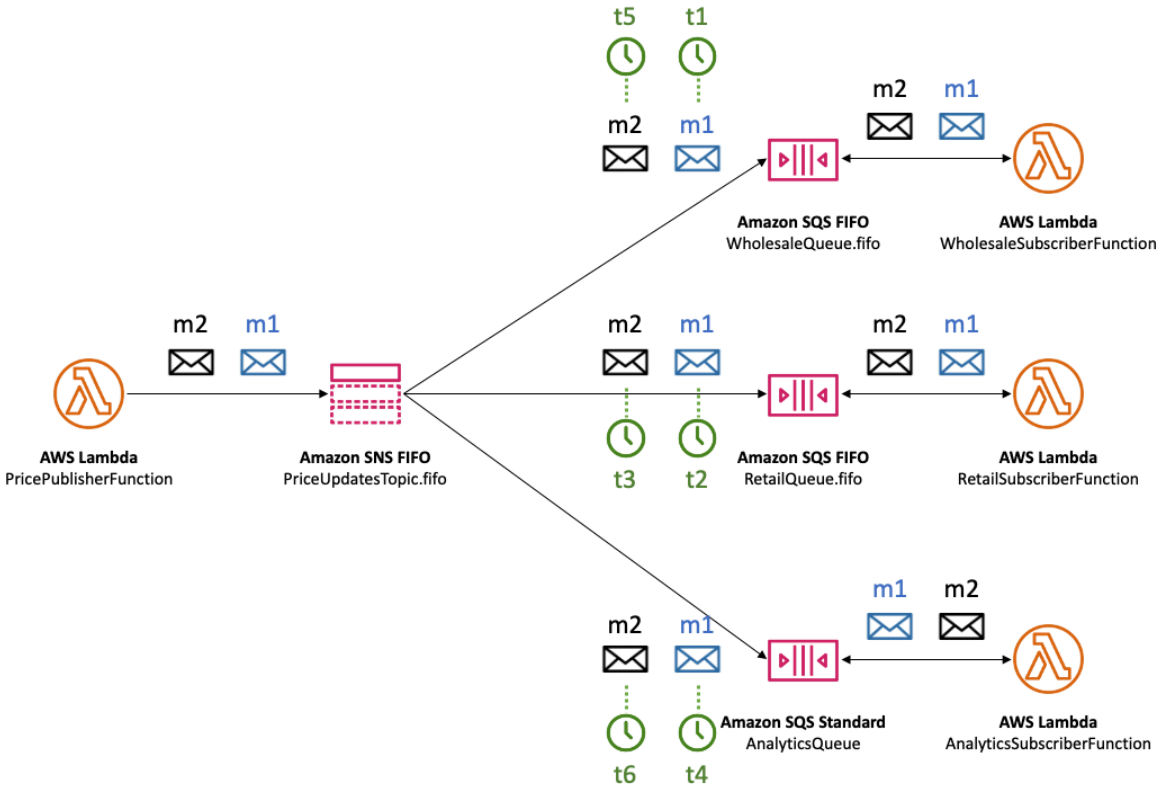


Si noti che non vi è alcun ordinamento implicito degli abbonati. L'esempio seguente ne è un'illustrazione m1 viene consegnato prima all'abbonato all'ingrosso, poi all'abbonato al dettaglio e quindi all'abbonato di analisi. Messaggio m2 viene consegnato prima all'abbonato al dettaglio, poi all'abbonato all'ingrosso e infine all'abbonato di analisi. Sebbene i due messaggi vengano recapitati agli abbonati in un ordine diverso, l'ordinamento dei messaggi viene mantenuto per ogni abbonato FIFO di Amazon SQS. Ogni abbonato è percepito in isolamento da qualsiasi altro abbonato.



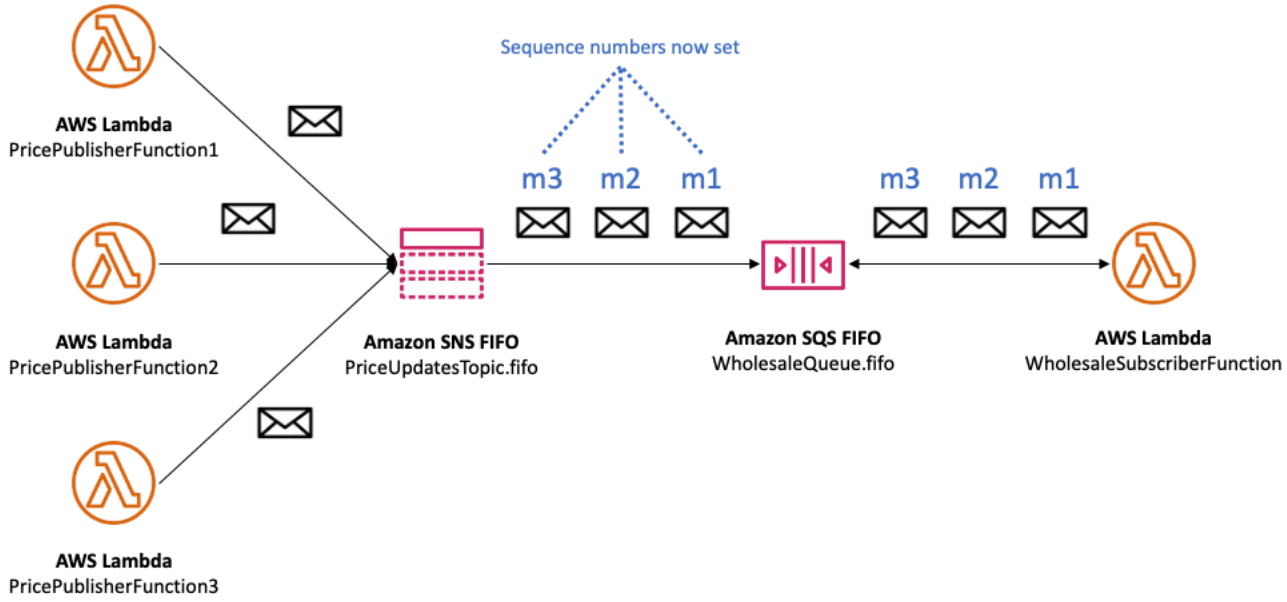
Se un sottoscrittore della coda di Amazon SQS diventa irraggiungibile, può uscire dalla sincronizzazione. Ad esempio, supponiamo che il proprietario della coda dell'applicazione all'ingrosso cambi erroneamente il Policy Coda Amazon SQS in modo da impedire all'entità servizio Amazon SNS di recapitare messaggi alla coda. In questo caso, le consegne degli aggiornamenti dei prezzi alla coda all'ingrosso non vanno a buon fine, mentre quelle alle code di vendita al dettaglio e di analisi hanno esito positivo, causando la mancata sincronizzazione degli abbonati. Quando il proprietario della coda delle applicazioni all'ingrosso corregge la relativa policy della coda, Amazon SNS riprende a consegnare i messaggi alla coda sottoscritta. Gli eventuali messaggi pubblicati

nell'argomento che hanno come target la coda configurata in modo errato vengono eliminati, a meno che la sottoscrizione corrispondente non disponga di un coda DLQ configurata.



È possibile disporre di più applicazioni (o più thread all'interno della stessa applicazione) che pubblicano messaggi in un argomento FIFO SNS in parallelo. Quando si esegue questa operazione, è possibile delegare in modo efficace il sequenziamento dei messaggi al servizio di Amazon SNS. Per determinare la sequenza stabilita di messaggi, è possibile controllare il numero di sequenza.

Il numero di sequenza è un numero grande, non consecutivo, che Amazon SNS assegna a ciascun messaggio. La lunghezza del numero di sequenza è di 128 bit e continua ad aumentare per ogni gruppo di messaggi. Il numero di sequenza viene passato alle code FIFO di Amazon SQS sottoscritte come parte del corpo del messaggio. Tuttavia, se si abilita Consegna di messaggi non elaborati, il messaggio recapitato alla coda di Amazon SQS non include il numero di sequenza o altri metadati del messaggio Amazon SNS.



Gli argomenti FIFO di Amazon SNS definiscono l'ordine nel contesto di un gruppo di messaggi. Per ulteriori informazioni, consulta Raggruppamento di messaggi per argomenti FIFO.

Raggruppamento di messaggi per argomenti FIFO

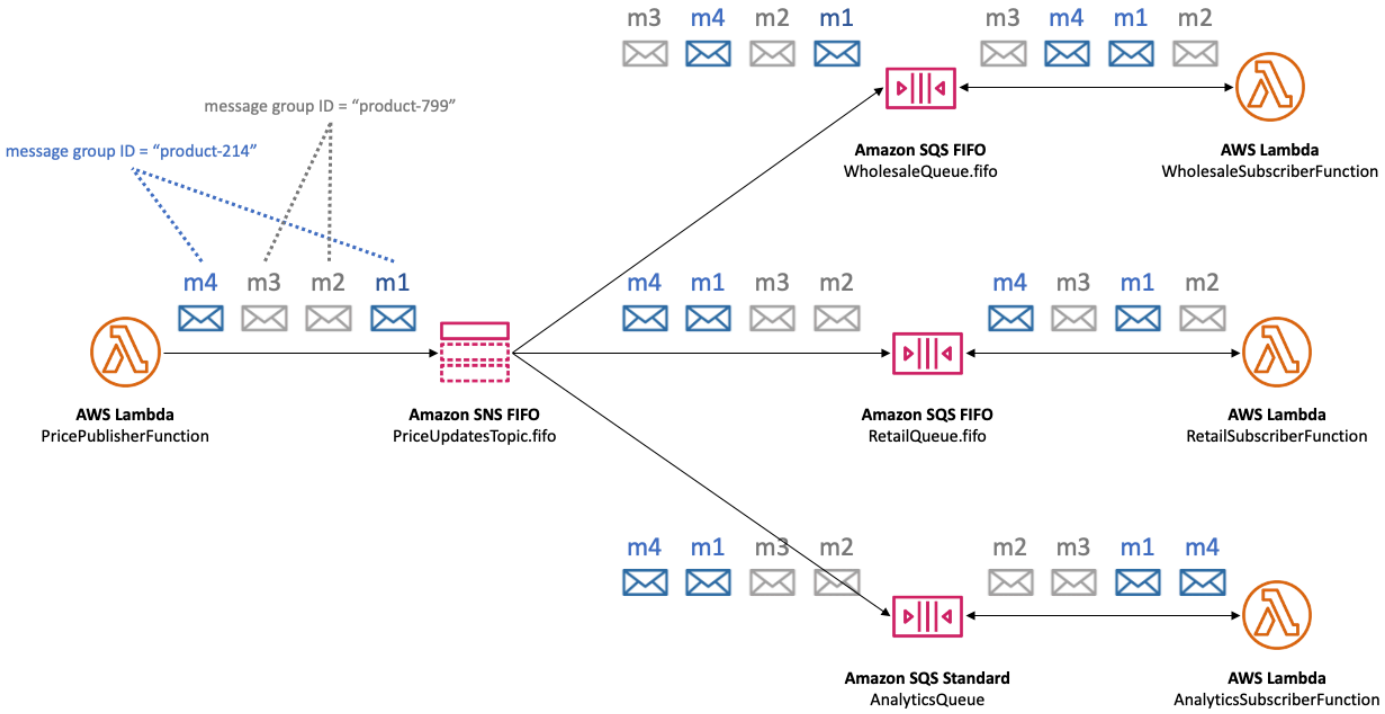
I messaggi che appartengono allo stesso gruppo vengono elaborati uno per uno, in un ordine rigoroso rispetto al gruppo.

Quando pubblichi messaggi su un argomento FIFO di Amazon SNS, imposti l'ID gruppo di messaggi. L'ID gruppo è un token obbligatorio che specifica che un messaggio appartiene a un gruppo di messaggi specifico. L'argomento FIFO SNS passa l'ID gruppo alle code FIFO di Amazon SQS sottoscritte. Non vi è alcun limite al numero di ID di gruppo negli argomenti SNS FIFO o nelle code FIFO SQS. L'ID del gruppo di messaggi non viene passato alle code standard di Amazon SQS.

Non esiste affinità tra un gruppo di messaggi e una sottoscrizione. Pertanto, i messaggi pubblicati in qualsiasi gruppo di messaggi vengono recapitati a tutte le code sottoscritte, in base alle policy di filtro associati alle sottoscrizioni. Per ulteriori informazioni, consulta Consegna dei messaggi per gli argomenti FIFO e Filtro dei messaggi per argomenti FIFO.

Nel caso d'uso esempio di gestione dei prezzi delle parti auto, c'è un gruppo di messaggi dedicato per ogni prodotto venduto nella piattaforma. Lo stesso argomento FIFO di Amazon SNS viene utilizzato per l'elaborazione di tutti gli aggiornamenti dei prezzi. La sequenza di aggiornamenti dei prezzi viene mantenuta nel contesto di un singolo prodotto di ricambi auto, ma no su più prodotti. Il seguente diagramma ne mostra il funzionamento. Si noti che, per il prodotto il cui ID gruppo

di messaggi è product-214, il messaggio m1 viene elaborato prima del messaggio m4. Questa sequenza viene mantenuta per tutti i flussi di lavoro che utilizzano FIFO di Amazon SNS in FIFO di Amazon SQS. Allo stesso modo, per il prodotto il cui ID del gruppo di messaggi è product-799, il messaggio m2 viene elaborato prima del messaggio m3, a condizione che i flussi di lavoro utilizzino FIFO di Amazon SNS e FIFO di Amazon SQS. Tuttavia, quando si utilizzano le code standard di Amazon SQS, l'ordine dei messaggi non è più garantito e i gruppi di messaggi non esistono. I product-214 e product-799 gruppi di messaggi sono indipendenti l'uno dall'altro, quindi non esiste alcuna relazione tra il modo in cui i messaggi vengono sequenziati.



Distribuzione dei dati in base agli ID dei gruppi di messaggi per migliorare le prestazioni

Per ottimizzare la velocità di trasmissione effettiva di consegna, gli argomenti FIFO di Amazon SNS recapitano i messaggi provenienti da diversi gruppi di messaggi in parallelo, mentre l'ordine dei messaggi viene mantenuto rigorosamente all'interno di ciascun gruppo di messaggi. Ogni singolo gruppo di messaggi può recapitare un massimo di 300 messaggi al secondo. Pertanto, per ottenere una velocità di trasmissione effettiva elevata per un singolo argomento, utilizza un gran numero di ID di gruppi di messaggi distinti. Utilizzando un set diversificato di gruppi di messaggi, gli

argomenti FIFO di Amazon SNS distribuiscono automaticamente i messaggi su un numero maggiore di partizioni parallele.

> **Note**
>
> Gli argomenti FIFO di Amazon SNS sono ottimizzati per la distribuzione uniforme dei messaggi tra gli ID dei gruppi di messaggi, indipendentemente dal numero di gruppi. AWS consiglia di utilizzare un gran numero di ID di gruppi di messaggi distinti per ottimizzare le prestazioni.

Quando pubblichi su Amazon SNS un argomento FIFO con velocità di trasmissione effettiva elevata e hai sottoscritto una o più code FIFO di Amazon SQS, ti consigliamo di abilitare una velocità di trasmissione effettiva sulle code. Per ulteriori informazioni, consulta High throughput for FIFO queues nella Guida per gli sviluppatori di Amazon Simple Queue Service.

Consegna dei messaggi per gli argomenti FIFO

Gli argomenti FIFO (first in, first out) di Amazon SNS supportano la distribuzione alle code standard e FIFO di Amazon SQS per offrire ai clienti la flessibilità e il controllo durante l'integrazione di applicazioni distribuite che richiedono la coerenza dei dati in tempo quasi reale.

Per i carichi di lavoro che devono mantenere un ordinamento dei messaggi rigoroso o la deduplicazione, la combinazione degli argomenti FIFO di Amazon SNS con le code FIFO di Amazon SQS sottoscritte come l'endpoint di distribuzione offre il miglioramento della messaggistica tra le applicazioni quando l'ordine delle operazioni e degli eventi è fondamentale o quando i duplicati non possono essere tollerati.

Per i carichi di lavoro che tollerano Miglior ordine possibile e Consegna almeno una volta, la sottoscrizione delle code standard di Amazon SQS agli argomenti FIFO di Amazon SNS offre la possibilità di ridurre i costi, oltre a condividere le code tra carichi di lavoro che non utilizzano FIFO.

> **Note**
>
> Per visualizzare i messaggi dagli argomenti FIFO di Amazon SNS a AWS Lambda funzioni, sono necessarie fasi aggiuntive. Per prima cosa, effettua la sottoscrizione delle code FIFO o standard di Amazon SQS all'argomento. Quindi configurare le code per attivare le funzioni. Per ulteriori informazioni, vedere SQS FIFO come origine evento nel AWS Blog Compute.

Gli argomenti FIFO SNS non possono recapitare messaggi agli endpoint gestiti dal cliente, ad esempio indirizzi e-mail, app per dispositivi mobili, numeri di telefono per SMS o endpoint HTTP (S). Questi tipi di endpoint non sono garantiti per preservare l'ordinamento rigoroso dei messaggi. I tentativi di sottoscrivere gli endpoint gestiti dal cliente agli argomenti FIFO SNS provocano errori.

Gli argomenti FIFO SNS supportano le stesse funzionalità di filtraggio dei messaggi degli argomenti standard. Per ulteriori informazioni, consulta Filtro dei messaggi per argomenti FIFO e il Semplificazione dei messaggi pub/secondari con il filtro messaggi Amazon SNS post sul AWSCompute blog.

Filtro dei messaggi per argomenti FIFO

Gli argomenti FIFO di Amazon SNS supportano il filtro dei messaggi. L'utilizzo del filtro dei messaggi semplifica l'architettura scaricando la logica di instradamento dei messaggi dai sistemi di pubblicazione e la logica di filtro dei messaggi dai sistemi di sottoscrizione.

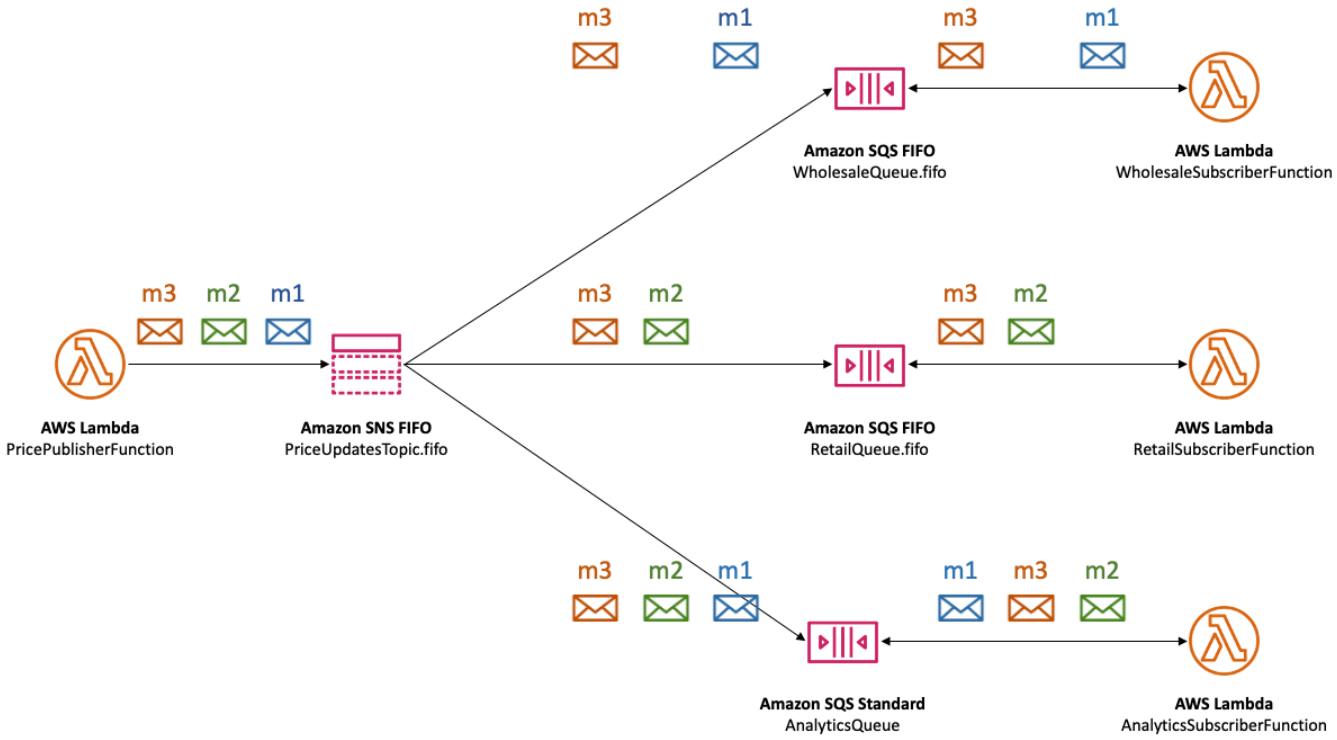
Quando effettui la sottoscrizione di una coda FIFO o standard di Amazon SQS a un argomento FIFO di SNS, puoi utilizzare il filtro dei messaggi per specificare che l'abbonato riceve un sottoinsieme di messaggi, anziché tutti. Ogni sottoscrittore può impostare la propria policy di filtro come attributi della sottoscrizione. In base al suo ambito, la policy di filtro viene confrontata con gli attributi o con il corpo dei messaggi in entrata. Se viene rilevata la corrispondenza con la policy di filtro, l'argomento invia una copia del messaggio al server del sottoscrittore. Se non c'è corrispondenza, l'argomento non recapita una copia del messaggio.

Nel caso d'uso esempio di gestione dei prezzi delle parti auto, si supponga che siano impostate le seguenti policy di filtro Amazon SNS e che l'ambito della policy sia `MessageBody`:

- Per la coda relativa al commercio all'ingrosso, la policy di filtro `{"business":["wholesale"]}` corrisponde a ogni messaggio contenente una chiave denominata `business` e con `wholesale` nel set di valori. Nel diagramma seguente, una delle chiavi nel messaggio m1 è `business` e ha un valore di `wholesale`. Una delle chiavi nel messaggio m3 è `business` e ha un valore di `["wholesale,retail"]`. Così, entrambi m1 e m3 corrispondono ai criteri della policy di filtro ed entrambi i messaggi vengono recapitati alla coda all'ingrosso.
- Per la coda relativa al commercio al dettaglio, la policy di filtro `{"business":["retail"]}` corrisponde a ogni messaggio contenente una chiave denominata `business` e `retail` nel set di valori. Nel diagramma, una delle chiavi nel messaggio m2 è `business` e ha un valore di `retail`. Una delle chiavi del messaggio m3 è `business` e ha un valore di `["wholesale,retail"]`.

Così, entrambi m2 e m3 corrispondono ai criteri della policy di filtro ed entrambi i messaggi vengono recapitati alla coda di vendita al dettaglio.

- Per la coda di analisi, Amazon Athena deve poter ricevere tutti i record, quindi non viene applicata alcuna policy di filtro.



Gli argomenti FIFO SNS supportano una varietà di operatori corrispondenti, inclusi i valori delle stringhe degli attributi, i valori numerici degli attributi e le chiavi degli attributi. Per ulteriori informazioni, consulta Filtraggio messaggi di Amazon SNS.

Gli argomenti FIFO SNS non recapitano messaggi duplicati agli endpoint sottoscritti. Per ulteriori informazioni, consulta Deduplicazione dei messaggi per argomenti FIFO.

Deduplicazione dei messaggi per argomenti FIFO

Gli argomenti FIFO di Amazon SNS e le code FIFO di Amazon SQS supportano la deduplicazione dei messaggi, che fornisce il recapito e l'elaborazione dei messaggi esattamente una volta, purché siano soddisfatte le seguenti condizioni:

- La coda FIFO di Amazon SQS sottoscritta esiste e dispone di autorizzazioni che consentono al principale del servizio Amazon SNS di distribuire i messaggi alla coda.
- Il consumer della coda FIFO di Amazon SQS elabora il messaggio e lo elimina dalla coda prima della scadenza del timeout di visibilità.
- L'argomento relativo all'abbonamento Amazon SNS non ha Filtro dei messaggi. Quando configuri il filtraggio dei messaggi, gli argomenti FIFO di Amazon SNS at-most-once supportano la consegna, poiché i messaggi possono essere filtrati in base alle politiche di filtro dell'abbonamento.
- Non ci sono interruzioni di rete che impediscono il riconoscimento del recapito dei messaggi.

> **Note**
>
> La deduplicazione dei messaggi si applica a un intero argomento FIFO di Amazon SNS, non a un singolo Gruppo di messaggi.

Quando si pubblica un messaggio in un argomento FIFO di Amazon SNS, il messaggio deve includere un ID di deduplicazione. Questo ID è incluso nel messaggio che l'argomento SNS di Amazon FIFO recapita alle code FIFO di Amazon SQS sottoscritte.

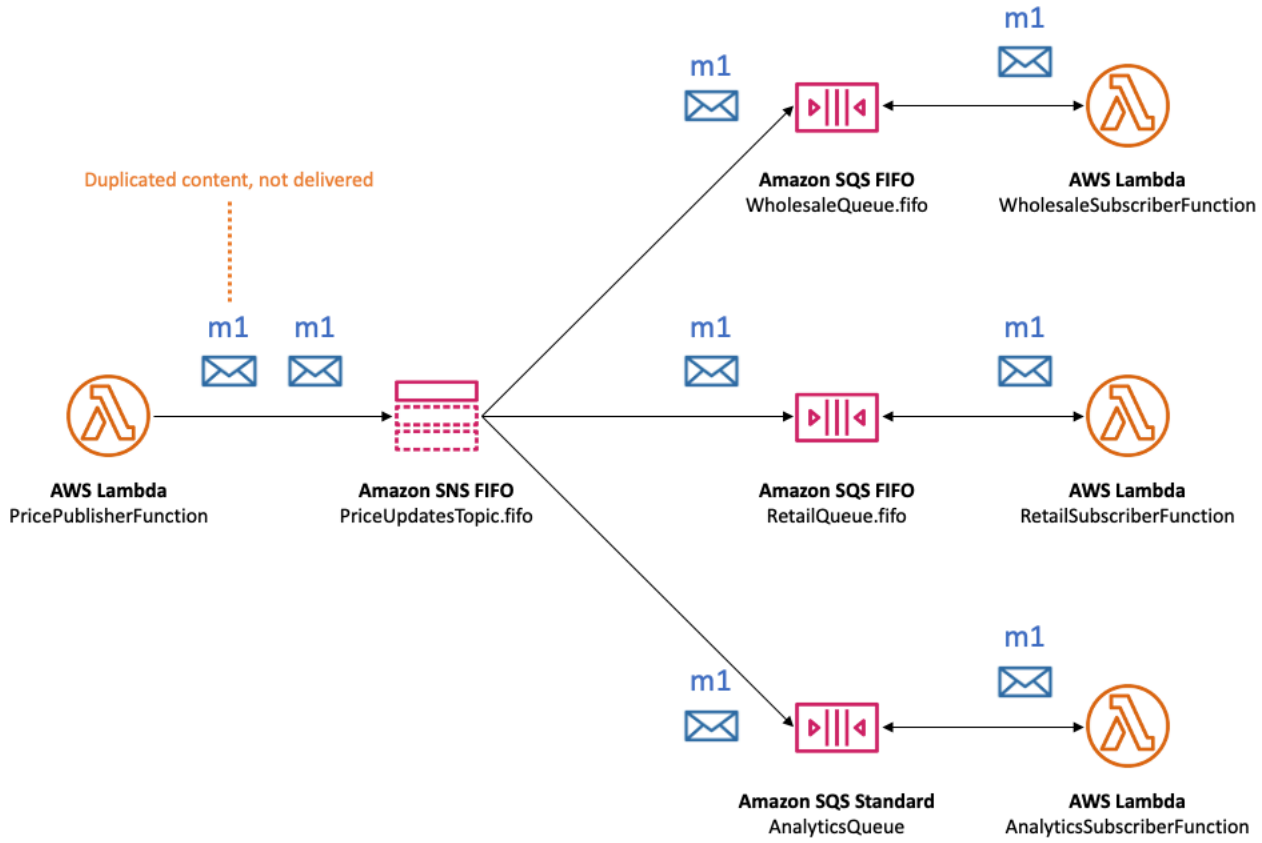
Se un messaggio con un particolare ID di deduplicazione viene pubblicato correttamente in un argomento FIFO di Amazon SNS, qualsiasi messaggio pubblicato con lo stesso ID di deduplicazione, entro l'intervallo di deduplicazione di cinque minuti, viene accettato ma non recapitato. L'argomento FIFO di Amazon SNS FIFO continua a tenere traccia dell'ID di deduplicazione dei messaggi, anche dopo che il messaggio viene recapitato agli endpoint sottoscritti.

Se è garantito che il corpo del messaggio sia univoco per ogni messaggio pubblicato, puoi abilitare la deduplicazione basata sul contenuto per un argomento FIFO di Amazon SNS e per le code FIFO di Amazon SQS sottoscritte. Amazon SNS utilizza il corpo del messaggio per generare un valore hash univoco da utilizzare come ID di deduplicazione per ogni messaggio, pertanto non è necessario impostare un ID di deduplicazione quando si invia ogni messaggio.

> **Note**
>
> Gli attributi del messaggio non sono inclusi nel calcolo hash.

Quando la deduplicazione basata sui contenuti è abilitata per un argomento FIFO di Amazon SNS e viene pubblicato un messaggio con un ID di deduplicazione, l'ID di deduplicazione pubblicato sostituisce l'ID di deduplicazione basato sul contenuto generato.

Nella caso d'uso esempio di gestione dei prezzi delle parti auto, l'azienda deve impostare un ID di deduplicazione universalmente univoco per ogni aggiornamento del prezzo. Questo perché il corpo del messaggio può essere identico anche quando l'attributo del messaggio è diverso per l'ingrosso e la vendita al dettaglio. Tuttavia, se l'azienda ha aggiunto il tipo di attività (all'ingrosso o al dettaglio) al corpo del messaggio insieme all'ID del prodotto e al prezzo del prodotto, potrebbe abilitare la duplicazione basata sul contenuto nell'argomento FIFO di Amazon SNS e nelle code FIFO di Amazon SQS sottoscritte.



Oltre all'ordinamento e alla deduplicazione dei messaggi, gli argomenti FIFO di Amazon SNS supportano la crittografia lato server dei messaggi (SSE) con chiavi e la privacy dei messaggi tramite endpoint VPC AWS KMS con. AWS PrivateLink Per ulteriori informazioni, consulta Sicurezza dei messaggi per gli argomenti FIFO.

Sicurezza dei messaggi per gli argomenti FIFO

Puoi scegliere di fare in modo che Amazon SNS e Amazon SQS crittografino i messaggi inviati agli argomenti e alle code FIFO, utilizzando AWS Key Management Service (AWS KMS) chiavi master del cliente (CMK). È possibile creare argomenti e code FIFO crittografati oppure scegliere di crittografare gli argomenti e le code FIFO esistenti. Amazon SNS e Amazon SQS crittografano solo il corpo del messaggio. Non crittografano gli attributi del messaggio, i metadati delle risorse o le metriche delle risorse.

> **Note**
>
> L'aggiunta di crittografia a un argomento o coda FIFO esistente non crittografa i messaggi arretrati e la rimozione della crittografia da un argomento o da una coda lascia crittografati i messaggi in backlog.

Gli argomenti SNS FIFO decifrano i messaggi immediatamente prima di consegnarli agli endpoint sottoscritti. SQS FIFO code decrittografare il messaggio appena prima di restituirli all'applicazione consumer. Per ulteriori informazioni, consulta Crittografia dei dati e la Crittografia dei messaggi pubblicati su Amazon SNS con AWS KMS post sul AWS Blog Compute.

Inoltre, gli argomenti SNS FIFO e le code FIFO SQS supportano la privacy dei messaggi con Endpoint VPC dell'interfaccia powered by AWS PrivateLink. Utilizzando gli endpoint dell'interfaccia, puoi inviare messaggi dalle subnet Amazon Virtual Private Cloud (Amazon VPC) agli argomenti e alle code FIFO senza attraversare Internet pubblico. Questo modello mantiene la messaggistica all'interno dell'infrastruttura AWS, che migliora la sicurezza complessiva dell'applicazione. Quando si utilizza AWS PrivateLink non è necessario configurare un gateway Internet, una NAT (Network Address Translation) o una rete privata virtuale (VPN). Per ulteriori informazioni, consulta Riservatezza del traffico Internet e il Proteggere i messaggi pubblicati su Amazon SNS con AWS PrivateLink post sul AWS Blog di sulla sicurezza.

Gli argomenti FIFO SNS supportano anche le code non recapitate e l'archiviazione dei messaggi nelle zone di disponibilità. Per ulteriori informazioni, consulta Durabilità messaggi per gli argomenti FIFO.

Durabilità messaggi per gli argomenti FIFO

Gli argomenti FIFO di Amazon SNS e le code di Amazon SQS sono durevoli. Entrambi i tipi di risorse memorizzano i messaggi in modo ridondante in più zone di disponibilità e forniscono code non recapitate per gestire casi eccezionali.

In Amazon SNS, la consegna dei messaggi non riesce quando l'argomento Amazon SNS non può accedere a una coda Amazon SQS sottoscritta a causa di un errore lato client o lato server:

- Gli errori lato client si verificano quando l'argomento FIFO di Amazon SNS dispone di metadati di sottoscrizione obsoleti. Due cause comuni di errori lato client sono quando il proprietario della coda FIFO di Amazon SQS esegue una delle seguenti operazioni:
  - Elimina la coda.
  - Modifica la policy della coda in modo da impedire all'entità servizio Amazon SNS di recapitare i messaggi.

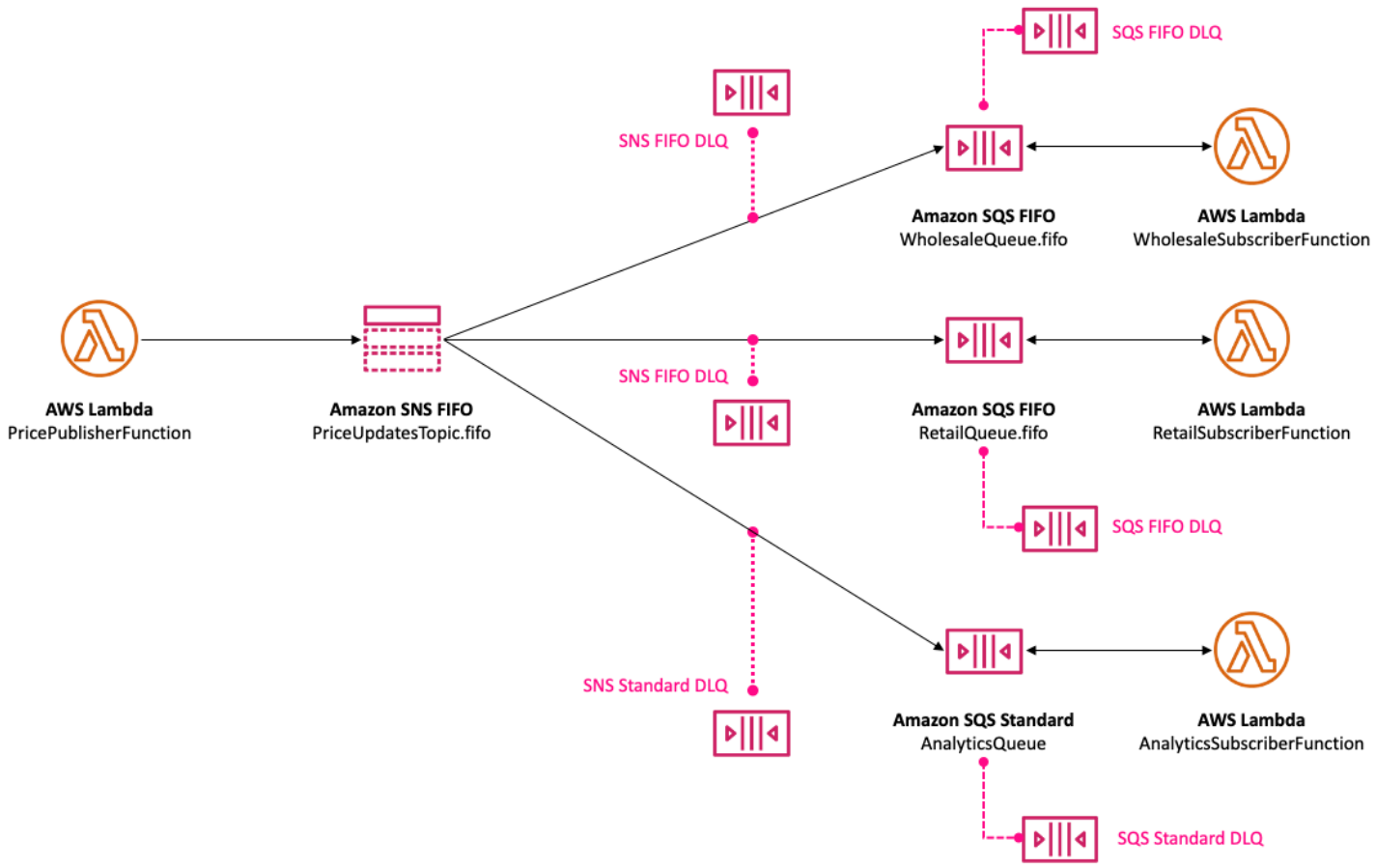
  Amazon SNS non riprova a recapitare messaggi non riusciti a causa di errori sul lato client.
- Errori sul lato server possono verificarsi nelle seguenti situazioni:
  - Il servizio Amazon SQS non è disponibile.
  - Amazon SQS non riesce a elaborare una richiesta valida dal servizio Amazon SNS.

  Quando si verificano errori lato server, gli argomenti FIFO di Amazon SNS riproveranno a eseguire le consegne non riuscite fino a 100.015 volte in 23 giorni. Per ulteriori informazioni, consulta Tentativi di consegna dei messaggi di Amazon SNS.

Per qualsiasi tipo di errore, Amazon SNS può mettere da parte i messaggi alle code con lettere non recapitate di Amazon SQS in modo che i dati non vengano persi.

In Amazon SQS, l'elaborazione dei messaggi ha esito negativo quando l'applicazione consumer non riceve il messaggio, lo elabora ed elimina dalla coda. Quando il numero massimo di richieste di ricezione non riesce, Amazon SQS può mettere da parte i messaggi alle code non recapitate in modo che i dati non vengano persi.

Nel caso d'uso di esempio di gestione dei prezzi delle parti auto, l'azienda può assegnare una coda DLQ di Amazon SQS a ogni sottoscrizione di argomento FIFO di Amazon SNS, nonché a ogni coda Amazon SQS sottoscritta. Questo protegge l'azienda da qualsiasi perdita di aggiornamento dei prezzi.



La coda DLQ associata a una sottoscrizione Amazon SNS deve essere una coda di Amazon SQS dello stesso tipo della coda di sottoscrizione. Ad esempio, la sottoscrizione FIFO di Amazon SNS per una coda FIFO di Amazon SQS deve avere una coda FIFO di Amazon SQS come coda DLQ. Analogamente, la sottoscrizione FIFO di Amazon SNS per una coda FIFO di Amazon SQS deve avere una coda FIFO di Amazon SQS come coda DLQ. Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS e la Progettazione di applicazioni serverless durevoli con DLQ per Amazon SNS, Amazon SQS,AWS Lambda post sul AWSBlog Compute.

Per una maggiore durabilità e facilitare il ripristino dagli errori downstream, i proprietari degli argomenti possono anche utilizzare gli argomenti FIFO per archiviare i messaggi fino a 365 giorni. Gli abbonati agli argomenti possono riprodurre i messaggi archiviati su un endpoint sottoscritto per recuperare i messaggi causati da un errore in un'applicazione downstream o per replicare lo stato di un'applicazione esistente. Per ulteriori informazioni, consulta Archiviazione e riproduzione dei messaggi per argomenti FIFO.

Archiviazione e riproduzione dei messaggi per argomenti FIFO

Argomenti

Che cosa è l'archiviazione e riproduzione dei messaggi?

L'archiviazione e riproduzione dei messaggi Amazon SNS è un archivio di messaggi locale senza codice che consente ai proprietari degli argomenti di archiviare i messaggi all'interno del proprio argomento. Gli abbonati agli argomenti possono quindi recuperare (o riprodurre) i messaggi archiviati su un endpoint sottoscritto che può essere usato per:

- Recuperare i messaggi che potrebbero essere andati persi a causa di un errore in un'applicazione downstream.
- Replicare lo stato di un'applicazione esistente su una nuova applicazione sottoscrivendo il nuovo endpoint e selezionando il timestamp desiderato da cui eseguire la replica.

Puoi utilizzare l'archiviazione e riproduzione dei messaggi con l'API AWS, l'SDK, AWS CloudFormation e la AWS Management Console.

> **Note**
>
> L'archiviazione e riproduzione dei messaggi di Amazon SNS è disponibile solo per gli argomenti FIFO da applicazione a applicazione (A2A).

L'archiviazione e riproduzione dei messaggi è costituita da due componenti principali:

1. Archiviazione dei messaggi: il proprietario dell'argomento attiva la funzionalità di archiviazione e riproduzione su un argomento e imposta un periodo di conservazione dei messaggi (fino a 365 giorni). Il proprietario dell'argomento può anche monitorare i messaggi archiviati utilizzando le metriche Amazon CloudWatch. Per ulteriori informazioni, consulta Archiviazione dei messaggi per i proprietari di argomenti FIFO.

2. Riproduzione dei messaggi: l'abbonato dell'argomento avvia una riproduzione di una serie di messaggi dall'argomento all'endpoint sottoscritto. Per ulteriori informazioni, consulta Riproduzione dei messaggi per gli abbonati all'argomento FIFO.

Archiviazione dei messaggi per i proprietari di argomenti FIFO

L'archiviazione dei messaggi offre la possibilità di archiviare una singola copia di tutti i messaggi pubblicati sull'argomento. Puoi archiviare i messaggi pubblicati all'interno del tuo argomento abilitando la policy di archiviazione dei messaggi sull'argomento, che consente l'archiviazione dei messaggi per tutte le sottoscrizioni collegate a quell'argomento. I messaggi possono essere archiviati da un minimo di un giorno a un massimo di 365 giorni.

Quando si imposta una policy di archiviazione, si applicano costi aggiuntivi. Per informazioni sui prezzi, consulta Prezzi di Amazon SNS.

Argomenti

- Creazione di una policy di archiviazione dei messaggi utilizzando la AWS Management Console
- Creazione di una policy di archiviazione dei messaggi utilizzando l'API
- Creazione di una policy di archiviazione dei messaggi utilizzando l'SDK
- Creazione di una policy di archiviazione dei messaggi utilizzando AWS CloudFormation
- Concessione dell'accesso a un archivio crittografato
- Monitoraggio delle metriche di archivio dei messaggi utilizzando Amazon CloudWatch

Creazione di una policy di archiviazione dei messaggi utilizzando la AWS Management Console

Utilizza questa opzione per creare una nuova policy di archiviazione dei messaggi utilizzando la AWS Management Console.

1. Accedi alla console Amazon SNS.
2. Scegli un argomento o creane uno nuovo. Per ulteriori informazioni sulla creazione di argomenti, consulta Creare un argomento Amazon SNS.

> **Note**
>
> L'archiviazione e riproduzione dei messaggi di Amazon SNS è disponibile solo per gli argomenti FIFO da applicazione a applicazione (A2A).

3. Nella pagina Modifica argomento espandi la sezione Policy di archiviazione.
4. Abilita la funzionalità Policy di archiviazione e inserisci il Numero di giorni per i quali desideri archiviare i messaggi nell'argomento.
5. Seleziona Salva modifiche.

Per visualizzare, modificare e disattivare una policy relativa all'argomento di archiviazione dei messaggi

- Nella pagina Dettagli dell'argomento, Policy di conservazione mostra lo stato della policy di archiviazione, incluso il numero di giorni per i quali è stata impostata. Seleziona la scheda Policy di conservazione per visualizzare i seguenti dettagli sull'archivio dei messaggi:
  - Stato: lo stato di archiviazione e riproduzione appare attivo quando viene applicata una policy di archiviazione. Lo stato di archiviazione e riproduzione appare come inattivo quando la policy di archiviazione è impostata su un oggetto JSON vuoto.
  - Periodo di conservazione dei messaggi: il numero di giorni specificato per la conservazione dei messaggi.
  - Data di inizio dell'archiviazione: la data a partire dalla quale gli abbonati possono riprodurre i messaggi.
  - Anteprima JSON: l'anteprima JSON della policy di archiviazione.
- (Facoltativo) Per modificare una policy di archiviazione, vai alla pagina di riepilogo dell'argomento e scegli Modifica.
- (Facoltativo) Per disattivare una policy di archiviazione, vai alla pagina di riepilogo dell'argomento e scegli Modifica. Disattiva la policy di archiviazione e scegli Salva modifiche.
- (Facoltativo) Per eliminare un argomento con una policy di archiviazione, è necessario prima disattivare la policy di archiviazione come descritto in precedenza.

> **Important**
>
> Per evitare eliminazioni accidentali dei messaggi, non puoi eliminare un argomento con una policy di archiviazione dei messaggi attiva. La policy di archiviazione dei messaggi

dell'argomento deve essere disattivata prima che l'argomento possa essere eliminato. Quando disattivi una policy di archiviazione dei messaggi, Amazon SNS elimina tutti i messaggi archiviati. Quando si elimina un argomento, le sottoscrizioni vengono rimosse e i messaggi in transito potrebbero non essere recapitati.

Creazione di una policy di archiviazione dei messaggi utilizzando l'API

Per creare una policy di archiviazione dei messaggi utilizzando l'API, devi aggiungere l'attributo `ArchivePolicy` al tuo argomento. Puoi impostare un `ArchivePolicy` utilizzando le operazioni API `CreateTopic` e`SetTopicAttributes`. `ArchivePolicy` ha un unico valore, `MessageRetentionPeriod`, che rappresenta il numero di giorni in cui Amazon SNS conserva i messaggi. Per attivare l'archiviazione dei messaggi per il tuo argomento, imposta `MessageRetentionPeriod` su un valore intero maggiore di zero. Ad esempio, per conservare i messaggi nell'archivio per 30 giorni, imposta `ArchivePolicy` su:

```
{
    "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
    }
}
```

Per disabilitare l'archiviazione dei messaggi relativi al tuo argomento e cancellare l'archivio, annulla l'impostazione di `ArchivePolicy`, come segue:

```
{}
```

Creazione di una policy di archiviazione dei messaggi utilizzando l'SDK

Per utilizzare un SDK AWS, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File condivisi `config` e `credentials` files nella AWS SDKs and Tools Reference Guide.

I seguenti esempi di codice mostrano come impostare la `ArchivePolicy` do un argomento Amazon SNS in modo che mantenga tutti i messaggi pubblicati sull'argomento per 30 giorni.

```
// Specify the ARN of the Amazon SNS topic to set the ArchivePolicy for.
String topicArn =
    "arn:aws:sns:us-east-2:123456789012:MyArchiveTopic.fifo";
```

```
// Set the MessageRetentionPeriod to 30 days for the ArchivePolicy.
String archivePolicy =
    "{\"MessageRetentionPeriod\":\"30\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetTopicAttributesRequest request = new SetTopicAttributesRequest()
    .withTopicArn(topicArn)
    .withAttributeName("ArchivePolicy")
    .withAttributeValue(archivePolicy);
sns.setTopicAttributes(request);
```

Creazione di una policy di archiviazione dei messaggi utilizzando AWS CloudFormation

Per creare una policy di archiviazione mediante AWS CloudFormation consulta `AWS::SNS::Topic` nella Guida per l'utente di AWS CloudFormation.

Concessione dell'accesso a un archivio crittografato

Prima che un abbonato possa iniziare a riprodurre i messaggi di un argomento crittografato, devi completare al procedura seguente. Poiché i messaggi precedenti vengono riprodotti, è necessario fornire ad Amazon SNS l'accesso `Decrypt` alla chiave KMS utilizzata per crittografare i messaggi nell'archivio.

1. Quando esegui la crittografia dei messaggi con una chiave KMS e li archivi all'interno dell'argomento, devi concedere ad Amazon SNS la possibilità di de-crittografare questi messaggi tramite la policy della chiave. Per ulteriori informazioni, consulta Concessione delle autorizzazioni di decrittografia ad Amazon SNS.
2. Abilitazione di AWS KMS per Amazon SNS. Per ulteriori informazioni, consulta Configurazione delle autorizzazioni per AWS KMS.

> **Important**
>
> Quando aggiungi le nuove sezioni alla policy della chiave KMS, non modificare le sezioni esistenti nella policy. Se la crittografia è abilitata in un argomento e la chiave KMS è disabilitata o eliminata oppure la policy della chiave KMS non è configurata correttamente per Amazon SNS, Amazon SNS non potrà riprodurre i messaggi ai tuoi abbonati.

Concessione delle autorizzazioni di decrittografia ad Amazon SNS

Affinché Amazon SNS possa accedere ai messaggi crittografati dall'archivio del tuo argomento e riprodurli sugli endpoint sottoscritti, devi abilitare il servizio Amazon SNS per decrittografare questi messaggi.

Di seguito è riportato un esempio di policy necessaria per consentire al principale del servizio Amazon SNS di de-crittografare i messaggi archiviati durante la riproduzione di messaggi cronologici dall'argomento in questione.

```
{
    "Sid": "Allow SNS to decrypt archived messages",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

Monitoraggio delle metriche di archivio dei messaggi utilizzando Amazon CloudWatch

Puoi monitorare i messaggi archiviati utilizzando Amazon CloudWatch utilizzando le seguenti metriche. Per ricevere notifiche sulle anomalie nei carichi di lavoro e contribuire a evitare impatti, puoi configurare gli allarmi di Amazon CloudWatch in base a queste metriche. Per ulteriori dettagli, consulta Registrazione e monitoraggio in Amazon SNS.

Metrica	Descrizione
ApproximateNumberOfMessagesArchived	Fornisce al proprietario dell'argomento il numero aggregato di messaggi archiviati nell'archivio degli argomenti, con una risoluzio ne di 60 minuti.
ApproximateNumberOfBytesArchived	Fornisce al proprietario dell'argomento il numero aggregato di byte archiviati in tutti i

Metrica	Descrizione
	messaggi dell'archivio degli argomenti, con una risoluzione di 60 minuti.
NumberOfMessagesArchiveProcessing	Fornisce al proprietario dell'argomento il numero di messaggi salvati nell'archivio degli argomenti durante l'intervallo con una risoluzio ne di 1 minuto.
NumberOfBytesArchiveProcessing	Fornisce al proprietario dell'argomento il numero aggregato di messaggi salvati nell'arch ivio degli argomenti durante l'intervallo con una risoluzione di 1 minuto.

L'API `GetTopicAttributes` ha una proprietà `BeginningArchiveTime` che rappresenta il timestamp più vecchio in base al quale un abbonato può avviare una riproduzione. Di seguito è riportato un esempio di risposta per questa operazione API:

```
{
 "ArchivePolicy": {
    "MessageRetentionPeriod": "<integer>"
  },
  "BeginningArchiveTime": "<timestamp>",
  ...
}
```

Riproduzione dei messaggi per gli abbonati all'argomento FIFO

La riproduzione di Amazon SNS consente agli abbonati agli argomenti di recuperare i messaggi archiviati dall'archivio dati degli argomenti e di ridistribuirli (o riprodurli) a un endpoint sottoscritto. I messaggi possono essere riprodotti non appena viene creato l'abbonamento. Un messaggio riprodotto ha lo stesso contenuto, `MessageId` e `Timestamp` della copia originale e contiene anche l'attributo `Replayed`, che consente di identificare che si tratta di un messaggio riprodotto. Per riprodurre solo messaggi selezionati, puoi aggiungere una policy di filtro al tuo abbonamento. Per ulteriori informazioni sul filtraggio dei messaggi, consulta Filtro dei messaggi riprodotti.

Argomenti

- Creazione di una policy di riproduzione dei messaggi utilizzando la AWS Management Console
- Aggiunta di una policy di riproduzione all'abbonamento utilizzando l'API
- Aggiunta di una policy di riproduzione all'abbonamento utilizzando l'SDK
- Filtro dei messaggi riprodotti
- Monitoraggio delle metriche di riproduzione utilizzando Amazon CloudWatch

Creazione di una policy di riproduzione dei messaggi utilizzando la AWS Management Console

Utilizza questa opzione per creare una nuova policy di riproduzione utilizzando la AWS Management Console.

1. Accedi alla console Amazon SNS.
2. Scegli una sottoscrizione all'argomento o creane una nuova. Per ulteriori informazioni sulla creazione di sottoscrizioni, consultaIscrizione a un argomento Amazon SNS.
3. Per avviare la riproduzione del messaggio, vai al menu a discesa Riproduci e scegli Inizia riproduzione.
4. Dalla finestra modale Intervallo di tempo di riproduzione, effettua le seguenti selezioni:
    a. Scegli data e ora di inizio della riproduzione: scegli la data (formato AAAA/MM/GG) e l'ora (formato hh:mm:ss di 24 ore) da cui iniziare a riprodurre i messaggi archiviati. L'ora di inizio deve essere successiva all'inizio dell'orario di archiviazione approssimativo.
    b. (Facoltativo) Scegli data e ora di fine della riproduzione: scegli la data (formato AAAA/MM/GG) e l'ora (formato hh:mm:ss di 24 ore) in cui terminare la riproduzione dei messaggi archiviati.
    c. Scegli Avvia riproduzione.
5. (Facoltativo) Per arrestare la riproduzione di un messaggio, vai alla pagina Dettagli sottoscrizione e scegli Interrompi riproduzione dal menu a discesa Riproduci.
6. (Facoltativo) Per monitorare le metriche di riproduzione dei messaggi dall'interno di questo flusso di lavoro utilizzando CloudWatch, consulta Monitoraggio delle metriche di riproduzione utilizzando Amazon CloudWatch.

Visualizzazione e modifica di una policy di riproduzione dei messaggi

Nella pagina Dettagli sottoscrizione puoi eseguire le seguenti operazioni:

- Per visualizzare lo stato di riproduzione dei messaggi, nel campo Stato riproduzione vengono visualizzati i seguenti valori:
  - Completato: la riproduzione ha correttamente ridistribuito tutti i messaggi e ora distribuisce i messaggi appena pubblicati.
  - In corso: la riproduzione sta attualmente riproducendo i messaggi selezionati.
  - Non riuscito: la riproduzione non è stata completata.
  - In sospeso: lo stato predefinito durante l'avvio della riproduzione.
- (Facoltativo) Per modificare la policy di riproduzione dei messaggi, vai alla pagina Dettagli sottoscrizione e scegli Interrompi riproduzione dal menu a discesa Riproduci. L'avvio di una riproduzione sostituirà la riproduzione esistente.

Aggiunta di una policy di riproduzione all'abbonamento utilizzando l'API

Per riprodurre i messaggi archiviati, usa l'attributo `ReplayPolicy`. `ReplayPolicy` può essere utilizzato con le operazioni API `Subscribe` e `SetSubscriptionAttributes`. Questa policy ha i seguenti valori:

- `StartingPoint` (Obbligatorio): segnala da dove iniziare a riprodurre i messaggi.
- `EndingPoint` (Facoltativo): segnala quando interrompere la riproduzione dei messaggi. Se `EndingPoint` viene omesso, la riproduzione continuerà fino a quando non verrà raggiunta l'ora corrente.
- `PointType` (Obbligatorio): imposta il tipo di punto iniziale e finale. Attualmente, l'unico valore supportato per `PointType` è `Timestamp`.

Ad esempio, per ripristinare un errore downstream e inviare nuovamente tutti i messaggi per un periodo di due ore il 1° ottobre 2023, utilizza l'operazione API `SetSubscriptionAttributes` per impostare una `ReplayPolicy` come segue:

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T10:00:00.000Z",
  "EndingPoint":"2023-10-01T12:00:00.000Z"
}
```

Per riprodurre tutti i messaggi inviati all'argomento a partire dal 1° ottobre 2023 e continuare a ricevere tutti i messaggi appena pubblicati sull'argomento, utilizza l'operazione API

`SetSubscriptionAttributes` per impostare una `ReplayPolicy` sull'abbonamento nel modo seguente:

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T00:00:00.000Z"
}
```

Per verificare che un messaggio sia stato riprodotto, l'attributo booleano `Replayed` viene aggiunto a ogni messaggio riprodotto.

Aggiunta di una policy di riproduzione all'abbonamento utilizzando l'SDK

Per utilizzare un SDK AWS, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File condivisi `config` e `credentials` files nella AWS SDKs and Tools Reference Guide.

Il seguente esempio di codice mostra come impostare la `ReplayPolicy` su un abbonamento per recapitare nuovamente i messaggi dall'archivio degli argomenti FIFO di Amazon SNS per una finestra temporale di 2 ore il 1° ottobre 2023.

```
// Specify the ARN of the Amazon SNS subscription to initiate the ReplayPolicy on.
String subscriptionArn =
    "arn:aws:sns:us-
east-2:123456789012:MyArchiveTopic.fifo:1d2a3e9d-7f2f-447c-88ae-03f1c68294da";

// Set the ReplayPolicy to replay messages from the topic's archive
// for a 2 hour time period on October 1st 2023 between 10am and 12pm UTC.
String replayPolicy =
    "{\"PointType\":\"Timestamp\",\"StartingPoint\":\"2023-10-01T10:00:00.000Z\",
\"EndingPoint\":\"2023-10-01T12:00:00.000Z\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("ReplayPolicy")
    .withAttributeValue(replayPolicy);
sns.setSubscriptionAttributes(request);
```

Filtro dei messaggi riprodotti

Il filtraggio dei messaggi di Amazon SNS ti consente di controllare i messaggi riprodotti che Amazon SNS riproduce sull'endpoint dell'abbonato. Quando il filtraggio e l'archiviazione dei messaggi sono

entrambi abilitati, Amazon SNS recupera prima il messaggio dall'archivio dati dell'argomento, quindi applica il messaggio alla `FilterPolicy` dell'abbonamento. Il messaggio viene recapitato all'endpoint sottoscritto quando c'è una corrispondenza, altrimenti il messaggio viene filtrato. Per ulteriori informazioni, consulta Policy di filtro degli abbonamenti Amazon SNS.

Monitoraggio delle metriche di riproduzione utilizzando Amazon CloudWatch

Puoi monitorare i messaggi riprodotti mediante Amazon CloudWatch utilizzando le seguenti metriche. Per ricevere notifiche sulle anomalie nei carichi di lavoro e contribuire a evitare impatti, puoi configurare gli allarmi di Amazon CloudWatch in base a queste metriche. Per ulteriori dettagli, consulta Registrazione e monitoraggio in Amazon SNS.

Metrica	Descrizione
NumberOfReplayedNotificationsDelivered	Fornisce all'abbonato il numero aggregato di messaggi riprodotti dall'archivio degli argomenti , con una risoluzione di 1 minuto.
NumberOfReplayedNotificationsFailed	Fornisce all'abbonato il numero aggregato di messaggi riprodotti non inviati dall'archivio degli argomenti, con una risoluzione di 1 minuto.

Esempi di codice per argomenti FIFO

Puoi utilizzare i seguenti esempi di codice per integrare il caso d'uso di esempio di gestione dei prezzi delle parti auto utilizzando un argomento FIFO di Amazon SNS con una coda FIFO di Amazon SQS o la coda standard.

Argomenti

- Utilizzo di un SDK AWS
- Uso di AWS CloudFormation

Utilizzo di un SDK AWS

Utilizzando un AWS SDK, puoi creare un argomento FIFO di Amazon SNS impostando il relativo attributo **`true`** su `FifoTopic`. Puoi creare una coda FIFO di Amazon SQS impostando il relativo attributo `FifoQueue` su **`true`**. Inoltre, è necessario aggiungere il **`.fifo`** suffisso al nome di ogni

risorsa FIFO. Dopo aver creato un argomento o una coda FIFO, non è possibile convertirlo in un argomento o coda standard.

Il codice di esempio seguente crea queste risorse della coda FIFO e standard:

- L'argomento FIFO di Amazon SNS che distribuisce gli aggiornamenti dei prezzi
- Le code FIFO di Amazon SQS che forniscono questi aggiornamenti alle applicazioni all'ingrosso e al dettaglio
- La coda standard di Amazon SQS per l'applicazione di analisi che archivia i record, su cui è possibile eseguire query per business intelligence (BI)
- Le sottoscrizioni FIFO di Amazon SNS che connettono le tre code all'argomento

In questo esempio vengono impostate policy di filtro sulle sottoscrizioni. Se esegui il test dell'esempio pubblicando un messaggio nell'argomento, assicurati di pubblicarlo con l'attributo `business`. Specifica `retail` o `wholesale` per il valore dell'attributo. In caso contrario, il messaggio viene filtrato e non recapitato alle code sottoscritte. Per ulteriori informazioni, consulta Filtro dei messaggi per argomenti FIFO.

Java

SDK per Java 2.x

> **Note**
>
> Ulteriori informazioni su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

In questo esempio

- viene creato un argomento Amazon SNS FIFO, due code FIFO Amazon SQS e una coda Standard.
- viene effettuata la sottoscrizione all'argomento e pubblicato un messaggio nell'argomento.

Il test verifica la ricezione del messaggio in ogni coda. L'esempio completo mostra anche l'aggiunta di policy di accesso e l'eliminazione delle risorse alla fine.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
```

```
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);
```

```
        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
```

```
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
```

```
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x.
  - CreateTopic
  - Pubblicare
  - Subscribe

Python

SDK per Python (Boto3)

> **Note**
>
> Ulteriori informazioni su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea un argomento FIFO, sottoscrivi una coda FIFO di Amazon SQS all'argomento e pubblica un messaggio su un argomento Amazon SNS.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
```

```
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
```

```
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
```

```
        Topic names must be made up of only uppercase and lowercase ASCII
letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
```

```
                                "Principal": {"AWS": "*"},
                                "Action": "SQS:SendMessage",
                                "Resource": queue.attributes["QueueArn"],
                                "Condition": {
                                    "ArnLike": {"aws:SourceArn": topic_arn}
                                },
                            }
                        ],
                    }
                )
            }
        )
        logger.info("Added trust policy to the queue.")
    except ClientError as error:
        logger.exception("Couldn't add trust policy to the queue!")
        raise error


@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error


@staticmethod
def publish_price_update(topic, payload, group_id):
    """
    Compose and publish a message that updates the wholesale price.
```

```
        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Python (Boto3).
  - CreateTopic
  - Pubblicare
  - Subscribe

SAP ABAP

SDK per SAP ABAP

> **Note**
>
> Ulteriori informazioni su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Crea un argomento FIFO, sottoscrivi una coda Amazon SQS FIFO all'argomento e pubblica un messaggio su un argomento Amazon SNS.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
```

```
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                        "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes
```

```
      ).
        ov_message_id = lo_result->get_messageid( ).                    "
ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per informazioni dettagliate sull'API, consulta gli argomenti seguenti nella Documentazione di riferimento dell'API dell'SDK AWS per SAP ABAP.
  - CreateTopic
  - Pubblicare
  - Subscribe

Ricezione di messaggi da sottoscrizioni FIFO

Ora puoi ricevere aggiornamenti dei prezzi nelle tre applicazioni sottoscritte. Come mostrato nel the section called “Caso d'uso degli argomenti FIFO”, il punto di entrata per ogni applicazione consumer è la coda Amazon SQS, per la quale la funzione AWS Lambda corrispondente può eseguire il polling automatico. Quando una coda Amazon SQS è un'origine di eventi per una funzione Lambda, Lambda ridimensiona il proprio parco istanze di strumenti per il polling in base alle esigenze per consumare i messaggi in modo efficiente.

Per ulteriori informazioni, consulta Utilizzo di AWS Lambda con Amazon SQS nella AWS Lambda Guida per gli sviluppatori. Per informazioni su come scrivere i propri pollers della coda, vedere Raccomandazioni per code standard e FIFO di Amazon SQS nella Giuda per lo sviluppatore Amazon Simple Queue Service e ReceiveMessage nella Informazioni di riferimento sulle API di Amazon Simple.

Uso di AWS CloudFormation

AWS CloudFormation consente di utilizzare un file di modello per creare e configurare una raccolta di risorse AWS come una singola unità. Questa sezione include un modello di esempio in grado di creare quanto segue:

- L'argomento FIFO di Amazon SNS che distribuisce gli aggiornamenti dei prezzi

- Le code FIFO di Amazon SQS che forniscono questi aggiornamenti alle applicazioni all'ingrosso e al dettaglio
- La coda standard di Amazon SQS per l'applicazione di analisi che archivia i record, su cui è possibile eseguire query per business intelligence (BI)
- Le sottoscrizioni FIFO di Amazon SNS che connettono le tre code all'argomento
- Una Policy di filtro che specifica che le applicazioni sottoscrittori ricevono solo gli aggiornamenti di prezzo di cui hanno bisogno

> **Note**
>
> Se esegui il test di questo esempio di codice pubblicando un messaggio nell'argomento, assicurati di pubblicare il messaggio con l'attributo `business`. Specifica `retail` o `wholesale` per il valore dell'attributo. In caso contrario, il messaggio viene filtrato e non recapitato alle code sottoscritte.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "PriceUpdatesTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "TopicName": "PriceUpdatesTopic.fifo",
        "FifoTopic": true,
        "ContentBasedDeduplication": false,
        "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
        }
      }
    },
    "WholesaleQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "WholesaleQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "RetailQueue": {
```

```
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "RetailQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "AnalyticsQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "AnalyticsQueue"
      }
    },
    "WholesaleSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "WholesaleQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "wholesale"
          ]
        }
      }
    },
    "RetailSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "RetailQueue",
```

```
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "retail"
          ]
        }
      }
    },
    "AnalyticsSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "AnalyticsQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false"
      }
    },
    "SalesQueuesPolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "sns.amazonaws.com"
              },
              "Action": [
                "sqs:SendMessage"
              ],
              "Resource": "*",
              "Condition": {
```

```
                "ArnEquals": {
                  "aws:SourceArn": {
                    "Ref": "PriceUpdatesTopic"
                  }
                }
              }
            }
          ]
        },
        "Queues": [
          {
            "Ref": "WholesaleQueue"
          },
          {
            "Ref": "RetailQueue"
          },
          {
            "Ref": "AnalyticsQueue"
          }
        ]
      }
    }
  }
}
```

Per ulteriori informazioni sulla distribuzione delle risorse AWS mediante un modello AWS CloudFormation, consulta Nozioni di base nella AWS CloudFormation Guida per l'utente.

Pubblicazione messaggi di Amazon SNS

Dopo aver creato un argomento Amazon SNS e avervi iscritto un endpoint, puoi pubblicare messaggi in tale argomento. Quando viene pubblicato un messaggio, Amazon SNS tenta di recapitare il messaggio a Endpoint iscritti.

Argomenti

- Per pubblicare messaggi su argomenti Amazon SNS utilizzando il AWS Management Console
- Pubblicazione di un messaggio in un argomento tramite AWS
- Pubblicazione di messaggi di grandi dimensioni con Amazon SNS e Amazon S3
- Attributi messaggio di Amazon SNS
- Batch di messaggi Amazon SNS

Per pubblicare messaggi su argomenti Amazon SNS utilizzando il AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).
3. Nella pagina Topics (Argomenti), seleziona un argomento e scegli Publish messaggio (Pubblica messaggio).

   La console apre la pagina Pubblica messaggio in argomento.
4. Nella sezione Basic details (Dettagli di base), eseguire le seguenti:

   a. (Facoltativo) Immettere un messaggio Oggetto.

   b. Per un Argomento FIFO, immettere un ID gruppo di messaggi. I messaggi nello stesso gruppo di messaggi vengono recapitati nell'ordine in cui vengono pubblicati.

   c. Per un argomento FIFO, immettere un ID deduplicazione messaggi. Questo ID è facoltativo se è stata attivata l'impostazione Deduplicazione dei messaggi basata sul contenuto per l'argomento.

   d. (Facoltativo) Per Notife push per dispositivi, immettere un valore Time to Live (TTL) in secondi. Questo è il tempo impiegato da un servizio di notifica push, come Apple Push Notification Service (APN) o Firebase Cloud Messaging (FCM), per consegnare il messaggio all'endpoint.

5. Nella sezione Message body (Corpo del messaggio), eseguire una delle seguenti operazioni:

   a. Scegliere Identical payload for all delivery protocols (Payload identico per tutti i protocolli di consegna) e immettere il messaggio.

   b. Scegliere Custom payload for each delivery protocol (Payload personalizzato per ogni protocollo di consegna) e inserire l'oggetto JSON per definire il messaggio da inviare a ogni protocollo.

   Per ulteriori informazioni, consulta Pubblicazione con payload specifici della piattaforma.

6. Nella sezione Message attributes (Attributi messaggio), aggiungere tutti gli attributi che Amazon SNS deve confrontare con l'attributo di sottoscrizione `FilterPolicy` per decidere se l'endpoint sottoscritto è interessato al messaggio pubblicato.

   a. Per Type (Tipo), scegliere un tipo di attributo, ad esempio String.Array.

   > **Note**
   >
   > Per il tipo di attributo String.Array, racchiudi la matrice tra parentesi (`[]`). All'interno della matrice, racchiudi i valori di stringa tra virgolette. Non hai bisogno di usare le virgolette per i numeri o per le parole chiave `true`, `false` e `null`.

   b. Immettere un attributo Nome, ad esempio `customer_interests`.

   c. Immettere un attributo Valore, ad esempio `["soccer", "rugby", "hockey"]`.

   Se il tipo di attributo è String, String.Array o Number, Amazon SNS valuta l'attributo del messaggio rispetto alla policy di filtro di una sottoscrizione, se presente, prima di inviare il messaggio a tale sottoscrizione se l'ambito della policy di filtro non impostato in modo esplicito su `MessageBody`.

   Per ulteriori informazioni, consulta Attributi messaggio di Amazon SNS.

7. Seleziona Publish message (Pubblica messaggio).

   Il messaggio viene pubblicato all'argomento e la console apre la pagina dell’argomento Dettagli.

Pubblicazione di un messaggio in un argomento tramite AWS

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

I seguenti esempi di codice mostrano come utilizzare. `Publish`

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Pubblicare un messaggio in un argomento.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
```

```
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Pubblica un messaggio in un argomento con opzioni di gruppo, duplicazione e attributo.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
```

```
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
```

```
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

Applica le selezioni dell'utente all'azione di pubblicazione.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
```

```
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for .NET .

C++

SDK per C++

> **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.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
```

```
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Pubblica un messaggio con un attributo.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
```

```
        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Esempio 1: pubblicazione di un messaggio in un argomento

Nell'esempio `publish` viene pubblicato il messaggio indicato in un argomento SNS specificato. Il messaggio proviene da un file di testo che consente di includere interruzioni di riga.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

Contenuto di `message.txt`.

```
Hello World
Second Line
```

Output:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

Esempio 2: pubblicare un messaggio SMS su un numero di telefono

Nell'esempio `publish` seguente viene pubblicato il messaggio `Hello world!` sul numero di telefono`+1-555-555-0100`.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

Output:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- Per ulteriori informazioni sulle API, consulta Publish nel Riferimento ai comandi AWS CLI .

Go

SDK per Go V2

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for Go .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
```

```
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Pubblica un messaggio in un argomento con opzioni di gruppo, duplicazione e attributo.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
```

```
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for JavaScript .

Kotlin

SDK per Kotlin

> **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.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API di SDK AWS per Kotlin.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for PHP .

PowerShell

Utensili per PowerShell

Esempio 1: Questo esempio mostra la pubblicazione di un messaggio con una sola riga MessageAttribute dichiarata.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Esempio 2: Questo esempio mostra la pubblicazione di un messaggio con più messaggi MessageAttributes dichiarati in anticipo.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Per i dettagli sull'API, vedere Publish in AWS Tools for PowerShell Cmdlet Reference.

Python

SDK per Python (Boto3)

> **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.

Pubblicare un messaggio con attributi in modo che una sottoscrizione possa filtrare in base agli attributi.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
```

```
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Pubblicare un messaggio che assume forme diverse in base al protocollo del sottoscrittore.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
```

```
        :param default_message: The default version of the message. This version
is
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per l'API SDK for Python (Boto3)AWS .

Ruby

SDK per Ruby

> **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.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content
```

```
  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for Ruby .

Rust

SDK per Rust

> **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.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);
```

```
    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta Publish nella documentazione di riferimento dell'SDK AWS per l'API SAP ABAP.

Pubblicazione di messaggi di grandi dimensioni con Amazon SNS e Amazon S3

Per pubblicare messaggi Amazon SNS di grandi dimensioni, puoi utilizzare la Libreria client ampia di Amazon SNS per Java o la Libreria client ampia di Amazon SNS per Python. Queste librerie sono utili per i messaggi di dimensioni superiori al massimo corrente di 256 KB, fino a un massimo di 2 GB. Entrambe le librerie salvano il payload effettivo in un bucket Amazon S3 e pubblicano il riferimento dell'oggetto Amazon S3 memorizzato nell'argomento Amazon SNS. Le code Amazon SQS sottoscritte possono utilizzare il Libreria client estesa Amazon SQS per Java per annullare il riferimento e recuperare i payload da Amazon S3. Altri endpoint, ad esempio Lambda, possono utilizzare Payload Offload Java Common Library per AWS per annullare il riferimento e recuperare il payload.

> **Note**
>
> Le Amazon SNS Extended Client Libraries sono compatibili con argomenti standard e FIFO.

Argomenti

Libreria client ampia per Java

Argomenti

Prerequisiti

Di seguito sono indicati i prerequisiti per l'utilizzo di Libreria client ampia di Amazon SNS per Java:

- Un AWS SDK.

L'esempio in questa pagina utilizza l'SDK AWS Java. Per installare e configurare l'SDK, consulta Impostare il AWS SDK per Java nella AWS SDK for Java Guida per gli sviluppatori.

- E Account AWS con le credenziali appropriate.

  Per crearne uno Account AWS, vai alla AWS home page, quindi scegli Crea un AWS account. Segui le istruzioni.

  Per informazioni sulle credenziali, consulta Configurare AWS le credenziali e la regione per lo sviluppo nella Guida per gli AWS SDK for Java sviluppatori.
- Java 8 o versione successiva.
- La libreria client ampia di Amazon SNS per Java (disponibile anche da Maven).

Configurazione dello storage dei messaggi

La libreria Amazon SNS Extended Client utilizza la Payload Offloading Java Common Library AWS per l'archiviazione e il recupero dei messaggi. Puoi configurare il seguente Amazon S3 Opzioni di archiviazione dei messaggi:

- Soglia delle dimensioni dei messaggi personalizzati – I messaggi con payload e attributi che superano questa dimensione vengono memorizzati automaticamente in Amazon S3.
- `alwaysThroughS3` flag – Imposta questo valore su `true` per forzare lo storage di tutti i payload dei messaggi in Amazon S3. Ad esempio:

```
SNSExtendedClientConfiguration snsExtendedClientConfiguration = new
SNSExtendedClientConfiguration() .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME).withAlwaysThroughS3(true);
```

- Chiave KMS personalizzata – la chiave da utilizzare per la crittografia lato server nel bucket Amazon S3.
- Nome del bucket – il nome del bucket Amazon S3 per la memorizzazione dei payload dei messaggi.

Esempio: pubblicazione di messaggi su Amazon SNS con payload memorizzato in Amazon S3

L'esempio di codice seguente mostra come:

- Creare una coda e un argomento di esempio.
- Iscriviti alla coda per ricevere messaggi dall'argomento.
- Pubblicare un messaggio di prova.

Il payload del messaggio è memorizzato in Amazon S3 e il riferimento ad esso è pubblicato. Amazon SQS Extended Client viene utilizzato per ricevere il messaggio.

SDK per Java 1.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.

Per pubblicare un messaggio di grandi dimensioni, utilizza Amazon SNS Extended Client Library per Java. Il messaggio che invii fa riferimento a un oggetto Amazon S3 contenente il contenuto effettivo del messaggio.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {
```

```
        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will be
allowed to
                // be published
                // through SNS using the extended client. Payload of messages
exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest subscriptionAttributesRequest
= new SetSubscriptionAttributesRequest();
                subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);
```

```
 subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");
                snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3 when
 the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration snsExtendedClientConfiguration
 = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

 .withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
 AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it exceeds
 the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration =
 new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

Altri protocolli per endpoint

Sia le librerie Amazon SNS che Amazon SQS utilizzano la Payload Offload Java Common Library per AWS per archiviare e recuperare i payload dei messaggi con Amazon S3. Qualsiasi endpoint abilitato a Java (ad esempio, un endpoint HTTPS implementato in Java) può utilizzare la stessa libreria per annullare il riferimento al contenuto del messaggio.

Gli endpoint che non possono utilizzare la Payload Offloading Java Common Library per AWS possono comunque pubblicare messaggi con payload archiviati in Amazon S3. Di seguito è illustrato un esempio di un riferimento di Amazon S3 pubblicato dall'esempio di codice sopra:

```
[
  "software.amazon.payloadoffloading.PayloadS3Pointer",
  {
    "s3BucketName": "extended-client-bucket",
    "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
  }
]
```

Libreria client ampia per Python

Argomenti

- Prerequisiti
- Configurazione dello storage dei messaggi
- Esempio: pubblicazione di messaggi su Amazon SNS con payload memorizzato in Amazon S3

Prerequisiti

Di seguito sono indicati i prerequisiti per l'utilizzo della Libreria client ampia di Amazon SNS per Python:

- Un SDK. AWS

  L'esempio in questa pagina utilizza AWS Python SDK Boto3. Per installare e configurare l'SDK, consultare la documentazione SDK AWS per Python.
- E Account AWS con le credenziali appropriate.

  Per crearne uno Account AWS, vai alla AWS home page, quindi scegli Crea un AWS account. Segui le istruzioni.

Per ulteriori informazioni sulle credenziali, consultare Credenziali nella Guida per sviluppatori di SDK AWS per Python.

- Python 3.x (o versioni successive) e pip.
- La libreria client ampia di Amazon SNS per Python (disponibile anche da PyPI).

Configurazione dello storage dei messaggi

Gli attributi seguenti sono disponibili su Boto3 Amazon SNS Client, Topic PlatformEndpointe Objects per configurare le opzioni di storage dei messaggi di Amazon S3.

- `large_payload_support` – Il nome del bucket Amazon S3 per archiviare messaggi di grandi dimensioni.
- `message_size_threshold` – La soglia per l'archiviazione del messaggio nel bucket dei messaggi di grandi dimensioni. Non può essere minore di 0 o maggiore di 262144. Il valore predefinito è 262144.
- `always_through_s3` – Se `True`, tutti i messaggi vengono archiviati in Amazon S3. Il valore predefinito è `False`.
- `s3` – L'oggetto `resource` di Boto3 Amazon S3 utilizzato per archiviare oggetti in Amazon S3. Utilizzalo se desideri controllare la risorsa Amazon S3 (ad esempio, una configurazione o credenziali Amazon S3 personalizzate). Se non è stato impostato in precedenza al primo utilizzo, l'impostazione predefinita è `boto3.resource("s3")`.

Esempio: pubblicazione di messaggi su Amazon SNS con payload memorizzato in Amazon S3

L'esempio di codice seguente mostra come:

- Crea un argomento Amazon SNS di esempio e una coda Amazon SQS.
- Iscriviti alla coda per ricevere messaggi dall'argomento.
- Pubblicare un messaggio di prova.
- Il payload del messaggio è memorizzato in Amazon S3 e il riferimento ad esso è pubblicato.
- Stampa il messaggio pubblicato dalla coda insieme al messaggio originale recuperato da Amazon S3.

Per pubblicare un messaggio di grandi dimensioni, utilizza la Libreria client ampia di Amazon SNS per Python. Il messaggio che invii fa riferimento a un oggetto Amazon S3 contenente il contenuto effettivo del messaggio.

```
import boto3
import sns_extended_client
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store"
TOPIC_NAME = "---TOPIC-NAME---"
QUEUE_NAME = "---QUEUE-NAME---"

# Create an helper to fetch message from S3
def get_msg_from_s3(body):
    json_msg = loads(body)
    s3_client = boto3.client("s3")
    s3_object = s3_client.get_object(
        Bucket=json_msg[1].get("s3BucketName"), Key=json_msg[1].get("s3Key")
    )
    msg = s3_object.get("Body").read().decode()
    return msg

# Create an helper to fetch and print message SQS queue and S3
def fetch_and_print_from_sqs(sqs, queue_url):
    """Handy Helper to fetch and print message from SQS queue and S3"""
    message = sqs.receive_message(
        QueueUrl=queue_url, MessageAttributeNames=["All"], MaxNumberOfMessages=1
    ).get("Messages")[0]
    message_body = message.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is: {}\n".format(get_msg_from_s3(message_body)))

# Initialize the SNS client and create SNS Topic
sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
demo_topic_arn = create_topic_response.get("TopicArn")

# Create and subscribe an SQS queue to the SNS client
sqs = boto3.client("sqs")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
demo_queue_arn = sqs.get_queue_attributes(QueueUrl=demo_queue_url,
AttributeNames=["QueueArn"])["Attributes"].get("QueueArn")
# Set the RawMessageDelivery subscription attribute to TRUE
```

```
sns_extended_client.subscribe(TopicArn=demo_topic_arn, Protocol="sqs",
Endpoint=demo_queue_arn, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# To store all messages content in S3, set always_through_s3 to True
# In the example, we set message size threshold as 32 bytes, adjust this threshold as
 per your usecase
# Message will only be uploaded to S3 when its payload size exceeded threshold
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=demo_topic_arn,
    Message="This message should be published to S3 as it exceeds the
 message_size_threshold limit",
)
# Print message stored in s3
fetch_and_print_from_sqs(sqs, demo_queue_url)
```

Output

```
Published Message:
[
    "software.amazon.payloadoffloading.PayloadS3Pointer",
    {
        "s3BucketName": "extended-client-bucket-store",
        "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
    }
]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the message_size_threshold limit
```

Attributi messaggio di Amazon SNS

Amazon SNS supporta la consegna di attributi di messaggio che consentono di fornire elementi di metadati strutturati (come timestamp, dati geospaziali, firme e identificatori) relativi al messaggio. Per gli abbonamenti SQS, è possibile inviare un massimo di 10 attributi di messaggio quando Consegna di messaggi non elaborati è abilitato. Per inviare più di 10 attributi di messaggio, è necessario disabilitare Consegna di messaggi non elaborati. I messaggi con più di 10 attributi indirizzati verso abbonamenti Amazon SQS abilitati alla consegna di messaggi non elaborati verranno eliminati come errori lato client.

Gli attributi di messaggio sono facoltativi e separati dal corpo del messaggio, sebbene vengano inviati insieme a esso. Il destinatario può utilizzare queste informazioni per decidere come gestire il messaggio senza dover prima elaborare il corpo del messaggio.

Per ulteriori informazioni sull'invio di messaggi con attributi tramite AWS Management Console o AWS SDK for Java, consulta il tutorial Per pubblicare messaggi su argomenti Amazon SNS utilizzando il AWS Management Console.

> **Note**
>
> Gli attributi di messaggio vengono inviati solo quando la struttura del messaggio è String, non JSON.

Puoi utilizzare gli attributi di messaggio anche per strutturare il messaggio di notifica push per gli endpoint mobili. In questo scenario, gli attributi di messaggio sono usati solo per strutturare il messaggio di notifica push. Gli attributi non vengono distribuiti all'endpoint così come sono quando si inviano i messaggi con gli attributi di messaggio agli endpoint Amazon SQS.

Puoi utilizzare gli attributi di messaggio anche per rendere i messaggi filtrabili utilizzando le policy di filtro della sottoscrizione. Le policy di filtro possono essere applicate alle sottoscrizioni dell'argomento. Quando viene applicata una policy di filtro con l'ambito impostato su `MessageAttributes` (valore predefinito), una sottoscrizione riceve solo i messaggi che hanno gli attributi accettati dalla policy. Per ulteriori informazioni, consulta Filtraggio messaggi di Amazon SNS.

> **Note**
>
> Quando si utilizzano gli attributi di messaggio per il filtro, il valore deve essere una stringa JSON valida. Questo garantisce che il messaggio venga distribuito a una sottoscrizione con il filtro degli attributi di messaggio abilitato.

Elementi dell'attributo di messaggio e convalida

Ogni attributo di messaggio è costituito dai seguenti elementi:

- Name – Il nome dell'attributo del messaggio può contenere i seguenti caratteri: A-Z, a-z, 0-9, sottolineatura (_), trattino (-) e punto (.). Il nome non deve iniziare o terminare con un punto e

non deve avere punti in successione. Il nome rispetta la distinzione tra maiuscole e minuscole e deve essere univoco tra tutti i nomi di attributo per il messaggio. Il nome può contenere fino a 256 caratteri. Il nome non può iniziare con `AWS.` o `Amazon.` (o qualsiasi variazione in maiuscole/minuscole) perché questi prefissi sono riservati per l'uso da parte di Amazon Web Services.

- Type – I tipi di dati di attributo di messaggio supportati sono `String`, `String.Array`, `Number` e `Binary`. Il tipo di dati ha le stesse restrizioni sul contenuto del corpo del messaggio. Il tipo di dati rispetta la distinzione tra maiuscole e minuscole e può avere una lunghezza massima di 256 byte. Per ulteriori informazioni, consulta la sezione Tipi di dati dell'attributo di messaggio e convalida.
- Value – Il valore dell'attributo di messaggio specificato dall'utente. Per i tipi di dati di stringa, l'attributo del valore ha le stesse restrizioni sul contenuto del corpo del messaggio. Per ulteriori informazioni, consulta l'operazione Pubblicare nella Documentazione di riferimento API di Amazon Simple Notification Service.

Nome, tipo e valore non devono essere vuoti o nulli. Inoltre, il corpo del messaggio non deve essere vuoto o nullo. Tutte le parti dell'attributo di messaggio, inclusi nome, tipo e valore, sono incluse nella limitazione della dimensione del messaggio che è di 256 KB.

Tipi di dati dell'attributo di messaggio e convalida

I tipi di dati degli attributi di messaggio identificano il modo in cui vengono gestiti i valori degli attributi dei messaggi da Amazon SNS. Ad esempio, se il tipo è un numero, Amazon SNS convalida che si tratta di un numero.

Amazon SNS supporta i seguenti tipi di dati logici per tutti gli endpoint tranne quanto indicato:

- String – Le stringhe sono Unicode con codifica binaria UTF-8. Per l'elenco dei valori di codifica, vedi http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters.

  > **Note**
  >
  > I valori surrogati non sono supportati negli attributi del messaggio. Ad esempio, l'utilizzo di un valore surrogato per rappresentare un'emoji ti darà il seguente errore: `Invalid attribute value was passed in for message attribute`.

- String.Array: una matrice, formattata come una stringa, che può contenere più valori. I valori possono essere stringhe, numeri o le parole chiave `true`, `false` e `null`. Uno String.Array di tipo numerico o booleano non richiede virgolette. I valori String.Array sono separati da virgole.

Questo tipo di dati non è supportato per AWS Lambda abbonamenti. Se si specifica questo tipo di dati per gli endpoint Lambda, viene passato come `String` tipo di dati nel payload JSON che Amazon SNS consegna a Lambda.

- Number – I numeri sono interi positivi o negativi oppure in virgola mobile. I numeri hanno un intervallo e una precisione sufficienti per comprendere la maggior parte dei possibili valori supportati in genere da valori interi, a virgola mobile e doppi. Un numero può avere un valore compreso fra -10^9 e 10^9, con 5 cifre di precisione dopo la virgola. Gli zero iniziali e finali vengono tagliati.

  Questo tipo di dati non è supportato per AWS Lambda abbonamenti. Se si specifica questo tipo di dati per gli endpoint Lambda, viene passato come `String` tipo di dati nel payload JSON che Amazon SNS consegna a Lambda.
- Binary – Gli attributi di tipo binario possono archiviare qualsiasi tipo di dati binari, ad esempio dati compressi, dati crittografati o immagini.

Attributi di messaggio riservati per notifiche push per dispositivi mobili

Nella tabella seguente sono elencati gli attributi di messaggio riservati per i servizi di notifica push di dispositivi mobili che puoi usare per strutturare il messaggio di notifica push:

Servizio di notifiche push	Attributo di messaggio riservato
ADM	`AWS.SNS.MOBILE.ADM.TTL`
APN[1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`
	`AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL`

Servizio di notifiche push	Attributo di messaggio riservato
	AWS.SNS.MOBILE.APNS.COLLAPSE_ID
	AWS.SNS.MOBILE.APNS.PRIORITY
	AWS.SNS.MOBILE.APNS.PUSH_TYPE
	AWS.SNS.MOBILE.APNS.TOPIC
	AWS.SNS.MOBILE.APNS.TTL
Baidu	AWS.SNS.MOBILE.BAIDU.DeployStatus
	AWS.SNS.MOBILE.BAIDU.MessageKey
	AWS.SNS.MOBILE.BAIDU.MessageType
	AWS.SNS.MOBILE.BAIDU.TTL
FCM	AWS.SNS.MOBILE.FCM.TTL
	AWS.SNS.MOBILE.GCM.TTL
macOS	AWS.SNS.MOBILE.MACOS_SANDBOX.TTL
	AWS.SNS.MOBILE.MACOS.TTL
MPNS	AWS.SNS.MOBILE.MPNS.NotificationClass
	AWS.SNS.MOBILE.MPNS.TTL
	AWS.SNS.MOBILE.MPNS.Type
WNS	AWS.SNS.MOBILE.WNS.CachePolicy
	AWS.SNS.MOBILE.WNS.Group
	AWS.SNS.MOBILE.WNS.Match
	AWS.SNS.MOBILE.WNS.SuppressPopup

Servizio di notifiche push	Attributo di messaggio riservato
	`AWS.SNS.MOBILE.WNS.Tag`
	`AWS.SNS.MOBILE.WNS.TTL`
	`AWS.SNS.MOBILE.WNS.Type`

[1] Apple rifiuterà le notifiche Amazon SNS se gli attributi di messaggio non soddisfano i requisiti. Per ulteriori informazioni, consultare Invio di richieste di notifica agli APN sul sito Web degli sviluppatori Apple.

Batch di messaggi Amazon SNS

Cos'è il batch di messaggi?

Un'alternativa alla pubblicazione di messaggi su argomenti Standard o FIFO in richieste API `Publish` individuali, è l'utilizzo dell'API `PublishBatch` di Amazon SNS per pubblicare fino a 10 messaggi in una singola richiesta API. L'invio di messaggi in batch può aiutare a ridurre i costi associati alla connessione di applicazioni distribuite (Messaggistica A2A) o all'invio di notifiche alle persone (Messaggistica A2P) con Amazon SNS di un fattore fino a 10. Amazon SNS ha quote su quanti messaggi è possibile pubblicare su un argomento al secondo in base alla regione in cui operi. Consultare Amazon SNS endpoints and quotas (Endpoint e quote di Amazon SNS) nella guida Riferimenti generali di AWS per ulteriori informazioni sulle quote delle API.

> **Note**
>
> La dimensione totale di tutti i messaggi inviati in una singola richiesta API `PublishBatch` non può superare 262.144 byte (256 KB).
> L'API `PublishBatch` utilizza lo stesso operazione API `Publish` per le policy IAM.

Come funziona il batch di messaggi?

Pubblicazione di messaggi con l'API `PublishBatch` è simile alla pubblicazione di messaggi con l'API `Publish`. La differenza principale è che ogni messaggio all'interno di una richiesta API `PublishBatch` deve essere assegnato un ID batch univoco (fino a 80 caratteri). In questo modo,

Amazon SNS può restituire risposte API individuali per ogni messaggio all'interno di un batch per confermare che ogni messaggio è stato pubblicato o che si è verificato un errore. Per i messaggi pubblicati su argomenti FIFO, oltre a includere l'assegnazione di un ID batch univoco, è comunque necessario includere un `MessageDeduplicationID` e `MessageGroupId` per ogni singolo messaggio.

Esempi

Pubblicazione di un batch di 10 messaggi su un argomento Standard

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
```

```
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });
    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

Pubblicazione di un batch di 10 messaggi su un argomento FIFO

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToFifoTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
```

```
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i)
                        .withMessageGroupId("groupId")
                        .withMessageDeduplicationId("deduplicationId" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
publishBatchResultEntry.getMessageId());
            System.out.println("SequenceNumber for successful message: " +
publishBatchResultEntry.getSequenceNumber());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
batchResultErrorEntry.getMessage());
        });

    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
```

```
}
```

Filtraggio messaggi di Amazon SNS

Per impostazione predefinita, un sottoscrittore di un argomento Amazon SNS riceve ogni messaggio pubblicato nell'argomento. Per ricevere solo un sottoinsieme dei messaggi, un sottoscrittore deve assegnare una policy di filtro alla sottoscrizione all'argomento.

Una policy di filtro è un oggetto JSON contenente proprietà che definiscono i messaggi ricevuti dal sottoscrittore. Amazon SNS supporta policy che agiscono sugli attributi o sul corpo del messaggio, in base all'ambito della policy di filtro che imposti per la sottoscrizione. Le policy di filtro per il corpo del messaggio presuppongono che il payload del messaggio sia un oggetto JSON ben formato.

Se non dispone di una policy di filtro, il sottoscrittore riceve ogni messaggio pubblicato nell'argomento. Quando pubblichi un messaggio in un argomento con una policy di filtro impostata, Amazon SNS confronta gli attributi o il corpo del messaggio con le proprietà specificate nella policy di filtro per ciascuna delle sottoscrizioni all'argomento. Se viene rilevata la corrispondenza tra gli attributi o il corpo del messaggio, Amazon SNS invia il messaggio al sottoscrittore. In assenza di corrispondenza, Amazon SNS non invia il messaggio al sottoscrittore.

Per ulteriori informazioni, consulta Filtro dei messaggi pubblicati negli argomenti.

Argomenti

Ambito delle policy di filtro per le sottoscrizioni Amazon SNS

L'attributo della sottoscrizione `FilterPolicyScope` consente di scegliere l'ambito del filtro impostando uno dei seguenti valori:

- `MessageAttributes`: la policy di filtro viene applicata agli attributi del messaggio. Questa è l'impostazione predefinita.
- `MessageBody`: la policy di filtro viene applicata al corpo del messaggio.

> **Note**
>
> Se per una policy di filtro esistente non è definito il relativo ambito, l'ambito predefinito è `MessageAttributes`.

Policy di filtro degli abbonamenti Amazon SNS

Una policy di filtro per le sottoscrizioni consente di specificare nomi di proprietà e di assegnare un elenco di valori a ciascuno di questi nomi. Per ulteriori informazioni, consulta Filtraggio messaggi di Amazon SNS.

Quando Amazon SNS valuta gli attributi del messaggio o le proprietà del corpo del messaggio rispetto alla policy di filtro per le sottoscrizioni, ignora quelli non specificati nella policy.

> **Important**
>
> AWS servizi come IAM e Amazon SNS utilizzano un modello di calcolo distribuito chiamato eventuale consistenza. Le aggiunte o le modifiche a una policy di filtro sottoscrizione richiedono fino a 15 minuti per essere pienamente effettive.

Una sottoscrizione accetta un messaggio nelle seguenti condizioni:

- Quando l'ambito della policy di filtro è impostato su `MessageAttributes`, ogni nome di proprietà nella policy di filtro corrisponde al nome dell'attributo del messaggio. Per ogni nome di proprietà corrispondente nella policy di filtro, almeno un valore di proprietà corrisponde al valore dell'attributo del messaggio.
- Quando l'ambito della policy di filtro è impostato su `MessageBody`, ogni nome di proprietà nella policy di filtro corrisponde al nome della proprietà del corpo del messaggio. Per ogni nome di proprietà corrispondente nella policy di filtro, almeno un valore di proprietà corrisponde al valore della proprietà del corpo del messaggio.

Attualmente Amazon RDS supporta i seguenti operatori di filtro:

- Logica AND
- Logica OR

- Operatore OR
- Corrispondenza di chiave
- Corrispondenza esatta dei valori numerici
- Corrispondenza Anything-but dei valori numerici
- Corrispondenza intervallo dei valori numerici
- Corrispondenza esatta dei valori di stringa
- Corrispondenza Anything-but delle stringhe
- Corrispondenza di stringhe utilizzando un prefisso con l'operatore anything-but
- equals-ignore case dei valori di stringa
- Corrispondenza dell'indirizzo IP dei valori di stringa
- Corrispondenza del prefisso dei valori di stringa
- Corrispondenza del suffisso dei valori di stringa

Esempi di policy di filtro

L'esempio seguente mostra un payload di messaggio inviato da un argomento Amazon SNS che elabora transazioni di clienti.

Il primo esempio include il campo `MessageAttributes`, che presenta attributi che descrivono la transazione:

- Interessi del cliente
- Nome dello store
- Stato dell'evento
- Prezzo di acquisto in USD

Poiché questo messaggio include il campo `MessageAttributes`, qualsiasi sottoscrizione all'argomento che imposta un `FilterPolicy` può accettare o rifiutare in modo selettivo il messaggio, a condizione che `FilterPolicyScope` sia impostato su `MessageAttributes` nella sottoscrizione. Per informazioni sull'applicazione di attributi a un messaggio, consulta Attributi messaggio di Amazon SNS.

```
{
   "Type": "Notification",
```

```
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "message-body-with-transaction-details",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url",
   "MessageAttributes": {
      "customer_interests": {
         "Type": "String.Array",
         "Value": "[\"soccer\", \"rugby\", \"hockey\"]"
      },
      "store": {
         "Type": "String",
         "Value":"example_corp"
      },
      "event": {
         "Type": "String",
         "Value": "order_placed"
      },
      "price_usd": {
         "Type": "Number",
         "Value": "210.75"
      }
   }
}
```

L'esempio seguente mostra gli stessi attributi inclusi nel campo `Message`, denominato anche payload del messaggio o corpo del messaggio. Qualsiasi sottoscrizione a un argomento che includa un `FilterPolicy` può accettare o rifiutare in modo selettivo il messaggio, a condizione che `FilterPolicyScope` sia impostato su `MessageBody` nella sottoscrizione.

```
{
"Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "{
      \"customer_interests\": [\"soccer\", \"rugby\", \"hockey\"],
      \"store\": \"example_corp\",
      \"event\":\"order_placed\",
      \"price_usd\":210.75
   }",
   "Timestamp": "2019-11-03T23:28:01.631Z",
```

```
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url"
}
```

Le seguenti policy di filtro accettano o rifiutano i messaggi in base ai relativi nomi e valori delle proprietà.

Policy che accetta il messaggio di esempio

Le proprietà nella seguente policy di filtro per le sottoscrizioni corrispondono a quelli assegnati al messaggio di esempio. È importante notare che la stessa policy di filtro funziona per un `FilterPolicyScope` indipendentemente dal fatto che sia impostata su `MessageAttributes` o su `MessageBody`. Ogni sottoscrittore sceglie il proprio ambito di filtro in base alla composizione dei messaggi che riceve dall'argomento.

Se una singola proprietà specificata in questa policy non corrisponde a un attributo assegnato al messaggio, la policy rifiuta il messaggio.

```
{
   "store": ["example_corp"],
   "event": [{"anything-but": "order_cancelled"}],
   "customer_interests": [
      "rugby",
      "football",
      "baseball"
   ],
   "price_usd": [{"numeric": [">=", 100]}]
}
```

Policy che rifiuta il messaggio di esempio

La policy seguente presenta diverse mancate corrispondenze tra le sue proprietà e quelle assegnate al messaggio di esempio. Ad esempio, poiché il nome della proprietà `encrypted` non figura fra gli attributi del messaggio, tale proprietà della policy causa il rifiuto del messaggio, indipendentemente dal valore a esso assegnato.

Se si verifica una qualsiasi mancata corrispondenza, la policy rifiuta il messaggio.

```
{
   "store": ["example_corp"],
```

```
   "event": ["order_cancelled"],
   "encrypted": [false],
   "customer_interests": [
      "basketball",
      "baseball"
   ]
}
```

Vincoli delle policy di filtro

Quando crei una politica di filtro per un abbonamento Amazon SNS, è importante capire come vengono conteggiate le chiavi della policy. Gli aspetti chiave da tenere a mente sono:

1. Chiavi principali: le chiavi principali sono le chiavi di primo livello nella politica di filtro. Queste sono le chiavi per le quali si specificano valori o vincoli.
2. Nomi degli attributi: le chiavi principali sono considerate i nomi degli attributi nella politica di filtro. I valori o i vincoli specificati per queste chiavi verranno applicati agli attributi corrispondenti nel payload del messaggio.
3. Valori validi: i valori specificati per le chiavi principali devono essere una stringa, una matrice di stringhe o un numero. Se il valore è un oggetto (ad esempio un oggetto JSON), non verrà conteggiato come chiave valida nella politica di filtro.

Consideriamo il seguente esempio di politica di filtro:

```
{
  "state": ["SUCCESS"],
  "severity": [{ "exists": true }],
  "message": [{ "exists": true }],
  "finding": {
    "standard_control": [{ "exists": true }],
    "region": [{ "exists": true }],
    "account": [{ "exists": true }]
  }
}
```

In questo esempio, le seguenti chiavi vengono conteggiate come parte della politica di filtro:

- `state`
- `severity`

- `message`
- `standard_control`
- `region`
- `account`

La ricerca della chiave non viene conteggiata, in quanto contiene un oggetto JSON come valore, anziché una stringa, una matrice di stringhe o un numero.

Un altro esempio:

```
{
  "key_a": {
    "key_b": {
      "key_c": {
        "key_d": ["value_one", "value_two", "value_three", "value_four"]
      }
    },
    "key_e": {
      "key_f": ["value_one", "value_two", "value_three"]
    }
  }
}
```

In questo caso, solo le chiavi `key_d` `key_f` vengono conteggiate come parte della politica di filtro, poiché a loro sono assegnati valori che sono una stringa o una matrice di stringhe. Le chiavi `key_a` principali non `key_c` vengono conteggiate, in quanto contengono oggetti JSON annidati come valori. `key_b`

Argomenti

- Vincoli comuni delle policy
- Vincoli delle policy per il filtro basato sugli attributi
- Vincoli delle policy per il filtro basato su payload

Vincoli comuni delle policy

- Corrispondenza delle stringhe: per la corrispondenza delle stringhe nella politica di filtro, il confronto fa distinzione tra maiuscole e minuscole.

- Corrispondenza numerica: per la corrispondenza numerica, il valore può variare da -10^{9} a 10^{9} (da -1 miliardo a 1 miliardo), con una precisione di cinque cifre dopo la virgola decimale.
- Complessità della politica di filtro: per la complessità della politica di filtro, la combinazione totale di valori non deve superare 150. Per calcolare la combinazione totale, moltiplica il numero di valori in ogni array nella politica di filtro.

Considerate il seguente esempio di politica:

```
{
   "key_a": ["value_one", "value_two", "value_three"],
   "key_b": ["value_one"],
   "key_c": ["value_one", "value_two"]
}
```

In questa politica:

- Il primo array ha 3 valori
- La seconda matrice ha 1 valore
- La terza matrice ha 2 valori

La combinazione totale si calcola nel seguente modo:

- 3 x 1 x 2 = 6

Sintassi della politica di filtro

Il JSON della policy di filtro può contenere quanto segue:

- Stringhe tra virgolette
- Numeri
- Le parole chiave `true`, `false` e `null` senza virgolette

Quando usi l'API Amazon SNS, devi passare il codice JSON della policy di filtro come stringa UTF-8 valida.

Limiti delle politiche di filtro

- La dimensione massima di una politica di filtro è 256 KB.
- Per impostazione predefinita, puoi avere fino a 200 politiche di filtro per argomento e 10.000 politiche di filtro per AWS account.
- Questo limite di policy non impedirebbe la creazione di abbonamenti alla coda Amazon SQS con l'API. `Subscribe` Tuttavia, l'operazione avrebbe esito negativo quando si collega la policy di filtro alla chiamata API `Subscribe` (o alla chiamata API `SetSubscriptionAttributes`).
- Per richiedere un aumento della quota, è possibile utilizzare AWS Service Quotas.

Vincoli delle policy per il filtro basato sugli attributi

- Il filtro basato sugli attributi è l'opzione predefinita. `FilterPolicyScope` è impostato su `MessageAttributes` nella sottoscrizione.
- Amazon SNS non accetta una policy di filtri annidati per il filtro basato su attributi.
- Amazon SNS confronta le proprietà della policy solo con gli attributi di messaggio che hanno i seguenti tipi di dati:
  - `String`
  - `String.Array`

    > **Important**
    >
    > Il passaggio di oggetti negli array non è consigliato in quanto potrebbe produrre risultati imprevisti a causa della nidificazione, che non è supportata dal filtraggio basato sugli attributi. Utilizzo del filtraggio basato sul payload per le policy nidificate.

  - `Number`
- Amazon SNS ignora gli attributi di messaggio con tipo di dati `Binary`.
- Una policy di filtro può contenere fino a cinque nomi di attributo.

Vincoli delle policy per il filtro basato su payload

Amazon SNS accetta una politica di filtro annidata per il filtraggio basato sul payload. Per calcolare la combinazione totale di valori nella politica di filtro, moltiplica il numero di valori in ogni array annidato.

Considerate il seguente esempio di politica:

```
{
```

```
"key_a": {
    "key_b": {
        "key_c": ["value_one", "value_two", "value_three", "value_four"]
        }
    },
"key_d": {
    "key_e": ["value_one", "value_two", "value_three"]
    }
}
```

In questa politica:

- Il primo array ha quattro valori in una chiave annidata a tre livelli.
- Il secondo ha tre valori in una chiave nidificata a due livelli.

La combinazione totale si calcola nel seguente modo:

- 4 x 3 x 3 x 2 = 72

Limiti delle politiche

Una politica di filtro può avere un massimo di cinque chiavi principali (chiavi di primo livello). Per una politica annidata, solo le chiavi principali vengono conteggiate ai fini del limite di cinque chiavi.

Intervallo numerico

Per la corrispondenza numerica nella politica di filtro, il valore può variare da -10 ^{9} a 10 ^{9} (da -1 miliardo a 1 miliardo), con una precisione di cinque cifre dopo la virgola decimale.

Passaggio al filtraggio basato sul payload

Per passare dal filtro basato sugli attributi (predefinito) al filtro basato sul payload, è necessario impostare `FilterPolicyScope` su `MessageBody` nella sottoscrizione.

Logica AND/OR

Puoi utilizzare operazioni che includono gli operatori logici AND/OR per creare corrispondenze tra le proprietà del corpo del messaggio.

Argomenti

- Logica AND

- Logica OR
- Operatore OR

Logica AND

Puoi applicare l'operatore logico AND utilizzando più nomi di proprietà.

Esaminiamo la seguente policy:

```
{
  "customer_interests": ["rugby"],
  "price_usd": [{"numeric": [">", 100]}]
}
```

Corrisponde a qualsiasi attributo di messaggio o corpo del messaggio che abbia il valore di `customer_interests` impostato su `rugby` e il valore di `price_usd` impostato su un numero superiore a 100.

> **Note**
>
> Non è possibile applicare la logica AND ai valori dello stesso attributo.

Logica OR

Puoi applicare l'operatore logico OR assegnando più valori a un nome di proprietà.

Esaminiamo la seguente policy:

```
{
   "customer_interests": ["rugby", "football", "baseball"]
}
```

Corrisponde a qualsiasi attributo di messaggio o proprietà del corpo di messaggio con il valore di `customer_interests` impostato su `rugby`, `football` o `baseball`.

Operatore OR

È possibile utilizzare l'operatore `"$or"` per definire in modo esplicito una policy di filtro per esprimere la relazione OR tra più attributi della policy.

Amazon SNS riconosce una relazione `"$or"` solo quando la policy soddisfa tutte le seguenti condizioni. Se tutte queste condizioni non sono soddisfatte, `"$or"` viene considerato come un normale nome di attributo, come qualsiasi altra stringa della policy.

- Nella regola seguita da un array è presente un attributo di campo `"$or"`, ad esempio, `“$or” : []`.
- Ci sono almeno 2 oggetti nell'array `"$or"`: `"$or": [{}, {}]`.
- Nessuno degli oggetti nell'array `"$or"` ha nomi di campo che sono parole chiave riservate.

Altrimenti `"$or"` viene considerato come un normale nome di attributo, come le altre stringhe della policy.

La seguente policy non viene analizzata come una relazione OR perché numerico e prefisso sono parole chiave riservate.

```
{
   "$or": [ {"numeric" : 123}, {"prefix": "abc"} ]
}
```

Esempi di operatori **OR**

0R standard:

```
{
  "source": [ "aws.cloudwatch" ],
  "$or": [
    { "metricName": [ "CPUUtilization" ] },
    { "namespace": [ "AWS/EC2" ] }
  ]
}
```

La logica di filtro per questa policy è:

```
"source" && ("metricName" || "namespace")
```

Corrisponde a uno dei seguenti set di attributi di messaggio:

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"metricName": {"Type": "String", "Value": "CPUUtilization"}
```

oppure

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"namespace": {"Type": "String", "Value": "AWS/EC2"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
    "source": "aws.cloudwatch",
    "metricName": "CPUUtilization"
}
```

oppure

```
{
    "source": "aws.cloudwatch",
    "namespace": "AWS/EC2"
}
```

Vincoli della policy che includono le relazioni **OR**

Esaminiamo la seguente policy:

```
{
    "source": [ "aws.cloudwatch" ],
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      {
        "metricType": [ "MetricType" ] ,
        "$or" : [
          { "metricId": [ 1234, 4321 ] },
          { "spaceId": [ 1000, 2000, 3000 ] }
        ]
      }
    ]
  }
```

La logica di questa policy può anche essere semplificata come segue:

```
("source" AND "metricName")
OR
```

```
("source" AND "metricType" AND "metricId")
OR
("source" AND "metricType" AND "spaceId")
```

Il calcolo della complessità per le policy con relazioni OR può essere semplificato come somma delle complessità di combinazioni per ogni istruzione OR.

La combinazione totale si calcola nel seguente modo:

```
(source * metricName) + (source * metricType * metricId) + (source * metricType *
 spaceId)
= (1 * 2) + (1 * 1 * 2) + (1 * 1 * 3)
= 7
```

`source` ha un valore, `metricName` ha due valori, `metricType` ha un valore, `metricId` ha due valori e `spaceId` ha tre valori.

Considera la seguente policy di filtro nidificata:

```
{
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      { "namespace": [ "AWS/EC2", "AWS/ES" ] }
    ],
    "detail" : {
      "scope" : [ "Service" ],
      "$or": [
        { "source": [ "aws.cloudwatch" ] },
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

La logica di questa policy può essere semplificata come segue:

```
("metricName" AND ("detail"."scope" AND "detail"."source")
OR
("metricName" AND ("detail"."scope" AND "detail"."type")
OR
("namespace" AND ("detail"."scope" AND "detail"."source")
OR
```

```
("namespace" AND ("detail"."scope" AND "detail"."type")
```

Il calcolo per le combinazioni totali è lo stesso per le policy non nidificate, tranne per il fatto che è necessario considerare il livello di nidificazione di una chiave.

La combinazione totale si calcola nel seguente modo:

```
(2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) = 32
```

`metricName` ha due valori, `namespace` ha due valori, `scope` è una chiave nidificata a due livelli con un valore, `source` è una chiave nidificata a due livelli con un valore e `type` è una chiave nidificata a due livelli con un valore.

Corrispondenza di chiave

Puoi utilizzare l'operatore `exists` per creare corrispondenze con i messaggi in arrivo con o senza proprietà specificate nella policy di filtro: la corrispondenza `exists` funziona solo su nodi foglia. Non funziona sui nodi intermedi.

- Utilizza `"exists": true` per creare corrispondenze con i messaggi in arrivo che includono la proprietà specificata. La chiave deve avere un valore non null e non vuoto.

  Ad esempio, la seguente proprietà di policy utilizza l'operatore `exists` con un valore di `true`:

  ```
  "store": [{"exists": true}]
  ```

  Corrisponde a qualsiasi elenco di attributi di messaggi contenente la chiave attributo `store`, ad esempio:

  ```
  "store": {"Type": "String", "Value": "fans"}
  "customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
  \"]"}
  ```

  Corrisponde anche a uno dei seguenti corpi di messaggi:

  ```
  {
      "store": "fans"
      "customer_interests": ["baseball", "basketball"]
  }
  ```

Tuttavia, non corrisponde a nessun elenco di attributi di messaggi senza la chiave attributo `store`, ad esempio:

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

- Utilizza `"exists": false` per creare corrispondenze con i messaggi in arrivo che non includono la proprietà specificata.

> **Note**
>
> `"exists": false` genera corrispondenze solo se è presente almeno un attributo. Un set vuoto di attributi non consente al filtro di generare corrispondenze.

Ad esempio, la seguente proprietà di policy utilizza l'operatore `exists` con un valore di `false`:

```
"store": [{"exists": false}]
```

Non corrisponde a nessun elenco di attributi di messaggi contenente la chiave attributo `store`, ad esempio:

```
"store": {"Type": "String", "Value": "fans"}
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Non corrisponde nemmeno al seguente corpo del messaggio:

```
{
    "store": "fans"
    "customer_interests": ["baseball", "basketball"]
}
```

Tuttavia, corrisponde a qualsiasi elenco di attributi di messaggi senza la chiave attributo `store`, ad esempio:

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Corrisponde anche al seguente corpo del messaggio:

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

Corrispondenza dei valori numerici

È possibile filtrare i messaggi creando una corrispondenza tra i valori numerici e i valori degli attributi del messaggio o i valori delle proprietà del corpo del messaggio. Nella policy JSON, i valori numerici non sono racchiusi tra virgolette doppie. È possibile disporre delle seguenti operazioni numeriche per il filtro.

> **Note**
>
> I prefissi sono supportati solo per la corrispondenza di stringa.

Argomenti

- Corrispondenza esatta
- Corrispondenza anything-but
- Corrispondenza dell'intervallo dei valori

Corrispondenza esatta

Quando un valore di proprietà della policy include la parola chiave `numeric` e l'operatore =, corrisponde a qualsiasi attributo di messaggio o proprietà del corpo del messaggio con lo stesso nome e lo stesso valore numerico.

Esaminiamo la seguente proprietà della policy:

```
"price_usd": [{"numeric": ["=",301.5]}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"price_usd": {"Type": "Number", "Value": 301.5}
```

```
"price_usd": {"Type": "Number", "Value": 3.015e2}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
   "price_usd": 301.5
}
```

```
{
   "price_usd": 3.015e2
}
```

Corrispondenza anything-but

Quando un valore della proprietà della policy include la parola chiave `anything-but`, corrisponde a qualsiasi attributo del messaggio o valore del corpo del messaggio che non include nessuno dei valori delle proprietà della policy.

Esaminiamo la seguente proprietà della policy:

```
"price": [{"anything-but": [100, 500]}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"price": {"Type": "Number", "Value": 101}
```

```
"price": {"Type": "Number", "Value": 100.1}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
```

```
   "price": 101
}
```

```
{
   "price": 100.1
}
```

Inoltre, corrisponde anche al seguente attributo di messaggio (poiché contiene un valore che non è `100` o `500`):

```
"price": {"Type": "Number.Array", "Value": "[100, 50]"}
```

E corrisponde anche al seguente corpo del messaggio (poiché contiene un valore che non è `100` né `500`):

```
{
   "price": [100, 50]
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"price": {"Type": "Number", "Value": 100}
```

Né corrisponde al seguente corpo del messaggio:

```
{
   "price": 100
}
```

Corrispondenza dell'intervallo dei valori

Oltre all'operatore =, una proprietà di policy numerica può includere i seguenti operatori: <, <=, > e >=.

Esaminiamo la seguente proprietà della policy:

```
"price_usd": [{"numeric": ["<", 0]}]
```

Corrisponde a qualsiasi attributo di messaggio o proprietà del corpo del messaggio che abbia valori numerici negativi.

Esaminiamo un altro attributo di messaggio:

```
"price_usd": [{"numeric": [">", 0, "<=", 150]}]
```

Corrisponde a qualsiasi attributo di messaggio o proprietà del corpo del messaggio che abbia numeri positivi fino a 150.

Corrispondenza dei valori di stringa

Puoi filtrare i messaggi creando una corrispondenza tra i valori della stringa e i valori degli attributi del messaggio o i valori delle proprietà del corpo del messaggio. Nella policy JSON, i valori di stringa sono racchiusi tra virgolette doppie. Puoi utilizzare le seguenti operazioni di stringa per creare una corrispondenza tra gli attributi del messaggio o il corpo del messaggio.

Argomenti

Corrispondenza esatta

La corrispondenza esatta si verifica quando un valore di proprietà della policy corrisponde a uno o più valori di attributo del messaggio.

Esaminiamo la seguente proprietà della policy:

```
"customer_interests": ["rugby", "tennis"]
```

Corrisponde ai seguenti attributi di messaggio:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String", "Value": "tennis"}
```

Corrisponde anche ai seguenti corpi dei messaggi:

```
{
   "customer_interests": "rugby"
}
```

```
{
   "customer_interests": "tennis"
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
   "customer_interests": "baseball"
}
```

Corrispondenza anything-but

Quando un valore della proprietà della policy include la parola chiave `anything-but`, corrisponde a qualsiasi attributo del messaggio o valore del corpo del messaggio che non include nessuno dei valori delle proprietà della policy. `anything-but` può essere combinato con `"exists": false`.

Esaminiamo la seguente proprietà della policy:

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "football"
}
```

Inoltre, corrisponde anche al seguente attributo di messaggio (poiché contiene un valore che non è `rugby` o `tennis`):

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

E corrisponde anche al seguente corpo del messaggio (poiché contiene un valore che non è `rugby` né `tennis`):

```
{
   "customer_interests": ["rugby", "baseball"]
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
   "customer_interests": ["rugby"]
}
```

Utilizzo di un prefisso con operatore **`anything-but`**

Per la corrispondenza di stringa, puoi anche utilizzare un prefisso con operatore `anything-but`. Ad esempio, la proprietà della policy seguente nega il prefisso `order-`:

```
"event":[{"anything-but": {"prefix": "order-"}}]
```

Corrisponde a uno dei seguenti attributi:

```
"event": {"Type": "String", "Value": "data-entry"}
```

```
"event": {"Type": "String", "Value": "order_number"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
   "event": "data-entry"
}
```

```
{
   "event": "order_number"
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"event": {"Type": "String", "Value": "order-cancelled"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
   "event": "order-cancelled"
}
```

quals-ignore-case Corrispondenza E

Quando una proprietà della policy include la parola chiave `equals-ignore-case`, verrà effettuata una corrispondenza che ignora le maiuscole/minuscole in qualsiasi valore di attributo dei messaggi o di proprietà del corpo.

Esaminiamo la seguente proprietà della policy:

```
"customer_interests": [{"equals-ignore-case": "tennis"}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"customer_interests": {"Type": "String", "Value": "TENNIS"}
```

```
"customer_interests": {"Type": "String", "Value": "Tennis"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
    "customer_interests": "TENNIS"
}
```

```
{
    "customer_interests": "teNnis"
{
```

Corrispondenza indirizzo IP

Puoi utilizzare l'operatore `cidr` per verificare se un messaggio in arrivo proviene da un indirizzo IP o da una subnet specifica.

Esaminiamo la seguente proprietà della policy:

```
"source_ip":[{"cidr": "10.0.0.0/24"}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"source_ip": {"Type": "String", "Value": "10.0.0.0"}
```

```
"source_ip": {"Type": "String", "Value": "10.0.0.255"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
   "source_ip": "10.0.0.0"
}
```

```
{
   "source_ip": "10.0.0.255"
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"source_ip": {"Type": "String", "Value": "10.1.1.0"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
   "source_ip": "10.1.1.0"
}
```

Corrispondenza in base al prefisso

Quando una proprietà della policy include la parola chiave `prefix`, corrisponde a qualsiasi valore di proprietà del messaggio che inizi con i caratteri specificati.

Esaminiamo la seguente proprietà della policy:

```
"customer_interests": [{"prefix": "bas"}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "basketball"
```

```
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
   "customer_interests": "rugby"
}
```

Corrispondenza dei suffissi

Quando una proprietà della policy include la parola chiave `suffix`, mette in corrispondenza qualsiasi valore di attributo dei messaggio o di proprietà del corpo che inizi con i caratteri specificati.

Esaminiamo la seguente proprietà della policy:

```
"customer_interests": [{"suffix": "ball"}]
```

Corrisponde a uno dei seguenti attributi di messaggio:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

Corrisponde anche a uno dei seguenti corpi dei messaggi:

```
{
    "customer_interests": "baseball"
}
```

```
{
    "customer_interests": "basketball"
}
```

Tuttavia, non corrisponde al seguente attributo di messaggio:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
    "customer_interests": "rugby"
}
```

Applicazione di una policy di filtro per le sottoscrizioni

Puoi applicare una policy di filtro a una sottoscrizione Amazon SNS utilizzando la console di Amazon SNS. Oppure, per applicare le policy a livello di codice, puoi utilizzare l'API Amazon SNS, AWS Command Line Interface il AWS CLI() o AWS qualsiasi SDK che supporti Amazon SNS. Puoi anche usare. AWS CloudFormation

> **Important**
>
> AWS servizi come IAM e Amazon SNS utilizzano un modello di calcolo distribuito chiamato eventuale consistenza. Le aggiunte o le modifiche a una policy di filtro sottoscrizione richiedono fino a 15 minuti per essere pienamente effettive.

AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, scegli Sottoscrizioni.
3. Seleziona una sottoscrizione e quindi scegli Edit (Modifica).
4. Nella pagina Edit (Modifica), espandi la sezione Policy di filtro per sottoscrizione.
5. Scegli tra il filtro basato sugli attributi o sul payload.
6. Nel campo Editor JSON, specifica il corpo JSON della policy di filtro.
7. Seleziona Save changes (Salva modifiche).

   Amazon SNS applica la policy di filtro alla sottoscrizione.

AWS CLI

Per applicare una politica di filtro con AWS Command Line Interface (AWS CLI), usa il `set-subscription-attributes`comando, come mostrato nell'esempio seguente. Per l'opzione `--attribute-name` specifica `FilterPolicy`. Per `--attribute-value`, specifica la policy JSON.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

Per specificare codice JSON valido per la policy, racchiudi i nomi e i valori degli attributi tra virgolette doppie. Devi inoltre racchiudere l'intero argomento della policy tra virgolette. Per evitare l'escape delle virgolette, puoi utilizzare virgolette singole per racchiudere la policy e virgolette doppie per racchiudere i nomi e i valori JSON, come mostrato nell'esempio qui sopra.

Se desideri passare dal filtraggio dei messaggi basato sugli attributi (impostazione predefinita) a quello basato sul payload, puoi utilizzare anche il comando. set-subscription-attributes Per l'opzione `--attribute-name` specifica `FilterPolicyScope`. Per `--attribute-value`, specificare `MessageBody`.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-
name FilterPolicyScope --attribute-value MessageBody
```

Per verificare l'applicazione della policy di filtro, usa il comando `get-subscription-attributes`. Gli attributi nell'output su terminale devono mostrare la policy di filtro per la chiave `FilterPolicy`, come mostrato nell'esempio seguente:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
    "Attributes": {
        "Endpoint": "endpoint . . .",
        "Protocol": "https",
        "RawMessageDelivery": "false",
        "EffectiveDeliveryPolicy": "delivery policy . . .",
        "ConfirmationWasAuthenticated": "true",
        "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed
\"]}",
        "FilterPolicyScope": "MessageAttributes",
        "Owner": "111122223333",
        "SubscriptionArn": "arn:aws:sns: . . .",
```

```
        "TopicArn": "arn:aws:sns: . . ."
    }
}
```

AWS SDK

I seguenti esempi di codice mostrano come utilizzare`SetSubscriptionAttributes`.

> **Important**
>
> Se si utilizza l'esempio SDK for Java 2.x, la classe `SNSMessageFilterPolicy` non è pronta all'uso. Per istruzioni su come installare questa classe, vedete l'esempio tratto dal GitHub sito Web.

CLI

AWS CLI

Impostazione degli attributi della sottoscrizione

Nell'esempio `set-subscription-attributes` seguente viene impostato l'attributo `RawMessageDelivery` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene impostato un attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene rimosso l'attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consultate SetSubscriptionAttributes AWS CLICommand Reference.

Java

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.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, SetSubscriptionAttributesconsulta AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
```

```
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Per i dettagli sull'API, consulta SetSubscriptionAttributes AWSSDK for Python (Boto3) API Reference.

API Amazon SNS

Per applicare una policy di filtro con l'API di Amazon SNS, effettua una richiesta all'operazione `SetSubscriptionAttributes`. Imposta il parametro `AttributeName` su `FilterPolicy` e il parametro `AttributeValue` sul JSON della policy di filtro.

Se desideri passare dal filtro dei messaggi basato sugli attributi (opzione predefinita) a quello basato sul payload, puoi utilizzare anche l'azione `SetSubscriptionAttributes` . Imposta il parametro `AttributeName` su `FilterPolicyScope` e il parametro `AttributeValue` su `MessageBody`.

AWS CloudFormation

Per applicare una politica di filtro utilizzando AWS CloudFormation, utilizzate un modello JSON o YAML per creare uno stack. AWS CloudFormation Per ulteriori informazioni, consultate la `FilterPolicy`proprietà della `AWS::SNS::Subscription` risorsa nella Guida per l'AWS CloudFormation utente e il modello di esempio. AWS CloudFormation

1. Accedere alla console AWS CloudFormation.
2. Scegli Create Stack (Crea stack).
3. Nella pagina Select Template (Scegli modello), scegli Upload a template to Amazon S3 (Carica un modello in Amazon S3), scegli il file, quindi scegli Next (Avanti).
4. Nella pagina Specify Details (Specifica dettagli), procedi come segue:
   a. Per Nome stack, digita `MyFilterPolicyStack`.
   b. Per myHttpEndpoint, digita l'endpoint HTTP a cui iscriverti al tuo argomento.

   > **Tip**
   >
   > Se non disponi di un endpoint HTTP, creane uno.

5. Nella pagina Opzioni, scegli Next (Avanti).
6. Nella pagina Revisione scegli Create (Crea).

Rimozione di una policy di filtro per le sottoscrizioni

Per interrompere il filtraggio dei messaggi inviati a una sottoscrizione, rimuovi la policy di filtro delle sottoscrizioni sovrascrivendola con un corpo JSON vuoto. Dopo aver rimosso la policy, la sottoscrizione accetta ogni messaggio pubblicato.

AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, scegli Sottoscrizioni.
3. Seleziona una sottoscrizione e quindi scegli Edit (Modifica).
4. Su Modificare ***EXAMPLE1-23bc-4567-d890-ef12g3hij456*** pagina, espandere la sezione Policy di filtro per sottoscrizione.

5. Nel campo JSON editor (Editor JSON) specificare un corpo JSON vuoto per la policy di filtro. `{}`
6. Seleziona Save changes (Salva modifiche).

   Amazon SNS applica la policy di filtro alla sottoscrizione.

AWS CLI

Per rimuovere una policy di filtro con la AWS CLI, utilizza il comando `set-subscription-attributes` e fornisci un corpo JSON vuoto per l'argomento `--attribute-value`:

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

API Amazon SNS

Per rimuovere una policy di filtro con l'API di Amazon SNS, effettua una richiesta all'operazione `SetSubscriptionAttributes`. Imposta il parametro `AttributeName` su `FilterPolicy`e fornisci un corpo JSON vuoto per il parametro `AttributeValue`.

Protezione dei dati dei messaggi

Argomenti

Cos'è la protezione dei dati dei messaggi?

La protezione dei dati dei messaggi garantisce la protezione dei dati pubblicati negli argomenti Amazon SNS tramite policy di protezione dei dati che prevedono la verifica, il mascheramento, l'oscuramento o il blocco delle informazioni sensibili in transito tra applicazioni o servizi AWS.

La protezione dei dati dei messaggi esegue la scansione dei dati in movimento alla ricerca di informazioni di identificazione personale (PII) e di dati sanitari protetti (PHI) tramite identificatori di dati. Puoi scegliere di utilizzare identificatori di dati predefiniti (o gestiti da Amazon SNS), ad esempio nomi, indirizzi, numeri di carte di credito e codici di farmaci soggetti a prescrizione medica, oppure puoi creare identificatori di dati personalizzati, specifici per il tuo caso d'uso aziendale. Utilizzando le informazioni scansionate, la protezione dei dati dei messaggi fornisce registri di controllo dettagliati e consente di eseguire operazioni specifiche volte a proteggere tali dati.

La protezione dei dati dei messaggi supporta le seguenti operazioni per proteggere le informazioni sensibili dei clienti:

- Audit (Verifica): controlla fino al 99% dei dati pubblicati in un argomento Amazon SNS. Puoi quindi scegliere di inviare i risultati ad Amazon CloudWatch, Amazon S3 o Amazon Data Firehose.
- Anonimizzazione: maschera o oscura i dati sensibili senza interrompere la pubblicazione o la distribuzione dei messaggi.
- Deny (Rifiuto): blocca la trasmissione dei dati tra applicazioni e risorse AWS se nel payload sono presenti dati sensibili.

> **Note**
>
> Amazon SNS supporta la protezione dei dati dei messaggi solo per gli argomenti standard di Amazon SNS.

Perché utilizzare la protezione dei dati dei messaggi?

L'introduzione della protezione dei dati dei messaggi nei programmi di governance, gestione del rischio e conformità ti consente di implementare policy di protezione che ti aiutano a individuare e prevenire la fuga di dati. Ciò fornisce ai team strumenti che possono aiutare a ridurre i rischi finanziari, legali e normativi rispettando le normative sulla privacy quali, ad esempio, HIPAA, GDPR, PCI e FedRAMP. Inoltre, libera gli sviluppatori dal sovraccarico operativo associato alla creazione e alla gestione di strumenti specifici per la protezione dei dati sensibili.

Ad esempio, puoi utilizzare la protezione dei dati dei messaggi per creare una policy di verifica per determinare se uno dei tuoi sistemi invia o riceve inavvertitamente dati sensibili. Se i risultati della verifica mostrano che i sistemi inviano i dati relativi alle carte di credito a sistemi che non li richiedono, puoi utilizzare una policy di blocco per impedire che ciò accada.

> **Note**
>
> Amazon SNS supporta la protezione dei dati dei messaggi solo per gli argomenti standard di Amazon SNS.

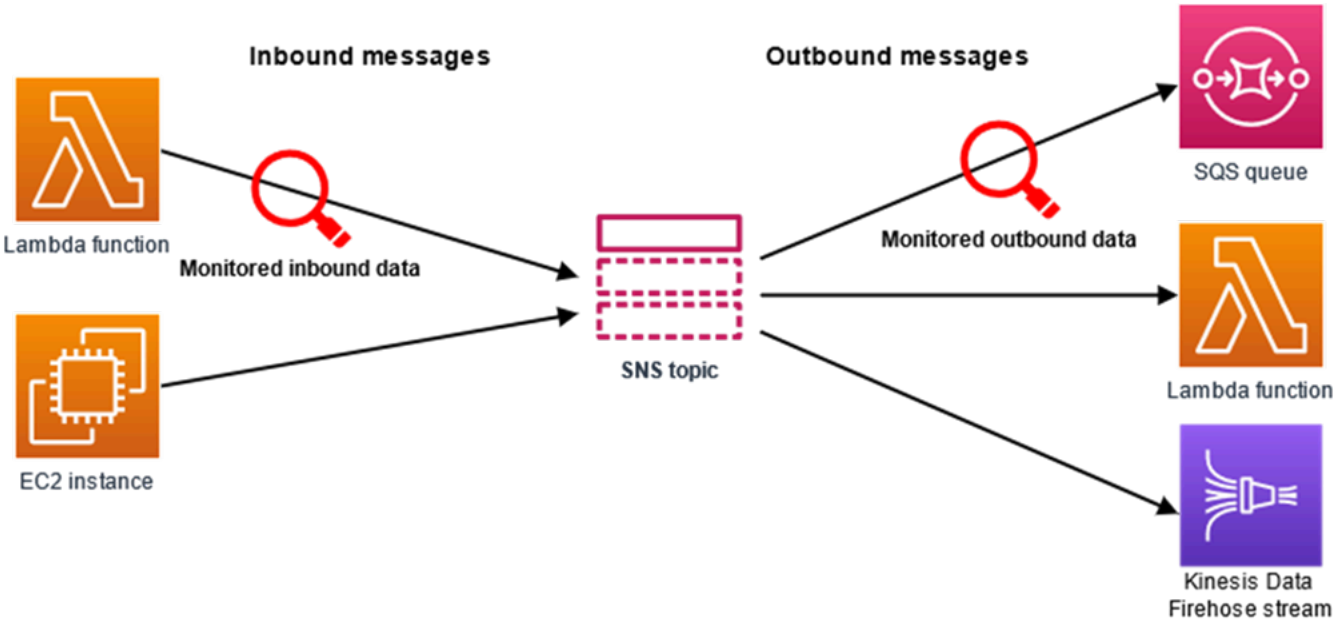
Informazioni sulle policy di protezione dei dati

Argomenti

- Eliminazione di policy di protezione dei dati in Amazon SNS

Cosa sono le policy di protezione dei dati?

Amazon SNS utilizza le policy di protezione dei dati per selezionare i dati sensibili da sottoporre a scansione e le operazioni che desideri intraprendere per proteggere tali dati dall'interscambio a livello di argomenti Amazon SNS. Per selezionare i dati sensibili di interesse, utilizza gli identificatori di dati. La protezione dei dati dei messaggi di Amazon SNS rileva quindi la presenza di dati sensibili utilizzando il machine learning e i criteri di ricerca. Per agire sugli identificatori di dati trovati, è possibile definire un'operazione di verifica, deidentificazione o rifiuto. Queste operazioni consentono di registrare i dati sensibili trovati (o non trovati), di mascherare o oscurare i dati sensibili o di rifiutare il recapito dei messaggi.

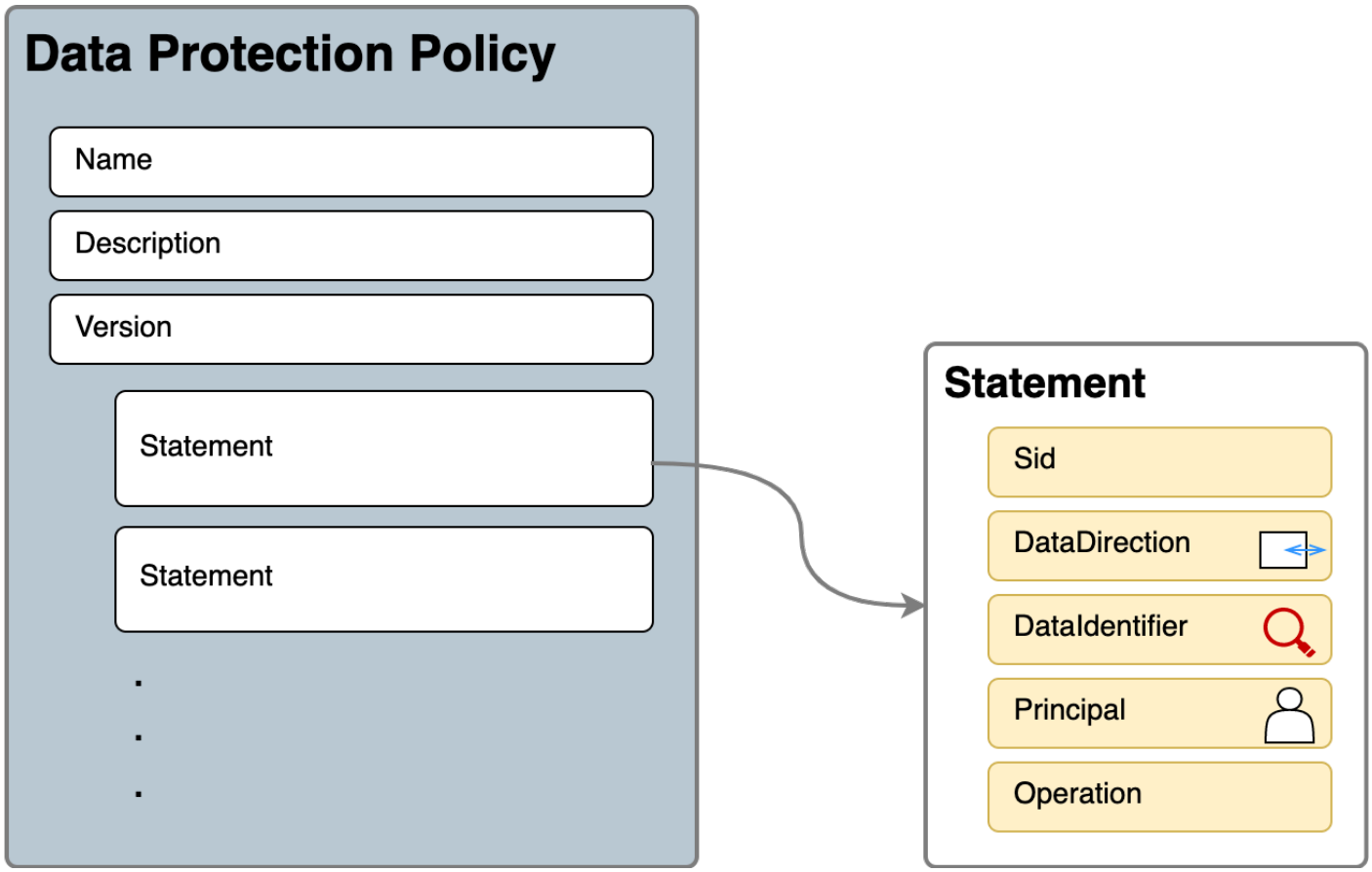


Come è strutturata una policy di protezione dei dati?

Come illustrato nella figura riportata di seguito, un documento relativo alla policy di protezione dei dati include questi elementi:

- Informazioni opzionali sulla policy nella parte superiore del documento
- Una o più istruzioni singole

Ogni istruzione include informazioni su una singola autorizzazione.



È possibile definire solo una policy di protezione dei dati per argomento Amazon SNS. La policy di protezione dei dati può includere una o più dichiarazioni di rifiuto o deidentificazione, ma solo una dichiarazione di verifica.

Proprietà JSON per la policy di protezione dei dati

Una policy di protezione dei dati richiede le seguenti informazioni di base ai fini dell'identificazione:

- Name (Nome): nome della policy.
- Description (Descrizione): (facoltativo) la descrizione della policy.
- Version (Versione): la versione del linguaggio della policy. La versione corrente è 2021-06-01.
- Statement (Dichiarazione): l'elenco di dichiarazioni che specificano le operazioni della policy di protezione dei dati.

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
```

```
  "Version": "2021-06-01",
  "Statement": [
        ...
  ]
}
```

Proprietà JSON per una dichiarazione di policy

Una dichiarazione di policy definisce il contesto di rilevamento per l'operazione di protezione dei dati.

- Sid: (facoltativo) l'identificatore della dichiarazione.
- DataDirection (Direzione dei dati): Inbound (In entrata), ad esempio per le richieste API di pubblicazione, o Outbound (In uscita), per la consegna delle notifiche, in relazione all'argomento Amazon SNS.
- DataIdentifier (Identificatore di dati): i dati sensibili che l'argomento Amazon SNS deve esaminare. Ad esempio, nome, indirizzo o numero di telefono.
- Principale: il principale IAM che ha pubblicato nell'argomento o il principale IAM che ha effettuato la sottoscrizione all'argomento.
- Operation (Operazione): l'operazione successiva, ovvero Audit (Verifica), De-identify (Deidentificazione) (mascheramento o oscuramento) o Deny (Rifiuto) (blocco), eseguita dall'argomento Amazon SNS se è stata rilevata la presenza di dati sensibili.

```
{
    "Sid": "basicPII-inbound-protection",
    "DataDirection": "Inbound",
    "Principal": ["*"],
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Name",
        "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
    ],
    "Operation": {
        ...
    }
}
```

Proprietà JSON per un'operazione della dichiarazione di policy

Una dichiarazione di policy definisce una delle seguenti operazioni di protezione dei dati.

- Audit (Verifica): genera parametri e registri di risultati della ricerca senza interrompere la pubblicazione o il recapito dei messaggi.
- De-identify (Deidentificazione): maschera o oscura i dati sensibili senza interrompere la pubblicazione dei messaggi.
- Deny (Rifiuto): blocca la richiesta di pubblicazione di Amazon SNS o non finalizza il recapito dei messaggi.

Come faccio a determinare i principali IAM per la policy di protezione dei dati?

La protezione dei dati dei messaggi utilizza due principali IAM che interagiscono con Amazon SNS.

1. Principale dell'API Publish (in entrata): il principale IAM autenticato che chiama l'API Amazon SNS `Publish`.
2. Subscription Principal (Principale di sottoscrizione) (in uscita): il principale IAM autenticato che ha chiamato l'API `Subscribe` durante la creazione della sottoscrizione.

`SubscriptionPrincipal` è una proprietà di sottoscrizione Amazon SNS disponibile pubblicamente e che può essere recuperata dall'API `GetSubscriptionAttributes`.

```
{
  "Attributes": {
    "SubscriptionPrincipal": "arn:aws:iam::123456789012:user/NoNameAccess",
    "Owner": "123412341234",
    "RawMessageDelivery": "true",
    "TopicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
    "Endpoint": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
    "Protocol": "sqs",
    "PendingConfirmation": "false",
    "ConfirmationWasAuthenticated": "true",
    "SubscriptionArn": "arn:aws:sns:us-east-1:123412341234:PII-data-
topic:5d8634ef-67ef-49eb-a824-4042b28d6f55"
  }
}
```

Operazioni delle policy di protezione dei dati

Di seguito sono riportati alcuni esempi di policy di protezione dei dati che puoi utilizzare per controllare e rifiutare i dati sensibili. Per un tutorial completo che include un'applicazione di esempio, consulta il post di blog relativo alla presentazione della protezione dei dati dei messaggi per Amazon SNS.

Argomenti

Operazione di verifica

L'operazione Audit (Verifica) campiona l'argomento dei messaggi in entrata e registra i risultati dei dati sensibili in una destinazione AWS. La frequenza di campionamento può essere un numero intero compreso tra 0 e 99. Questa operazione richiede uno dei seguenti tipi di destinazione della registrazione:

1. FindingsDestination— La destinazione di registrazione quando l'argomento Amazon SNS trova dati sensibili nel payload.
2. NoFindingsDestination— La destinazione di registrazione quando l'argomento Amazon SNS non trova dati sensibili nel payload.

Puoi utilizzare i seguenti Servizi AWS in ciascuno dei tipi di destinazione della registrazione riportati di seguito:

- Amazon CloudWatch Logs (opzionale): `LogGroup` deve essere nell'area tematica e il nome deve iniziare con /aws/vendedlogs/.
- Amazon Data Firehose (opzionale): `DeliveryStream` deve trovarsi nell'area tematica e avere Direct PUT come fonte del flusso di distribuzione. Per ulteriori dettagli, consulta Source, Destination and Name nella Amazon Data Firehose Developer Guide.
- Amazon S3 (facoltativo): il nome di un bucket Amazon S3. L'utilizzo del bucket Amazon S3 con la crittografia SSE-KMS abilitata richiede azioni aggiuntive.

```
{
  "Operation": {
    "Audit": {
      "SampleRate": "99",
      "FindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
      },
      "NoFindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
      }
    }
  }
}
```

Autorizzazioni richieste per specificare le destinazioni della registrazione

Quando specifichi le destinazioni di registrazione nella policy di protezione dei dati, devi aggiungere le seguenti autorizzazioni alla policy di identità IAM del principale IAM che chiama l'API `PutDataProtectionPolicy` di Amazon SNS o l'API `CreateTopic` con il parametro `--data-protection-policy`.

Destinazione della verifica	Autorizzazione IAM
Predefinita	logs:CreateLogDelivery logs:GetLogDelivery

Destinazione della verifica	Autorizzazione IAM
	logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy L'utilizzo del bucket Amazon S3 con la crittografia SSE-KMS abilitata richiede azioni aggiuntive.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs:DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ]
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:region:account-id:SampleLogGroupName:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}
```

Policy di chiave richiesta per l'uso con SSE-KMS

Se utilizzi un bucket Amazon S3 come una destinazione di log, puoi proteggere i dati nel bucket abilitando la crittografia lato server con chiavi gestite da Amazon S3 (SSE-S3) o la crittografia lato server con AWS KMS keys (SSE-KMS). Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server nella Guida per l'utente di Amazon S3.

Se si sceglie SSE-S3, non è richiesta alcuna configurazione aggiuntiva. Amazon S3 gestisce la chiave di crittografia.

Se si sceglie SSE-KMS, è necessario utilizzare una chiave gestita dal cliente. Devi aggiornare la policy delle chiavi per la chiave gestita dal cliente in modo che l'account di consegna del log possa scrivere nel bucket S3. Per ulteriori informazioni sulla politica delle chiavi richiesta per l'uso con SSE-KMS, consulta la crittografia lato server con bucket Amazon S3 nella Amazon Logs User Guide. CloudWatch

Esempi di registro della destinazione della verifica

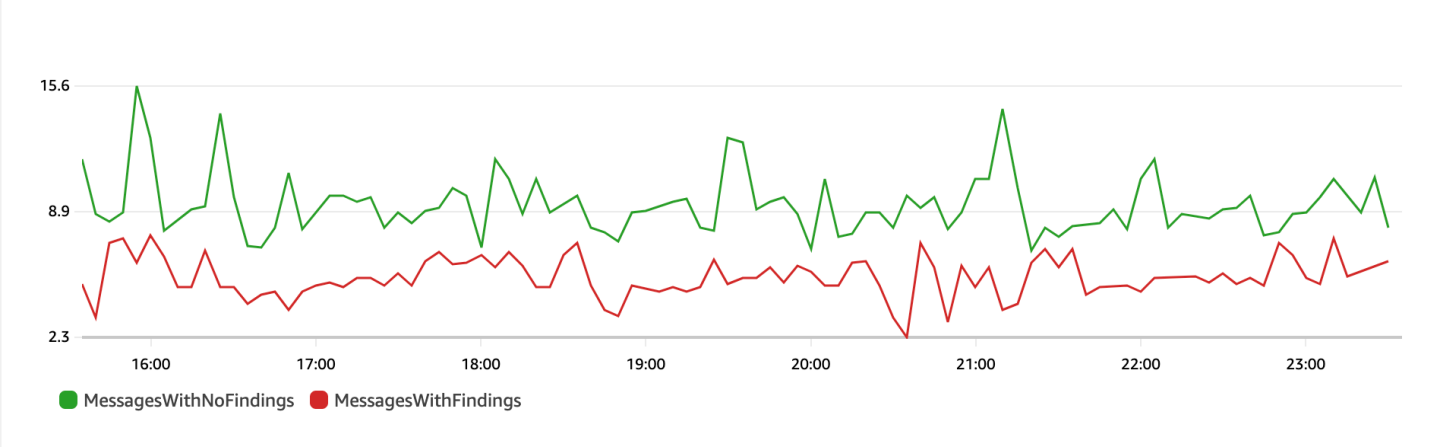
Nell'esempio seguente, `callerPrincipal` viene utilizzato per identificare l'origine del contenuto sensibile e `messageID` viene utilizzato come un riferimento per eseguire il controllo in base alla risposta dell'API `Publish`.

```
{
  "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
  "auditTimestamp": "2022-05-12T2:10:44Z",
  "callerPrincipal": "arn:aws:iam::123412341234:role/Publisher",
  "resourceArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
  "dataIdentifiers": [
    {
      "name": "Name",
      "count": 1,
      "detections": [
        {
          "start": 1,
          "end": 2
        }
      ]
    },
    {
      "name": "PhoneNumber",
      "count": 2,
      "detections": [
        {
          "start": 3,
          "end": 4
        },
        {
          "start": 5,
          "end": 6
        }
      ]
    }
  ]
```

```
}
```

Parametri dell'operazione di verifica

Quando un'operazione di controllo ha specificato la proprietà `FindingsDestination` o la `NoFindingsDestination` proprietà, i proprietari dell'argomento ricevono anche i parametri e. CloudWatch `MessagesWithFindings` `MessagesWithNoFindings`



Operazione di deidentificazione

L'operazione Anonimizza maschera o oscura i dati sensibili dai messaggi pubblicati o consegnati. Questa operazione è disponibile per i messaggi in entrata e richiede uno dei seguenti tipi di configurazione:

- MaskConfig— Maschera utilizzando un carattere supportato dalla tabella seguente. Ad esempio, ssn: `123-45-6789` diventa ssn: `###########`.

```
{
"Operation": {
    "Deidentify": {
        "MaskConfig": {
            "MaskWithCharacter": "#"
          }
    }
}
```

Carattere di mascheramento supportato	Nome
*	Asterisco

Carattere di mascheramento supportato	Nome
A-Z, a-z e 0-9	Carattere alfanumerico
	Spazio
!	Punto esclamativo
$	Simbolo del dollaro
%	Segno percentuale
&	E commerciale
()	Parentesi
+	Segno più
,	Virgola
-	Trattino
.	Periodo
/\	Barra, barra rovesciata
#	Segno numerico
:	Due punti
;	Punto e virgola
=, <>	Uguale a, minore di o maggiore di
@	Chiocciola
[]	Parentesi
^	Simbolo dell'accento circonflesso
_	Carattere di sottolineatura

Carattere di mascheramento supportato	Nome
`	Accento grave
|	Barra verticale
~	Simbolo della tilde

- RedactConfig— Redigi rimuovendo completamente i dati. Ad esempio, ssn: `123-45-6789` diventa ssn: .

```
{
"Operation": {
    "Deidentify": {
        "RedactConfig": {}
    }
}
```

In un messaggio in entrata, i dati sensibili vengono deidentificati dopo l'operazione di verifica e il chiamante dell'API `SNS:Publish` riceve il seguente errore di parametro non valido quando l'intero messaggio è sensibile.

```
Error code: AuthorizationError ...
```

Operazione di rifiuto

L'operazione Deny (Rifiuto) interrompe l'esecuzione della richiesta API `Publish` o la consegna del messaggio se il messaggio contiene dati sensibili. L'oggetto dell'operazione Deny (Rifiuto) è vuoto in quanto non richiede una configurazione aggiuntiva.

```
"Operation": {
    "Deny": {}
}
```

In un messaggio in entrata, il chiamante dell'API `SNS:Publish` riceve un errore di autorizzazione.

```
Error code: AuthorizationError ...
```

In un messaggio in uscita, l'argomento Amazon SNS non consegna il messaggio alla sottoscrizione. Per tenere traccia di operazioni di consegna non autorizzate, abilita l'opzione Delivery status logging

(Registrazione dello stato della consegna) dell'argomento. Nell'esempio seguente viene mostrato un esempio di registro dello stato della consegna:

```
{
    "notification": {
        "messageMD5Sum": "29638742ffb68b32cf56f42a79bcf16b",
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "topicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
        "timestamp": "2022-05-12T2:12:44Z"
    },
    "delivery": {
        "deliveryId": "98236591c-56aa-51ee-a5ed-0c7d43493170",
        "destination": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
        "providerResponse": "The topic's data protection policy prohibits this message
 from being delivered to <subscription-arn>",
        "dwellTimeMs":20,
        "attempts":1,
        "statusCode": 403
    },
    "status": "FAILURE"
}
```

Esempi di policy di protezione dei dati

Gli esempi riportati di seguito sono policy di protezione dei dati che puoi utilizzare per controllare e rifiutare i dati sensibili. Per un tutorial completo che include un'applicazione di esempio, consulta il post di blog relativo alla presentazione della protezione dei dati dei messaggi per Amazon SNS.

Argomenti

- Esempio di policy per la verifica
- Policy di esempio con dichiarazione di deidentificazione tramite mascheramento in entrata
- Policy di esempio con dichiarazione di deidentificazione tramite oscuramento in entrata
- Policy di esempio con dichiarazione di anonimizzazione tramite mascheramento in uscita
- Policy di esempio con dichiarazione di anonimizzazione tramite oscuramento in uscita
- Esempio di policy con una dichiarazione di rifiuto in entrata
- Esempio di policy con una dichiarazione di rifiuto in uscita

Esempio di policy per la verifica

Le politiche di controllo consentono di controllare fino al 99% dei messaggi in entrata e di inviare i risultati ad Amazon CloudWatch, Amazon Data Firehose e Amazon S3.

Ad esempio, puoi creare una policy di verifica per valutare se uno dei tuoi sistemi invia o riceve inavvertitamente dati sensibili. Se i risultati della verifica mostrano che i sistemi inviano i dati relativi alle carte di credito a sistemi che non li richiedono, puoi implementare una policy di blocco per impedire che ciò accada.

L'esempio seguente verifica il 99% dei messaggi che attraversano l'argomento cercando i numeri delle carte di credito e inviando i risultati a CloudWatch Logs, Firehose e Amazon S3.

Policy di protezione dei dati:

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": ["*"],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Audit": {
          "SampleRate": "99",
          "FindingsDestination": {
            "CloudWatchLogs": {
              "LogGroup": "<example log name>"
            },
            "Firehose": {
              "DeliveryStream": "<example stream name>"
            },
            "S3": {
              "Bucket": "<example bucket name>"
            }
          }
        }
      }
```

```
    }
  ]
}
```

Esempio di formato dei risultati della verifica:

```
{
    "messageId": "...",
    "callerPrincipal": "arn:aws:sts::123456789012:assumed-role/ExampleRole",
    "resourceArn": "arn:aws:sns:us-east-1:123456789012:ExampleArn",
    "dataIdentifiers": [
        {
            "name": "CreditCardNumber",
            "count": 1,
            "detections": [
                { "start": 1, "end": 2 }
            ]
        }
    ],
    "timestamp": "2021-04-20T00:33:40.241Z"
}
```

Policy di esempio con dichiarazione di deidentificazione tramite mascheramento in entrata

L'esempio seguente impedisce a un utente di pubblicare un messaggio in un argomento con `CreditCardNumber` mascherando i dati sensibili contenuti nel messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
```

```
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}
```

Esempio di risultati di deidentificazione tramite mascheramento in entrata:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ################
```

Policy di esempio con dichiarazione di deidentificazione tramite oscuramento in entrata

L'esempio seguente impedisce a un utente di pubblicare un messaggio in un argomento con `CreditCardNumber` oscurando i dati sensibili dal contenuto del messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
```

```
    }
  ]
}
```

Esempio di risultati di deidentificazione in entrata:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Policy di esempio con dichiarazione di anonimizzazione tramite mascheramento in uscita

L'esempio seguente impedisce a un utente di ricevere un messaggio con `CreditCardNumber` mascherando i dati sensibili contenuti nel messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "-"
          }
        }
      }
    }
  ]
}
```

Esempio di risultati di anonimizzazione tramite mascheramento in uscita:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ----------------
```

Policy di esempio con dichiarazione di anonimizzazione tramite oscuramento in uscita

L'esempio seguente impedisce a un utente di ricevere un messaggio con `CreditCardNumber` oscurando i dati sensibili contenuti nel messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

Esempio di risultati di anonimizzazione in uscita:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Esempio di policy con una dichiarazione di rifiuto in entrata

L'esempio seguente impedisce a un utente di pubblicare in un argomento un messaggio contenente `CreditCardNumber`. I payload negati nella risposta dell'API hanno il codice di stato "403 AuthorizationError".

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

Esempio di policy con una dichiarazione di rifiuto in uscita

L'esempio seguente blocca la ricezione di messaggi contenenti `CreditCardNumber` da parte di un account AWS.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
```

```
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

Esempio di risultati di rifiuto in uscita, registrato in Amazon: CloudWatch

```
{
  "notification": {
    "messageMD5Sum": "2e8f58ff2eeed723b56b15493fbfb5a5",
    "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
    "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
    "timestamp": "2022-09-08 15:40:57.144"
  },
  "delivery": {
    "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
    "destination": "arn:aws:sqs:us-east-2:664555388960:test",
    "providerResponse": "The topic's data protection policy prohibits this message from
 being delivered to <subscription arn>",
    "dwellTimeMs": 22,
    "attempts": 1,
    "statusCode": 403
  },
  "status": "FAILURE"
}
```

Creazione di policy di protezione dei dati

Le policy di protezione dei dati semplificano la protezione dei dati pubblicati negli argomenti Amazon SNS tramite la verifica, la deidentificazione (mascheramento o oscuramento) e il rifiuto (blocco) delle informazioni sensibili in transito tra applicazioni o Servizi AWS. Puoi utilizzare l'API AWS, la AWS CLI, AWS CloudFormation o la AWS Management Console per creare policy di protezione dei dati in Amazon SNS. È possibile definire solo una policy per argomento Amazon SNS. Ciascuna policy di protezione dei dati può avere una o più dichiarazioni di deidentificazione e rifiuto, ma solo una dichiarazione di verifica.

Argomenti

- Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (API)
- Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (CLI)
- Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (CloudFormation)
- Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (console)
- Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (SDK)

Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (API)

Il numero e la dimensione delle risorse IAM in un account AWS sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di policy di protezione dei dati (API AWS)

Puoi creare una policy di protezione dei dati di Amazon SNS utilizzando l'API AWS.

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (API AWS)

Utilizza la proprietà `DataProtectionPolicy` di un argomento standard di Amazon SNS:

- `CreateTopic`

Per recuperare o creare una policy di protezione dei dati per un argomento Amazon SNS esistente (API AWS)

Chiamare una delle seguenti operazioni:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (CLI)

Il numero e la dimensione delle risorse IAM in un account AWS sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di policy di protezione dei dati (AWS CLI)

Puoi creare una policy di protezione dei dati di Amazon SNS utilizzando la AWS Command Line Interface.

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (AWS CLI)

Usa questa opzione per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

- create-topic

Per creare o recuperare una policy di protezione dei dati per un argomento Amazon SNS esistente (AWS CLI)

Chiamare una delle seguenti operazioni:

- get-data-protection-policy
- put-data-protection-policy

Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (CloudFormation)

Il numero e la dimensione delle risorse IAM in un account AWS sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di policy di protezione dei dati (CloudFormation)

Puoi creare una policy di protezione dei dati di Amazon SNS utilizzando AWS CloudFormation.

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (CloudFormation)

Usa questa opzione per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

- AWS::SNS::Topic

Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (console)

Il numero e la dimensione delle risorse IAM in un account AWS sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (console)

Usa questa opzione per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS.

1. Accedi alla console Amazon SNS.
2. Scegli un argomento o creane uno nuovo. Per ulteriori dettagli sulla creazione di argomenti, consulta Creare un argomento Amazon SNS.
3. Nella pagina Create topic (Crea argomento), nella sezione Details (Dettagli) scegli Standard.
   a. Immetti un nome per l'argomento.
   b. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.
4. Espandi Data protection policy (Policy di protezione dei dati).
5. Scegli una modalità in Configuration mode (Modalità di configurazione):
   - Basic (Base): consente di definire una policy di protezione dei dati utilizzando un semplice menu.
   - Advanced (Avanzato): consente di definire una policy di protezione dei dati personalizzata utilizzando JSON.
6. (Facoltativo) Per creare un identificatore di dati personalizzato, espandi la sezione Configurazione dell'identificatore di dati personalizzato, procedi come segue:
   a. Immetti un nome univoco per l'identificatore di dati personalizzato. I nomi di identificatori di dati personalizzati supportano i caratteri alfanumerici, il carattere di sottolineatura (_) e il trattino (-). Sono supportati fino a 128 caratteri. Questo nome non può condividere lo stesso nome di un identificatore di dati gestito. Per un elenco completo delle limitazioni relative agli identificatori di dati personalizzati, consulta Vincoli degli identificatori di dati personalizzati.
   b. Immettete un'espressione regolare (RegEx) per l'identificatore di dati personalizzato. RegExsupporta caratteri alfanumerici, caratteri RegEx riservati e simboli. RegEx ha una lunghezza massima di 200 caratteri. Se RegEx è troppo complicato, Amazon SNS fallirà la chiamata API. Per un elenco completo delle RegEx limitazioni, consultaVincoli degli identificatori di dati personalizzati.

c. (Facoltativo) Scegli Aggiungi identificatore di dati personalizzato per aggiungere altri identificatori di dati, se necessario. Ciascuna policy di protezione dei dati attualmente supporta un massimo di 10 identificatori di dati personalizzati.

7. Scegli le dichiarazioni che desideri aggiungere alla policy di protezione dei dati. È possibile aggiungere i tipi di dichiarazione verifica, deidentificazione (mascheramento o oscuramento) e rifiuto (blocco) alla stessa policy di protezione dei dati.

   a. Add audit statement (Aggiungi dichiarazione di verifica): configura quali dati sensibili controllare, quale percentuale di messaggi desideri controllare per tali dati e dove inviare i registri di verifica.

      > **Note**
      >
      > È consentita una sola dichiarazione di verifica per policy o argomento di protezione dei dati.

      i. In Data identifiers (Identificatori di dati) seleziona gli identificatori di dati per definire i dati sensibili che si desidera verificare.

      ii. In Audit sample rate (Frequenza di campionamento della verifica), inserisci la percentuale di messaggi per i quali verificare la presenza di informazioni sensibili, fino a un massimo del 99%.

      iii. In Audit destination (Destinazione della verifica), seleziona i Servizi AWS per l'invio dei risultati della verifica e inserisci un nome per destinazione che ogni Servizio AWS deve utilizzare. Puoi scegliere tra i seguenti servizi Amazon Web Services:

      - Amazon CloudWatch — CloudWatch Logs è la soluzione di registrazione AWS standard. Utilizzando CloudWatch Logs, puoi eseguire analisi dei log utilizzando Logs Insights (vedi esempi qui) e creare metriche e allarmi. CloudWatch Logs è il luogo in cui molti servizi pubblicano i log, il che semplifica l'aggregazione di tutti i log utilizzando un'unica soluzione. Per informazioni su Amazon CloudWatch, consulta la Amazon CloudWatch User Guide.
      - Amazon Data Firehose — Firehose soddisfa le richieste di streaming in tempo reale su Splunk OpenSearch e Amazon Redshift per ulteriori analisi dei log. Per informazioni su Amazon Data Firehose, consulta la Amazon Data Firehose User Guide.

- Amazon Simple Storage Service: Amazon S3 è una soluzione economica per l'archiviazione dei registri. Potrebbe essere necessario conservare i registri per alcuni anni. In questo caso, puoi archiviare i registri in Amazon S3 per evitare di sostenere costi aggiuntivi. Per informazioni su Amazon Simple Storage Service, consulta la Guida per l'utente di Amazon Simple Storage Service.

b. Add a de-identify statement (Aggiungi una dichiarazione di deidentificazione): configura i dati sensibili che desideri deidentificare nel messaggio stabilendo se mascherarli o oscurarli e imposta gli account per bloccare la consegna di tali dati.

   i. In Data identifiers (Identificatori di dati) seleziona i dati sensibili che desideri deidentificare.

   ii. In Define this de-identify statement for (Definisci questa dichiarazione di deidentificazione per) seleziona gli account AWS o i responsabili IAM a cui si applica questa dichiarazione di deidentificazione. Puoi applicare la dichiarazione a tutti gli account AWS, ad account AWS specifici oppure a entità IAM (ruoli, utenti o root di account) che utilizzano ID account o ARN di entità IAM. Separa più ID o ARN utilizzando una virgola (,).

   Sono supportati i seguenti principali IAM:

   - IAM account principals (Principali degli account IAM): ad esempio, `arn:aws:iam::AWS-account-ID:root`.
   - IAM role principals (Principali dei ruoli IAM): ad esempio, `arn:aws:iam::AWS-account-ID:role/role-name`.
   - IAM user principals (Principali degli utenti IAM): ad esempio, `arn:aws:iam::AWS-account-ID:user/user-name`.

   iii. Per De-identify Option (Opzione di deidentificazione), seleziona il modo in cui desideri deidentificare i dati sensibili. Sono supportate le seguenti opzioni:

   - Redact (Oscuramento): rimuove completamente i dati. Ad esempio, l'e-mail: `classified@amazon.com` diventa l'e-mail: .
   - Mask (Mascheramento): sostituisce i dati con caratteri singoli. Ad esempio, l'e-mail: `classified@amazon.com` diventa l'e-mail: `**********`.

   iv. (Facoltativo) Continua ad aggiungere le dichiarazioni di deidentificazione, se necessario.

c. Add deny statement (Aggiungi dichiarazione di rifiuto): configura per quali dati sensibili impedire lo spostamento nell'argomento e per quali principali impedire la trasmissione di tali dati.

   i. Per Data direction (Direzione dei dati), scegli la direzione dei messaggi per la dichiarazione di rifiuto:

      - Inbound messages (Messaggi in entrata): applica questa dichiarazione di rifiuto ai messaggi inviati all'argomento.
      - Outbound messages (Messaggi in uscita): applica questa dichiarazione di rifiuto ai messaggi che l'argomento invia agli endpoint di sottoscrizione.

   ii. Scegli Data identifiers (Identificatori di dati) per definire i dati sensibili che desideri negare.

   iii. In IAM principals (Principali IAM) seleziona i principali IAM a cui applicare questa dichiarazione di rifiuto. Puoi applicare la dichiarazione a tutti gli account AWS, ad account Account AWS specifici oppure a entità IAM (ad esempio, ruoli, utenti o root di account) che utilizzano ID account o ARN di entità IAM. Separa più ID o ARN utilizzando una virgola (,). Sono supportati i seguenti principali IAM:

      - IAM account principals (Principali degli account IAM): ad esempio, `arn:aws:iam::AWS-account-ID:root`.
      - IAM role principals (Principali dei ruoli IAM): ad esempio, `arn:aws:iam::AWS-account-ID:role/role-name`.
      - IAM user principals (Principali degli utenti IAM): ad esempio, `arn:aws:iam::AWS-account-ID:user/user-name`.

   iv. (Facoltativo) Continua ad aggiungere le dichiarazioni di rifiuto, se necessario.

Creazione di policy di protezione dei dati per proteggere i dati dei messaggi (SDK)

Il numero e la dimensione delle risorse IAM in un account AWS sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di policy di protezione dei dati (SDK AWS)

Puoi creare una policy di protezione dei dati di Amazon SNS utilizzando l'SDK AWS.

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (SDK AWS)

Usa le seguenti opzioni per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

Java

```
/**
 * For information regarding CreateTopic see this documentation topic:
 *
 * https://docs.aws.amazon.com/code-samples/latest/catalog/javav2-sns-src-main-java-
com-example-sns-CreateTopic.java.html
 */

public static String createSNSTopicWithDataProtectionPolicy(SnsClient snsClient,
 String topicName, String dataProtectionPolicy) {

    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        CreateTopicResponse result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {CreateTopicCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { Name: "TOPIC_NAME", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };
```

```
const run = async () => {
  try {
    const data = await snsClient.send(new CreateTopicCommand(params));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
run();
```

Creare o recuperare una policy di protezione dei dati per un argomento Amazon SNS esistente (SDK AWS)

Usa le seguenti opzioni per creare o recuperare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

Java

```
public static void putDataProtectionPolicy(SnsClient snsClient, String topicName,
 String dataProtectionPolicy) {

    try {
        PutDataProtectionPolicyRequest request =
 PutDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        PutDataProtectionPolicyResponse result =
 snsClient.putDataProtectionPolicy(request);
        System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                + "\n\nTopic " + request.resourceArn()
                + " DataProtectionPolicy " + request.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static void getDataProtectionPolicy(SnsClient snsClient, String topicName) {

    try {
        GetDataProtectionPolicyRequest request =
 GetDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .build();

        GetDataProtectionPolicyResponse result =
 snsClient.getDataProtectionPolicy(request);

        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
        + "\n\nDataProtectionPolicy: \n\n" + result.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {PutDataProtectionPolicyCommand, GetDataProtectionPolicyCommand } from "@aws-
sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const putParams = { ResourceArn: "TOPIC_ARN", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };

const runPut = async () => {
  try {
    const data = await snsClient.send(new
 PutDataProtectionPolicyCommand(putParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runPut();
```

```
// Set the parameters
const getParams = { ResourceArn: "TOPIC_ARN" };

const runGet = async () => {
  try {
    const data = await snsClient.send(new
 GetDataProtectionPolicyCommand(getParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runGet();
```

Eliminazione di policy di protezione dei dati in Amazon SNS

È possibile eliminare le policy di protezione dei dati di Amazon SNS tramite l'API AWS, la AWS CLI, AWS CloudFormation oppure la AWS Management Console.

Per informazioni generali sulle policy di protezione dei dati di Amazon SNS, consulta Informazioni sulle policy di protezione dei dati.

Il numero e la dimensione delle risorse delle policy di protezione dei dati di Amazon SNS in un account AWS sono limitati. Per ulteriori informazioni, consultare Amazon SNS API throttling (Limitazione della larghezza di banda della rete nell'API SNS) nei Riferimenti generali di AWS.

Argomenti

- Eliminazione di policy di protezione dei dati (console)
- Eliminazione di una policy di protezione dei dati utilizzando una stringa JSON vuota
- Eliminazione di una policy di protezione dei dati tramite la AWS CLI

Eliminazione di policy di protezione dei dati (console)

Per eliminare una policy di protezione dei dati gestita (console)

1. Accedi alla console Amazon SNS.
2. Scegli l'argomento contenente la policy di protezione dei dati che desideri eliminare.
3. Scegliere Modifica.

4. Espandi la sezione Data protection policy (Policy di protezione dei dati).
5. Scegli Remove (Rimuovi) accanto alla dichiarazione della policy di protezione dei dati che desideri rimuovere.
6. Seleziona Salva modifiche.

Eliminazione di una policy di protezione dei dati utilizzando una stringa JSON vuota

È possibile eliminare una policy di protezione dei dati aggiornandola con una stringa JSON vuota.

Eliminazione di una policy di protezione dei dati tramite la AWS CLI

È possibile eliminare una policy di protezione dei dati tramite la AWS CLI.

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-
protection-policy ""
```

Identificatori di dati

Amazon SNS utilizza una combinazione di criteri e tecniche, tra cui machine learning e criteri di ricerca, per rilevare i dati sensibili. Questi criteri e tecniche, denominati collettivamente identificatori di dati, possono rilevare un elenco ampio e crescente di tipi di dati sensibili per molti Paesi e regioni. Gli identificatori di dati gestiti di Amazon SNS offrono tipi di dati pre-configurati per proteggere i dati finanziari, le informazioni sanitarie personali (PHI) e le informazioni di identificazione personale (PII). Puoi anche utilizzare identificatori di dati personalizzati per creare identificatori di dati personalizzati in base al tuo caso d'uso specifico.

Argomenti

- Utilizzo di identificatori di dati gestiti in Amazon SNS
- Utilizzo di identificatori di dati personalizzati in Amazon SNS

Utilizzo di identificatori di dati gestiti in Amazon SNS

Argomenti

- Cosa sono gli identificatori di dati gestiti?
- Tipi di dati sensibili: credenziali
- Tipi di dati sensibili: dispositivi

- Tipi di dati sensibili: finanziari
- Tipi di dati sensibili: dati sanitari protetti (PHI)
- Tipi di dati sensibili: informazioni personali di identificazione (PII)

Cosa sono gli identificatori di dati gestiti?

Gli identificatori di dati gestiti da Amazon SNS sono progettati per rilevare un tipo specifico di dati sensibili, ad esempio i numeri delle carte di credito, le chiavi di accesso segrete AWS o i numeri di passaporto per un determinato Paese o regione. Quando crei una policy di protezione dei dati, puoi configurare Amazon SNS in modo che utilizzi questi identificatori per analizzare i messaggi che attraversano l'argomento ed eseguire operazioni specifiche quando tali dati vengono rilevati.

Amazon SNS è in grado di rilevare le seguenti categorie di dati sensibili utilizzando gli identificatori di dati gestiti:

- Credenziali, ad esempio chiavi private o chiavi di accesso segrete AWS
- Identificatori di dispositivo, ad esempio indirizzo IP o indirizzo MAC
- Informazioni finanziarie, ad esempio i numeri di carte di credito
- Dati sanitari personali (PHI), ad esempio l'assicurazione sanitaria o i numeri di assistenza sanitaria
- Informazioni di identificazione personale (PII), ad esempio patenti di guida o codici fiscali

All'interno di ogni categoria, Amazon SNS è in grado di rilevare più tipi di dati sensibili. Negli argomenti di questa sezione sono elencati e descritti i vari tipi e tutti i requisiti pertinenti per la rilevazione. Per ogni tipo, vengono indicati anche gli identificatori univoci (ID) degli identificatori di dati gestiti progettati per rilevare i dati. Quando si crea una policy di protezione dei dati, è possibile utilizzare questo ID per includere l'identificatore di dati gestito da rilevare per la protezione dei dati dei messaggi.

Requisiti delle parole chiave

Per rilevare determinati tipi di dati sensibili, Amazon SNS cerca parole chiave in prossimità dei dati. Se questo è il caso di un particolare tipo di dati, in un argomento più avanti in questa sezione sono indicati i requisiti specifici relativi alle parole chiave per tali dati.

Le parole chiave non distinguono tra maiuscole e minuscole. Inoltre, se una parola chiave contiene uno spazio, Amazon SNS abbina automaticamente le varianti di parole chiave che non contengono lo spazio o che contengono un trattino basso (_) o un trattino (-) anziché lo spazio. In alcuni casi,

Amazon SNS espande o accorcia una parola chiave per rispondere alle varianti più comuni della parola chiave.

Identificatori di dati gestiti da Amazon SNS per tipi di dati sensibili

Nella tabella seguente sono elencati e descritti i tipi di dati relativi a credenziali, dispositivi, informazioni finanziarie e dati medici sanitari protetti (PHI) che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti. Questi dati si aggiungono a determinati tipi di dati che potrebbero anche essere considerati informazioni personali di identificazione (PII).

Gli identificatori di dati dipendenti dalla regione prevedono il nome dell'identificatore con un trattino e i codici a due lettere (ISO 3166-1 alpha-2). Ad esempio, DriversLicense -US.

Identificatore	Categoria	Paesi/lingue
BankAccountNumber	Servizi finanziari	DE, ES, FR, GB, IT
CepCode	Personale	BR
Cnpj	Personale	BR
CpfCode	Personale	BR
DriversLicense	Personale	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Integrità	US
ElectoralRollNumber	Personale	GB
HealthInsuranceCardNumber	Integrità	UE
HealthInsuranceClaimNumber	Integrità	US
HealthInsuranceNumber	Integrità	FR
HealthcareProcedureCode	Integrità	US

Identificatore	Categoria	Paesi/lingue
IndividualTaxIdentification Number	Personale	US
InseeCode	Personale	FR
MedicareBeneficiaryNumber	Integrità	US
NationalDrugCode	Integrità	US
NationalIdentificationNumber	Personale	DE, ES, IT
NationalInsuranceNumber	Personale	GB
NationalProviderId	Integrità	US
NhsNumber	Integrità	GB
NieNumber	Personale	ES
NifNumber	Personale	ES
PassportNumber	Personale	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	Personale	CA
PersonalHealthNumber	Integrità	CA
PhoneNumber	Personale	BR, DE, ES, FR, GB, IT, US
PostalCode	Personale	CA
RgNumber	Personale	BR
SocialInsuranceNumber	Personale	CA
Ssn	Personale	ES, US
TaxId	Personale	DE, ES, FR, GB
ZipCode	Personale	US

Identificatori supportati indipendenti dalla lingua/regione

Identificatore	Categoria
Indirizzo	Personale
AwsSecretKey	Credenziali
CreditCardExpiration	Servizi finanziari
CreditCardNumber	Servizi finanziari
CreditCardSecurityCode	Servizi finanziari
EmailAddress	Personale
IpAddress	Personale
LatLong	Personale
Nome	Personale
OpenSshPrivateKey	Credenziali
PgpPrivateKey	Credenziali
PkcsPrivateKey	Credenziali
PuttyPrivateKey	Credenziali
VehicleIdentificationNumber	Personale

Tipi di dati sensibili: credenziali

Nella tabella seguente sono elencati e descritti i tipi di credenziali che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Chiave di accesso segreta AWS	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Qualsiasi
Chiave privata OpenSSH	OpenSshPrivateKey	No	Qualsiasi
Chiave privata PGP	PgpPrivateKey	No	Qualsiasi
Chiave privata PKCS (Public-Key Cryptography Standard)	PkcsPrivateKey	No	Qualsiasi
Chiave privata PuTTY	PuttyPrivateKey	No	Qualsiasi

ARN degli identificatori di dati per i tipi di dati credenziali

Di seguito sono elencati i nomi delle risorse Amazon (ARN) per gli identificatori dei dati che è possibile utilizzare per le policy di protezione dei dati.

ARN degli identificatori dei dati delle credenziali
arn:aws:protezione dei dati: :aws:data-identifier/ AwsSecretKey
arn:aws:protezione dei dati: :aws:data-identifier/ OpenSshPrivateKey
arn:aws:protezione dei dati: :aws:data-identifier/ PgpPrivateKey
arn:aws:protezione dei dati: :aws:data-identifier/ PkcsPrivateKey
arn:aws:protezione dei dati: :aws:data-identifier/ PuttyPrivateKey

Tipi di dati sensibili: dispositivi

Nella tabella seguente sono elencati e descritti i tipi di identificatori di dispositivi che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Indirizzo IP	IpAddress	No	Qualsiasi

ARN degli identificatori di dati per i tipi di dati dispositivo

Di seguito sono elencati i nomi delle risorse Amazon (ARN) per gli identificatori dei dati che è possibile utilizzare per le policy di protezione dei dati.

ARN di identificatore dei dati del dispositivo
arn:aws:protezione dei dati: :aws:data-identifier/ IpAddress

Tipi di dati sensibili: finanziari

Nella tabella seguente sono elencati e descritti i tipi di informazioni finanziarie che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero del conto bancario	BankAccountNumber BankAccountNumber-US	Sì, consulta Parole chiave per i numeri di conto bancario.	Sono incluse le seguenti informazioni: codici IBAN (International Bank Account Number) composti da un massimo	Francia, Germania, Italia, Regno Unito, Spagna

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
			di 34 caratteri alfanumerici, inclusi elementi come il prefisso internazionale.	
Data di scadenza della carta di credito	CreditCardExpiration	data scadenza, anno scadenza, giorno scadenza, scadenza, scade	–	Qualsiasi
Dati della banda magnetica della carta di credito	CreditCardMagneticStripe	Sì, inclusi: dati della carta, iso7813, magnetica, banda magnetica, strisciare.	Ciò include le tracce 1 e 2.	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero di carta di credito	CreditCardNumber	numero di conto, american express, amex, carta bancaria, carta, num carta, numero carta, n. cc, ncc, carta di debito, credito, n. carta di credito, dankort, debito, carta di debito, diners club, discover, electron, codice di verifica elo, japanese card bureau, jcb, mastercard, mc, pan, numero conto di pagamento, numero carta di pagamento, pcn, union pay, visa	Il rilevamento richiede che i dati siano una sequenza di 13-19 cifre che rispetti la formula Luhn check e utilizzi un prefisso numerico di carta standard per uno dei seguenti tipi di carte di credito: American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card Bureau (JCB), Mastercard e Visa (link in apice sotto 1). UnionPay	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Codice di verifica della carta di credito	CreditCardSecurityCode	ID carta, codice identificativo carta, numero di identificazione della carta, codice di sicurezza della carta, codice di convalida della carta, dati di verifica della carta, valore di verifica della carta, cvc, cvc2, cvv, cvv2, codice di verifica elo	–	Qualsiasi

1. Amazon SNS non segnala le occorrenze delle seguenti sequenze, che gli emittenti delle carte di credito hanno riservato a test pubblici:

   122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 4444333322221111, 4462030000000000, 4484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017, 5204740009900014, 5420923878724339, 5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194, 5555555555554444, 5610591081018250, 6011000990139424, 6011000400000000,

6011111111111117, 630490017740292441, 630495060000000000, 6331101999990016, 6759649826438453, 6799990100000000019 e 76009244561.

Parole chiave per i numeri di conto bancario

Utilizza le seguenti parole chiavi per rilevare i codici IBAN (International Bank Account Number), composti da un massimo di 34 caratteri alfanumerici, inclusi elementi come il prefisso internazionale.

Paese o regione	Parole chiave			
Francia	account code, account number, accountno #, accountnu mber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
Germania	account code, account number, accountno #, accountnu mber#, bankleitz ahl, bban, customer account id, customer account number, customer bank			

Paese o regione	Parole chiave			
	account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
Italia	account code, account number, accountno #, accountnumber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

Paese o regione	Parole chiave			
Spagna	account code, account number, accountno #, accountnu mber#, bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente			
UK	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

Paese o regione	Parole chiave			
US	conto bancario, c. bancario, conto corrente, c. corrente, conto di deposito, c. di deposito, conto di risparmio, c. di risparmio			

ARN degli identificatori di dati per i tipi di dati finanziari

Di seguito sono elencati i nomi delle risorse Amazon (ARN) per gli identificatori dei dati che è possibile utilizzare per le policy di protezione dei dati.

ARN degli identificatore dei dati finanziari
arn:aws:protezione dei dati: :aws:data-identifier/ BankAccountNumber -DE
arn:aws:protezione dei dati: :aws:data-identifier/ -ES BankAccountNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -FR BankAccountNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -GB BankAccountNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -IT BankAccountNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US BankAccountNumber
arn:aws:protezione dei dati: :aws:data-identifier/ CreditCardExpiration
arn:aws:protezione dei dati: :aws:data-identifier/ CreditCardNumber
arn:aws:protezione dei dati: :aws:data-identifier/ CreditCardSecurityCode

Tipi di dati sensibili: dati sanitari protetti (PHI)

Nella tabella seguente sono elencati e descritti i tipi di dati sanitari protetti (PHI) che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Numero di registrazione DEA (Drug Enforcement Agency)	DrugEnforcementAgencyNumber	dea number, dea registration	US
Numero EHIC (Health Insurance Card)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance,	UE

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
		numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	US
Numero di identificazione medica e assistenza sanitaria	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR
Codice HCPCS (Healthcare Common Procedure Coding System)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	US

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Numero MBN (Medicare Beneficiary Number)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	US
National Drug Code (NDC)	NationalDrugCode	national drug code, ndc	US
National Provider Identifier (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	US
Numero NHS (National Health Service)	NhsNumber	national health service, NHS	GB
Personal Health Number (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Parole chiave per numeri di identificazione medica e assistenza sanitaria

Per rilevare vari tipi di assicurazioni sanitarie e numeri di identificazione medica, Amazon SNS richiede la prossimità di parole chiave e numeri. Sono inclusi i numeri della tessera sanitaria europea (UE, Finlandia), i numeri di assicurazione sanitaria (Francia), gli identificatori dei beneficiari Medicare (Stati Uniti), i numeri di assicurazione sanitaria (Regno Unito), i numeri NHS (Regno Unito) e i numeri Personal Health Number (PHN) (Canada).

Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Canada	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
UE	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
Finlandia	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti

Paese o regione	Parole chiave
Francia	carte d'assuré social, carte vitale, insurance card
UK	servizio sanitario nazionale, NHS
US	mbi, medicare beneficiary

ARN degli identificatori di dati per i tipi di dati sanitari protetti (PHI)

Di seguito sono elencati i nomi delle risorse Amazon (ARN) per gli identificatori di dati possono essere utilizzati nelle policy di protezione dei dati sanitari protetti.

ARN per identificatori di dati sanitari protetti
arn:aws:protezione dei dati: :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US HealthcareProcedureCode
arn:aws:protezione dei dati: :aws:data-identifier/ -UE HealthInsuranceCardNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US HealthInsuranceClaimNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -FR HealthInsuranceNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US MedicareBeneficiaryNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US NationalDrugCode
arn:aws:protezione dei dati: :aws:data-identifier/ -GB NationalInsuranceNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US NationalProviderId
arn:aws:protezione dei dati: :aws:data-identifier/ -GB NhsNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -CA PersonalHealthNumber

Tipi di dati sensibili: informazioni personali di identificazione (PII)

Nella tabella seguente sono elencati e descritti i tipi di informazioni personali di identificazione (PII) che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Data di nascita	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	Il supporto include la maggior parte dei formati di data, ad esempio tutte le cifre e le combinazioni di cifre e nomi dei mesi. I componenti della data possono essere separati da spazi, barre (/) o trattini (-).	Qualsiasi
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brasile
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	Brasile

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	Brasile

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero identificativo della patente di guida	DriversLicense	Sì, consulta Parole chiave per i numeri identificativi delle patenti di guida.	–	Australia, Austria, Belgio, Bulgaria, Canada, Cipro, Croazia, Danimarca, Estonia, Finlandia, Francia, Germania, Grecia, Irlanda, Italia, Lettonia, Lituania, Lussemburgo, Malta, Paesi Bassi, Polonia, Portogallo, Regno Unito, Repubblica Ceca, Romania, Slovacchia, Slovenia, Spagna, Stati Uniti, Svezia, Ungheria

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero di lista elettorale	ElectoralRollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
Identificazione del singolo contribuente	IndividualTaxIdentificationNumber	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	US
Institut national de la statistique et des études économiques (INSEE)	InseeCode	Sì, consulta Parole chiave per i numeri di carta d'identità.	–	Francia

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numeri di carta d'identità	NationalIdentificationNumber	Sì, consulta Parole chiave per i numeri di carta d'identità.	Sono inclusi gli identificatori DNI (Documento Nacional de Identidad, Spagna), il codice fiscale (Italia) e i numeri di carta d'identità nazionale (Germania).	Germania, Italia, Spagna
Numero NINO (National Insurance Number)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	UK
Número de identidad de extranjero (NIE)	NieNumber	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	Spagna

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Número de Identificación Fiscal (NIF)	NifNumber	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	Spagna
Numero di passaporto	PassportNumber	Sì, consulta Parole chiave per i numeri di passaporto.	–	Canada, Francia, Germania, Italia, Regno Unito, Spagna, Stati Uniti

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero di residenza permanente (Green Card)	Permanent Residence Number	carte résident permanent , numéro carte résident permanent, numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	–	Canada

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero di telefono	PhoneNumber	Brasile: le parole chiave includono anche: cel, celular, fone, móvel, número residencial, numero residencial, telefone Altri paesi: cellulare, contatto, fax, numero di fax, smartphone, telefono, mobile, numero di telefono	Sono inclusi i numeri verdi negli Stati Uniti e i numeri di fax. Se una parola chiave si trova in prossimità dei dati, il numero non deve includere il prefisso internazionale. Se una parola chiave non è in prossimità dei dati, il numero deve includere un prefisso internazionale.	Brasile, Canada, Francia, Germania, Italia, Regno Unito, Spagna, Stati Uniti
Codice postale	PostalCode	No	–	Canada
Registro Geral (RG)	RgNumber	Sì, consulta Parole chiave per i numeri di carta d'identità.	–	Brasile
Social Insurance Number (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Canada

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Social Security number (SSN)	Ssn	Spagna: número de la seguridad social, n. di previdenza sociale, numero di previdenza sociale, npreviden zasociale#, ssn, ssn# Stati Uniti: previdenza sociale, ss#, ssn	–	Spagna, Stati Uniti
Numero identificativo del contribuente o codice fiscale	TaxId	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	Sono inclusi TIN (Francia); Steueridentifikationsnummer (Germania); CIF (Spagna) e TRN, UTR (Regno Unito).	Francia, Germania, Regno Unito, Spagna
Codice postale (Stati Uniti)	ZipCode	zip code, zip+4	–	US

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Indirizzo postale	Address	No	Sebbene non sia richiesta una parola chiave, il rilevamento richiede che l'indirizzo includa il nome di una città o di un luogo e un CAP o un codice postale.	Australia, Canada, Francia, Germania, Italia, Regno Unito, Spagna, Stati Uniti
Indirizzo e-mail	EmailAddress	email, indirizzo email, e mail, indirizzo e mail	–	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Coordinate del sistema di posizionamento globale (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS è in grado di rilevare le coordinate GPS se le coordinate di latitudine e longitudine sono memorizzate in coppia e sono nel formato di gradi decimali (DD), ad esempio 41,948614, -87.65531 1. Il supporto non include le coordinate nel formato DDM (Gradi Decimali Minuti), ad esempio 41°56.9168'N 87°39.3187'W, o nel formato Gradi, Minuti, Secondi (DMS), ad esempio 41°56'55.0104"N 87°39'19.1196"W.	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Nome completo	Name	No	Amazon SNS è in grado di rilevare solo i nomi completi. Il supporto è limitato ai set di caratteri latini.	Qualsiasi
Numeri di matricola del veicolo (VIN)	VehicleIdentificationNumber	Fahrgestellnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	Amazon SNS è in grado di rilevare i numeri VIN costituiti da una sequenza di 17 caratteri e conformi agli standard ISO 3779 e 3780. Questi standard sono stati progettati per l'uso a livello mondiale.	Qualsiasi

Parole chiave per i numeri identificativi delle patenti di guida

Per rilevare vari tipi di numeri identificativi delle patenti di guida, Amazon SNS richiede la prossimità di parole chiave e numeri. Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Australia	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
Austria	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Belgio	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgaria	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Canada	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit, permis de conduire
Croazia	vozačka dozvola
Cipro	άδεια οδήγησης

Paese o regione	Parole chiave
Repubblica Ceca	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
Danimarca	kørekort, kørekortnummer
Estonia	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finlandia	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
Francia	permis de conduire
Germania	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Grecia	δεια οδήγησης, adeia odigisis
Ungheria	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Irlanda	ceadúnas tiomána
Italia	patente di guida, patente di guida numero, patente guida, patente guida numero
Lettonia	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Lituania	vairuotojo pažymėjimas

Paese o regione	Parole chiave
Lussemburgo	fahrerlaubnis, führerschäin
Malta	liċenzja tas-sewqan
Paesi Bassi	permis de conduire, rijbewijs, rijbewijsnummer
Polonia	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
Portogallo	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
Romania	numărul permisului de conducere, permis de conducere
Slovacchia	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Slovenia	vozniško dovoljenje
Spagna	carnet conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
Svezia	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

Paese o regione	Parole chiave
UK	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
US	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

Parole chiave per i numeri di carta d'identità

Per rilevare vari tipi di numeri di carta d'identità, Amazon SNS richiede la prossimità di parole chiave e numeri. Sono inclusi gli identificatori DNI (Documento Nacional de Identidad, Spagna), i codici INSEE (Institut national de la statistique et des études économiques), i numeri delle carte d'identità tedesche e i numeri RG (Registro Geral, Brasile).

Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Brasile	registro geral, rg
Francia	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance

Paese o regione	Parole chiave
	number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Germania	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italia	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Spagna	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationali dno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

Parole chiave per i numeri di passaporto

Per rilevare vari tipi di numeri di passaporto, Amazon SNS richiede la prossimità di parole chiave e numeri. Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Canada	passeport, passeport#, passport, passport#, passportno, passportno#
Francia	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

Paese o regione	Parole chiave
Germania	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
Italia	italian passport number, numéro passeport , numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Spagna	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
UK	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
US	passport, travel document

Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento

Per rilevare vari tipi di numeri di identificazione dei contribuenti e di numeri di riferimento, Amazon SNS richiede la prossimità di parole chiave e numeri. Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Brasile	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf

Paese o regione	Parole chiave
Francia	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Germania	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
Spagna	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
UK	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
US	Individual Taxpayer Identification Numbers (ITIN o i.t.i.n.)

ARN degli identificatori dei dati per le informazioni personali di identificazione (PII)

Nella seguente tabella sono elencati i nomi delle risorse Amazon (ARN) per gli identificatori dei dati che è possibile utilizzare per le policy di protezione dei dati.

ARN delle informazioni personali di identificazione (PII)
arn:aws:dataprotection::aws:data-identifier/Address
arn:aws:protezione dei dati: :aws:data-identifier/ -BR CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR

ARN delle informazioni personali di identificazione (PII)
arn:aws:protezione dei dati: :aws:data-identifier/ -BR CpfCode
arn:aws:protezione dei dati: :aws:data-identifier/ DateOfBirth
arn:aws:protezione dei dati: :aws:data-identifier/ -AT DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -AU DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -BE DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -BG DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -CA DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ DriversLicense -CY
arn:aws:protezione dei dati: :aws:data-identifier/ DriversLicense -CZ
arn:aws:protezione dei dati: :aws:data-identifier/ -DE DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -DK DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -EE DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -ES DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -FI DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -FR DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -GB DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -GR DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -HR DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -HU DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -IE DriversLicense

ARN delle informazioni personali di identificazione (PII)
arn:aws:protezione dei dati: :aws:data-identifier/ -IT DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -LT DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -LU DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -LV DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ DriversLicense -MT
arn:aws:protezione dei dati: :aws:data-identifier/ -NL DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -PL DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -PT DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -RO DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -SE DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -SI DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -SK DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -US DriversLicense
arn:aws:protezione dei dati: :aws:data-identifier/ -GB ElectoralRollNumber
arn:aws:protezione dei dati: :aws:data-identifier/ EmailAddress
arn:aws:protezione dei dati: :aws:data-identifier/ -US IndividualTaxIdentificationNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -FR InseeCode
arn:aws:protezione dei dati: :aws:data-identifier/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws:protezione dei dati: :aws:data-identifier/ -DE NationalIdentificationNumber

ARN delle informazioni personali di identificazione (PII)
arn:aws:protezione dei dati: :aws:data-identifier/ -ES NationalIdentificationNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -IT NationalIdentificationNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -ES NieNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -ES NifNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -CA PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -DE PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -ES PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -FR PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -GB PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -IT PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US PassportNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -CA PermanentResidenceNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -BR PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -DE PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -ES PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -FR PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -GB PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -IT PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US PhoneNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -CA PostalCode

ARN delle informazioni personali di identificazione (PII)
arn:aws:protezione dei dati: :aws:data-identifier/ -BR RgNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -CA SocialInsuranceNumber
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:aws:protezione dei dati: :aws:data-identifier/ -DE TaxId
arn:aws:protezione dei dati: :aws:data-identifier/ -ES TaxId
arn:aws:protezione dei dati: :aws:data-identifier/ -FR TaxId
arn:aws:protezione dei dati: :aws:data-identifier/ -GB TaxId
arn:aws:protezione dei dati: :aws:data-identifier/ VehicleIdentificationNumber
arn:aws:protezione dei dati: :aws:data-identifier/ -US ZipCode

Utilizzo di identificatori di dati personalizzati in Amazon SNS

Argomenti

Cosa sono gli identificatori di dati personalizzati?

Gli identificatori di dati personalizzati (CDI) consentono di definire espressioni regolari personalizzate che possono essere utilizzate nella policy di protezione dei dati. Utilizzando gli identificatori di dati personalizzati, puoi indirizzare i casi d'uso delle informazioni di identificazione personale (PII) specifici dell'azienda che gli identificatori di dati gestiti non sono in grado di fornire. Ad esempio, puoi utilizzare un identificatore di dati personalizzato per cercare gli ID dei dipendenti specifici dell'azienda. Gli identificatori di dati personalizzati possono essere utilizzati insieme agli identificatori di dati gestiti.

Utilizzo degli identificatori di dati personalizzati nella policy di protezione dei dati

La seguente policy di protezione dei dati indica all'argomento Amazon SNS di rilevare i payload che contengono ID dei dipendenti specifici dell'azienda, quindi di mascherare questi ID utilizzando il simbolo cancelletto (#).

1. Creazione di un blocco `Configuration` all'interno della policy di protezione dei dati.
2. Inserisci un `Name` per l'identificatore di dati personalizzato. Ad esempio, **`EmployeeId`**.
3. Inserisci un `Regex` per l'identificatore di dati personalizzato. Ad esempio, **`EID-\d{9}-US`**.
4. Riferimento al seguente identificatore di dati personalizzato in una dichiarazione di policy.

```
{
 "Name": "__example_data_protection_policy",
 "Description": "Example data protection policy",
 "Version": "2021-06-01",
 "Configuration": {
   "CustomDataIdentifier": [
     {"Name": "EmployeeId", "Regex": "EID-\d{9}-US"}
   ]
 },
 "Statement": [
   {
     "DataDirection": "Inbound",
     "Principal": ["*"],
     "DataIdentifier": [
        "EmployeeId"
     ],
     "Operation": {
       "Deidentify": {
         "MaskConfig": {
           "MaskWithCharacter": "#"
         }
       }
     }
   }
 ]
}
```

5. (Facoltativo) Continua ad aggiungere altri identificatori di dati personalizzati al blocco `Configuration`, se necessario. Le policy di protezione dei dati attualmente supportano un massimo di 10 identificatori di dati personalizzati.

Vincoli degli identificatori di dati personalizzati

Gli identificatori di dati personalizzati di Amazon SNS presentano le seguenti limitazioni:

- Ciascuna policy di protezione dei dati attualmente supporta un massimo di 10 identificatori di dati personalizzati.
- I nomi degli identificatori di dati personalizzati hanno una lunghezza massima di 128 caratteri. Sono supportati i seguenti caratteri:
  - Alfanumerici: (a-zA-Z0-9)
  - Simboli: ('_' | '-')
- RegEx ha una lunghezza massima di 200 caratteri. Sono supportati i seguenti caratteri:
  - Alfanumerici: (a-zA-Z0-9)
  - Simboli: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - Caratteri riservati RegEx: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- Gli identificatori di dati personalizzati non possono condividere lo stesso nome di un identificatore di dati gestito.
- Gli identificatori di dati personalizzati devono essere specificati in ogni policy di protezione dei dati per ogni argomento di Amazon SNS.

Consegna dei messaggi Amazon SNS

In questa sezione viene descritto come funziona la consegna dei messaggi.

Argomenti

Consegna di messaggi non elaborati Amazon SNS

Per evitare che gli endpoint Amazon Data Firehose, Amazon SQS e HTTP/S elaborino la formattazione JSON dei messaggi, Amazon SNS consente la consegna di messaggi non elaborati:

- Quando abiliti la consegna di messaggi non elaborati per gli endpoint Amazon Data Firehose o Amazon SQS, tutti i metadati Amazon SNS vengono rimossi dal messaggio pubblicato e il messaggio viene inviato così com'è.
- Quando si abilita il recapito dei messaggi non elaborati per gli endpoint HTTP/S, l'intestazione HTTP `x-amz-sns-rawdelivery` con il valore impostato su `true` viene aggiunta al messaggio, indicando che il messaggio è stato pubblicato senza formattazione JSON.
- Quando si abilita la consegna dei messaggi non elaborati per gli endpoint HTTP/S, vengono consegnati il corpo del messaggio, l'IP del client e le intestazioni richieste. Quando si specificano gli attributi del messaggio, questi non verranno inviati.
- Quando si abilita il recapito di messaggi non elaborati per gli endpoint Firehose, il corpo del messaggio viene recapitato. Quando si specificano gli attributi del messaggio, questi non verranno inviati.

Per abilitare la consegna di messaggi non elaborati utilizzando un AWS SDK, è necessario utilizzare l'azione `SetSubscriptionAttribute` API e impostare il valore dell'`RawMessageDelivery`attributo su. `true`

Abilitazione della consegna di messaggi non elaborati utilizzando la AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Argomenti, scegli un argomento sottoscritto a un endpoint Firehose, Amazon SQS o HTTP/S.
4. Nella ***MyTopic***pagina, nella sezione Abbonamento, scegli un abbonamento e scegli Modifica.
5. Sulla pagina Modificare***esempio1-23BC-4567-D890-EF12G3HiJ456*** della sezione Dettagli, scegliere Abilita consegna di messaggi non elaborati.
6. Seleziona Save changes (Salva modifiche).

Esempi di formati di messaggi

Negli esempi seguenti, lo stesso messaggio viene inviato due volte alla stessa coda Amazon SQS. L'unica differenza è che il recapito dei messaggi non elaborati è disabilitato per il primo messaggio e abilitato per il secondo.

- Consegna di messaggi non elaborati disabilitato

```
{
  "Type": "Notification",
  "MessageId": "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
  "TopicArn": "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
  "Subject": "TestSubject",
  "Message": "This is a test message.",
  "Timestamp": "2021-02-16T21:41:19.978Z",
  "SignatureVersion": "1",
  "Signature":
 "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/
zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm
+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy
+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/
LsDr0Iak4ZDpg8dXg==",
  "SigningCertURL": "https://sns.us-east-2.amazonaws.com/
SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
```

```
  "UnsubscribeURL": "https://sns.us-east-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
}
```

- Consegna di messaggi non elaborati abilitato

```
This is a test message.
```

Attributi dei messaggi e consegna di messaggi non elaborati per gli abbonamenti Amazon SQS

Amazon SNS supporta la consegna degli attributi del messaggio, che consentono di fornire elementi di metadati strutturati, come timestamp, dati geospaziali, firme e identificatori, sul messaggio. Per gli abbonamenti Amazon SQS con Raw Message Delivery abilitato, è possibile inviare un massimo di 10 attributi di messaggio. Per inviare più di 10 attributi del messaggio, devi disabilitare Raw Message Delivery. Tuttavia, Amazon SNS elimina i messaggi con più di 10 attributi di messaggio diretti agli abbonamenti Amazon SQS con Raw Message Delivery abilitato, trattandoli come errori lato client.

Invio di messaggi Amazon SNS a una coda Amazon SQS; in un altro account

Questo documento descrive come pubblicare una notifica in un argomento Amazon SNS con una o più sottoscrizioni a code Amazon SQS in un altro account. La procedura di configurazione di argomento e code è identica a quella utilizzata quando questi si trovano nello stesso account (vedi Fan-out a code Amazon SQS). La differenza principale riguarda la gestione della conferma della sottoscrizione, che varia in base al modo in cui viene eseguita la sottoscrizione della coda all'argomento.

È consigliabile quando possibile seguire i passaggi di cui si fa riferimento nella sezione Creazione della sottoscrizione da parte del proprietario della coda, perché la conferma è automatica quando il proprietario della coda crea la sottoscrizione.

> **Note**
>
> Se la coda Amazon SQS ha un volume di messaggi elevato, raccomandiamo che il proprietario della coda crei la sottoscrizione.

Argomenti

Creazione della sottoscrizione da parte del proprietario della coda

L'account che ha creato la coda Amazon SQS è il proprietario della coda. Quando il proprietario della coda crea la sottoscrizione, la conferma della sottoscrizione non è necessaria. La coda inizia a ricevere le notifiche dall'argomento al termine dell'operazione `Subscribe`. Per consentire al proprietario della coda di effettuare la sottoscrizione all'argomento, il proprietario dell'argomento deve autorizzare l'account del proprietario della coda a chiamare l'operazione `Subscribe` sull'argomento.

Fase 1: per impostare la policy dell'argomento utilizzando AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Selezionare un argomento, quindi scegliere Edit (Modifica).
4. Nella pagina Edit ***myTopic*** (Modifica myTopic) espandere la sezione Policy di accesso.
5. Immettere la seguente policy:

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "sns:Subscribe",
         "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
   ]
}
```

   Questa policy consente all'account `111122223333` di chiamare `sns:Subscribe` su `MyTopic` nell'account `123456789012`.

Un utente con le credenziali per l'account `111122223333` può sottoscrivere a `MyTopic`. Questa autorizzazione consente all'ID account di delegare l'autorizzazione al proprio utente/ruolo IAM. Solo l'account root o gli utenti amministratori saranno autorizzati a richiamare `sns:Subscribe`. Anche l'utente/ruolo IAM deve avere `sns:subscribe` per consentire alla propria coda di iscriversi.

6. Seleziona Salva modifiche.

   Un utente con le credenziali per l'account `111122223333` può sottoscrivere a MyTopic.

Fase 2: aggiungere una sottoscrizione di una coda Amazon SQS a un argomento in un altro Account AWS utilizzando AWS Management Console.

Prima di iniziare, assicurarsi di disporre degli ARN per l'argomento e la coda e di aver fornito l'autorizzazione all'argomento per inviare messaggi alla coda.

1. Accedere alla console Amazon SQS.
2. Nel pannello di navigazione, scegliere Code.
3. Dall'elenco di code, scegliere la coda per iscriversi all'argomento Amazon SNS.
4. Scegli Subscribe to Amazon SNS topic (Iscriviti all'argomento Amazon SNS).
5. Da Specify an Amazon SNS topic available for this queue menu (Specifica un argomento Amazon SNS disponibile per questo menu di coda), scegliere Amazon SNS topic (Argomento Amazon SNS) per la coda.
6. Scegliere Enter Amazon SNS topic ARN (Inserisci ARN dell'argomento Amazon SNS) e quindi inserire l'Amazon Resource Name (ARN) dell'argomento.
7. Seleziona Salva.

> **Note**
>
> - Per poter comunicare con il servizio, la coda deve disporre delle autorizzazioni per Amazon SNS.
> - Poiché sei il proprietario della coda, non devi confermare la sottoscrizione.

Creazione di una sottoscrizione da parte di un utente non proprietario della coda

Qualsiasi utente che crea una sottoscrizione ma non è il proprietario della coda deve confermare la sottoscrizione.

Quando si utilizza l'azione `Subscribe`, Amazon SNS invia una conferma di sottoscrizione alla coda. La sottoscrizione viene visualizzata nella console Amazon SNS, con l'ID di sottoscrizione impostato su Conferma in sospeso.

Per confermare la sottoscrizione, un utente autorizzato a leggere i messaggi dalla coda deve recuperare l'URL di conferma della sottoscrizione e il proprietario della sottoscrizione deve confermare la sottoscrizione utilizzando l'URL di conferma della sottoscrizione. Fino alla conferma della sottoscrizione, nessuna delle notifiche pubblicate nell'argomento viene inviata alla coda. Per confermare la sottoscrizione, è possibile utilizzare la console Amazon SQS o l'azione `ReceiveMessage`.

> **Note**
>
> Prima di sottoscrivere un endpoint all'argomento, assicurarsi che la coda possa ricevere messaggi dall'argomento impostando l'autorizzazione `sqs:SendMessage` per la coda. Per ulteriori informazioni, consulta Fase 2: Concedere all'argomento Amazon SNS l'autorizzazione a inviare messaggi alla coda Amazon SQS.

Fase 1: aggiunta di una sottoscrizione di una coda Amazon SQS a un argomento in un altro Account AWS utilizzando la AWS Management Console.

Prima di iniziare, assicurarsi di disporre degli ARN per l'argomento e la coda e di aver fornito l'autorizzazione all'argomento per inviare messaggi alla coda.

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegliere Create subscription (Crea sottoscrizione).
4. Nella pagina Create subscription (Crea sottoscrizione), nella sezione Dettagli, eseguire queste operazioni:

   a. Per Argomento ARN, immettere l'ARN dell'argomento.

b. Per Protocollo, scegliere Amazon SQS.

c. Per Endpoint, immettere l'ARN della coda.

d. Scegli Create Subscription (Crea sottoscrizione).

> **Note**
>
> - Per poter comunicare con il servizio, la coda deve disporre delle autorizzazioni per Amazon SNS.

Il seguente è un esempio di istruzione di policy che consente all'argomento Amazon SNS di inviare un messaggio alla coda Amazon SQS.

```
{
   "Sid": "Stmt1234",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "sqs:SendMessage",
   "Resource": "arn:aws:sqs:us-west-2:111111111111:QueueName",
   "Condition": {
      "ArnEquals": {
         "aws:SourceArn": "arn:aws:sns:us-west-2:555555555555:TopicName"
      }
   }
}
```

Fase 2: conferma di una sottoscrizione utilizzando la AWS Management Console

1. Accedere alla console Amazon SQS.
2. Selezionare la coda con una sottoscrizione in sospeso all'argomento.
3. Scegli Send and receive messages (Invia e ricevi messaggi), quindi seleziona Poll for messages (Polling per i messaggi).

   Nella coda viene ricevuto un messaggio con la conferma della sottoscrizione.
4. Nella colonna Corpo eseguire le operazioni seguenti:

   a. Scegliere Maggiori dettagli.

b. Nella finestra di dialogo Message Details (Dettagli messaggio), trovare e annotare il valore SubscribeURL. Questo è il collegamento di sottoscrizione (esempio qui di seguito). Per ulteriori dettagli sulla convalida dei token API, consulta `ConfirmSubscription`nella Documentazione di riferimento delle API di Amazon SNS.

```
https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

c. Prendere nota del collegamento di conferma della sottoscrizione. L'URL deve essere passato dal proprietario della coda al proprietario della sottoscrizione. Il proprietario della sottoscrizione deve inserire l'URL nella console Amazon SNS.

5. Accedere come proprietario della sottoscrizione alla console Amazon SNS. Il proprietario della sottoscrizione effettua la conferma.
6. Scelta dell'argomento pertinente.
7. Scegliere la sottoscrizione pertinente nella tabella degli elenchi di sottoscrizione dell'argomento. È etichettato come "In attesa di conferma".
8. Scegliere Confirm subscription (Conferma sottoscrizione).
9. Viene visualizzato un modale che richiede il collegamento di conferma della sottoscrizione. Incollare il collegamento di conferma della sottoscrizione.
10. Selezionare Confirm subscription (Conferma sottoscrizione).nel modale.

    Viene visualizzata una risposta XML, ad esempio:

```
<ConfirmSubscriptionResponse>
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-east-2:123456789012:MyTopic:1234a567-
bc89-012d-3e45-6fg7h890123i</SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>abcd1efg-23hi-jkl4-m5no-p67q8rstuvw9</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

    La coda per la quale hai confermato la sottoscrizione è pronta a ricevere messaggi dall'argomento.

11. (Facoltativo) Se visualizzi la sottoscrizione dell'argomento nella console Amazon SNS, puoi verificare che il messaggio Pending Confirmation (Conferma in sospeso) è stato sostituito dall'ARN della sottoscrizione nella colonna Subscription ID (ID sottoscrizione).

Come faccio a forzare una sottoscrizione a richiedere l'autenticazione per le richieste di annullamento della sottoscrizione?

Il proprietario della sottoscrizione deve impostare il flag `AuthenticateOnUnsubscribe` su true alla conferma della sottoscrizione.

- `AuthenticateOnUnsubscribe` viene automaticamente impostato su true quando il proprietario della coda crea la sottoscrizione.
- `AuthenticateOnUnsubscribe` non può essere impostato su true quando si passa al collegamento di conferma della sottoscrizione senza autenticazione.

Invio di messaggi Amazon SNS a una coda Amazon SQS o AWS Lambda funzione in una regione diversa

Amazon SNS supporta le spedizioni tra aree geografiche, sia per le regioni abilitate per impostazione predefinita che per Regioni opt-in. Per l'elenco corrente di regioni AWS supportate da Amazon SNS, incluse le regioni abilitate, consultare Amazon Simple Notification Service endpoints and quotas (Endpoint e quote Amazon Simple Notification Service) nei Riferimenti generali di Amazon Web Services.

Amazon SNS supporta la consegna in più aree geografiche delle notifiche alle code Amazon SQS e a AWS Lambda funzioni . Quando una delle regioni è una regione opt-in, devi specificare un entità Amazon SNS diverso nella policy della risorsa sottoscritta.

Il comando di sottoscrizione ad Amazon SNS deve essere eseguito nell'account di destinazione nella Regione in cui è ospitato Amazon SNS. Ad esempio, se Amazon SNS si trova nell'account "A" nella regione us-east-1 e la funzione Lambda si trova nell'account "B" nella regione us-east-2, il comando CLI di sottoscrizione deve essere eseguito nell'account "A" nella regione us-east-1.

Regioni opt-in

Amazon SNS supporta le seguenti regioni opt-in:

Nome regione	Regione
Regione Africa (Città del Capo)	af-south-1
Regione Asia Pacifico (Hong Kong)	ap-east-1
Regione Asia Pacifico (Hyderabad)	ap-south-2
Regione Asia Pacifico (Giacarta)	ap-southeast-3
Regione Asia Pacifico (Melbourne)	ap-southeast-4
Regione Asia Pacifico (Osaka-Locale)	ap-northeast-3
Regione Europa (Milano)	eu-south-1
Regione Europa (Spagna)	eu-south-2
Regione Europa (Zurigo)	eu-central-2
Regione di Israele (Tel Aviv)	il-central-1
Regione Medio Oriente (Bahrein)	me-south-1
Regione Medio Oriente (EAU)	me-central-1

Per informazioni sull'attivazione di una regione opt-in, vedere Gestione delle AWS regioni in.Riferimenti generali di Amazon Web Services

Quando si utilizza Amazon SNS per recapitare messaggi dalle regioni di accesso alle regioni abilitate per impostazione predefinita, è necessario modificare le policy delle risorse create per la coda. Sostituire l'entità `sns.amazonaws.com` con `sns.<opt-in-region>.amazonaws.com`. Ad esempio:

- Ad esempio, se si desidera sottoscrivere una coda Amazon SQS negli Stati Uniti orientali (Virginia settentrionale) a un argomento Amazon SNS in Asia Pacifico (Hong Kong), modificare il principale nella policy della coda in `sns.ap-east-1.amazonaws.com`. Le regioni di attivazione includono tutte le regioni lanciate dopo il 20 marzo 2019 e includono Asia Pacifico (Hong Kong), Asia Pacific (Giacarta) Medio Oriente (Bahrein), Europa (Milano) e Africa (Città del Capo). Le regioni lanciate prima del 20 marzo 2019 sono abilitate per impostazione predefinita.

Supporto per la consegna tra regioni ad Amazon SQS

Tipo di consegna tra regioni	Supportato/Non supportato	
Regione abilitata per impostazione predefinita per la regione di attivazione	Supportato utilizzando `sns.<opt-in-region>.amazonaws.com` nel principale di servizio per la coda	
Regione di attivazione per la regione abilitata per impostazione predefinita	Supportato utilizzando `sns.<opt-in-region>.amazonaws.com` nel principale di servizio per la coda	
Regione di attivazione a regione di attivazione	Non supportato	

Di seguito è riportato un esempio di dichiarazione sulla politica di accesso che consente a un argomento Amazon SNS in una regione opzionale (af-south-1) di effettuare la consegna a una coda Amazon SQS in una regione (us-east-1). enabled-by-default Contiene la configurazione del principale di servizio regionalizzato necessario nel percorso `Statement/Principal/Service`.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.af-south-1.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
      "Condition": {
```

```
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
        }
      }
    },
    ...
  ]
}
```

- Per sottoscrivere una AWS Lambda funzione negli Stati Uniti orientali (Virginia settentrionale) a un argomento di Amazon SNS in Asia Pacifico (Hong Kong), modifica il principio nella politica AWS Lambda della funzione in. `sns.ap-east-1.amazonaws.com` Le regioni di attivazione includono tutte le regioni lanciate dopo il 20 marzo 2019 e includono Asia Pacifico (Hong Kong), Asia Pacific (Giacarta) Medio Oriente (Bahrein), Europa (Milano) e Africa (Città del Capo). Le regioni lanciate prima del 20 marzo 2019 sono abilitate per impostazione predefinita.

  Supporto per la consegna in più regioni a AWS Lambda

Tipo di consegna tra regioni	Supportato/Non supportato	
Regione abilitata per impostazione predefinita per la regione di attivazione	Non supportato	
Regione di attivazione per la regione abilitata per impostazione predefinita	Supportato utilizzando `sns.<opt-in-region>.amazonaws.com` nel principale di servizio per la funzione Lambda	
Regione di attivazione a regione di attivazione	Non supportato	

Stato di consegna dei messaggi Amazon SNS

Amazon SNS fornisce supporto per la registrazione dello stato di consegna dei messaggi di notifica inviati agli argomenti con i seguenti endpoint Amazon SNS:

- HTTP
- Amazon Data Firehose
- AWS Lambda
- Endpoint applicazione piattaforma
- Amazon Simple Queue Service

Dopo aver configurato gli attributi dello stato di consegna dei messaggi, le voci di registro vengono inviate ai CloudWatch registri per i messaggi inviati agli abbonati all'argomento. La registrazione dello stato di consegna dei messaggi consente di ottenere informazioni operative più precise, ad esempio:

- Sapere se un messaggio è stato consegnato all'endpoint Amazon SNS.
- Identificare la risposta inviata dall'endpoint Amazon SNS a Amazon SNS.
- Determinare il tempo di attesa del messaggio (il periodo di tempo tra il timestamp di pubblicazione e l'istante immediatamente precedente alla consegna a un endpoint Amazon SNS).

Per configurare gli attributi dell'argomento per lo stato di consegna dei messaggi, puoi utilizzare i AWS Management Console kit di sviluppo AWS software (SDK), l'API di interrogazione o. AWS CloudFormation

Argomenti

Configurazione della registrazione dello stato di consegna utilizzando la AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti), selezionare un argomento quindi scegliere Edit (Modifica).
4. Nella *MyTopic*pagina Modifica, espandi la sezione Registrazione dello stato di consegna.
5. Scegliere il protocollo per cui registrare lo stato di consegna, ad esempio AWS Lambda.
6. Inserisci la percentuale di campionamento di successo (la percentuale di messaggi riusciti per i quali desideri ricevere CloudWatch i registri).
7. Nella sezione IAM roles (Ruoli IAM) eseguire una delle operazioni seguenti:

   - Per scegliere un ruolo del servizio esistente dall'account, scegliere Use existing service role (Usa ruolo del servizio esistente) e quindi specificare i ruoli IAM per le consegne riuscite e non riuscite.

   - Per creare un nuovo ruolo del servizio nell'account, scegliere Create new service role (Crea nuovo ruolo del servizio), quindi Create new roles (Crea nuovi ruoli) per definire i ruoli IAM per le consegne riuscite e non riuscite nella console IAM.

     Per consentire ad Amazon SNS l'accesso in scrittura per utilizzare CloudWatch i log per tuo conto, scegli Consenti.
8. Seleziona Salvataggio delle modifiche.

   Ora puoi visualizzare e analizzare i CloudWatch log contenenti lo stato di consegna dei messaggi. Per ulteriori informazioni sull'utilizzo CloudWatch, consulta la CloudWatch documentazione.

Configurazione della registrazione dello stato di consegna tramite gli SDK AWS

Gli AWS SDK forniscono API in diverse lingue per l'utilizzo degli attributi dello stato di consegna dei messaggi con Amazon SNS.

Attributi di argomento

Puoi utilizzare i seguenti valori di nome di attributo di argomento per lo stato di consegna dei messaggi:

HTTP

- `HTTPSuccessFeedbackRoleArn`: indica lo stato di corretta consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint HTTP.
- `HTTPSuccessFeedbackSampleRate`: indica la percentuale di campionamenti di messaggi riusciti per un argomento Amazon SNS sottoscritto a un endpoint HTTP.
- `HTTPFailureFeedbackRoleArn`: indica lo stato di mancata consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint HTTP.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`: indica lo stato di corretta consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate`: indica la percentuale di campionamenti di messaggi riusciti per un argomento Amazon SNS sottoscritto a un endpoint Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn`: indica lo stato di mancata consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`: indica lo stato di corretta consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint Lambda.
- `LambdaSuccessFeedbackSampleRate`: indica la percentuale di campionamenti di messaggi riusciti per un argomento Amazon SNS sottoscritto a un endpoint Lambda.
- `LambdaFailureFeedbackRoleArn`: indica lo stato di mancata consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint Lambda.

Endpoint applicazione piattaforma

- `ApplicationSuccessFeedbackRoleArn`— Indica lo stato di corretta consegna dei messaggi per un argomento di Amazon SNS sottoscritto a un AWS endpoint applicativo.

- `ApplicationSuccessFeedbackSampleRate`— Indica la percentuale di messaggi riusciti da campionare per un argomento di Amazon SNS sottoscritto a un AWS endpoint applicativo.
- `ApplicationFailureFeedbackRoleArn`— Indica lo stato di invio non riuscito dei messaggi per un argomento di Amazon SNS sottoscritto a un AWS endpoint applicativo.

> **Note**
>
> Oltre a configurare attributi di argomento per lo stato di consegna dei messaggi di notifica inviati a endpoint applicazione Amazon SNS, puoi anche configurare attributi di applicazione per lo stato di consegna dei messaggi di notifica push inviati ai servizi di notifica push. Per ulteriori informazioni, consulta la pagina sull'utilizzo degli attributi di applicazione di Amazon SNS per lo stato di consegna dei messaggi.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`: indica lo stato di corretta consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint Amazon SQS.
- `SQSSuccessFeedbackSampleRate`: indica la percentuale di campionamenti di messaggi riusciti per un argomento Amazon SNS sottoscritto a un endpoint Amazon SQS.
- `SQSFailureFeedbackRoleArn`: indica lo stato di mancata consegna dei messaggi per un argomento Amazon SNS sottoscritto a un endpoint Amazon SQS.

> **Note**
>
> `<ENDPOINT>FailureFeedbackRoleArn`Gli attributi `<ENDPOINT>SuccessFeedbackRoleArn` and vengono utilizzati per consentire ad Amazon SNS l'accesso in scrittura per utilizzare CloudWatch i log per tuo conto. L'attributo `<ENDPOINT>SuccessFeedbackSampleRate` consente di specificare la percentuale della frequenza di campionamento (0-100) dei messaggi consegnati. Dopo aver configurato l'`<ENDPOINT>FailureFeedbackRoleArn`attributo, tutte le consegne di messaggi non riuscite generano log. CloudWatch

AWS Esempi SDK per configurare gli attributi degli argomenti

I seguenti esempi di codice mostrano come utilizzare`SetTopicAttributes`.

CLI

AWS CLI

Impostazione di un attributo per un argomento

Nell'esempio `set-topic-attributes` seguente vengono impostati gli attributi `DisplayName` per l'argomento specificato.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta SetTopicAttributes AWS CLICommand Reference.

Java

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.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, SetTopicAttributesconsulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, SetTopicAttributesconsulta AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?)
 {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Per i dettagli sull'API, SetTopicAttributesconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */
```

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, SetTopicAttributesconsulta AWS SDK for PHP API Reference.

Ruby

SDK per Ruby

> **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.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
```

```
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
#{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                    Sid: "__default_statement_ID",
                    Effect: "Allow",
                    Principal: { "AWS": "*" },
                    Action: ["SNS:Publish"],
                    Resource: topic_arn,
                    Condition: {
                      ArnEquals: {
                        "AWS:SourceArn": resource_arn
                      }
```

```
                    }
                  }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, SetTopicAttributesconsulta AWS SDK for Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value
        ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
```

```
    ENDTRY.
```

- Per i dettagli sulle API, SetTopicAttributesconsulta AWS SDK for SAP ABAP API reference.

Configurazione della registrazione dello stato di consegna utilizzando AWS CloudFormation

Per configurare `DeliveryStatusLogging` l'utilizzo AWS CloudFormation, utilizza un modello JSON o YAML per creare uno stack. AWS CloudFormation Per ulteriori informazioni, consultate la `DeliveryStatusLogging` proprietà della `AWS::SNS::Topic` risorsa nella Guida per l'utente. AWS CloudFormation Di seguito sono riportati alcuni esempi di AWS CloudFormation modelli in JSON e YAML per creare un nuovo argomento o aggiornare un argomento esistente con tutti `DeliveryStatusLogging` gli attributi per il protocollo Amazon SQS.

JSON

```
"Resources": {
    "MySNSTopic" : {
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "TestTopic",
            "DisplayName" : "TEST",
            "SignatureVersion" : "2",
            "DeliveryStatusLogging" : [{
                "Protocol": "sqs",
                "SuccessFeedbackSampleRate": "45",
                "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
                "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
            }]
        }
    }
}
```

YAML

```
Resources:
```

```
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName:TestTopic
      DisplayName:TEST
      SignatureVersion:2
      DeliveryStatusLogging:
       - Protocol: sqs
         SuccessFeedbackSampleRate: 45
         SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
         FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Tentativi di consegna dei messaggi di Amazon SNS

Amazon SNS definisce una policy di consegna per ogni protocollo di consegna. La policy di consegna definisce il modo in cui Amazon SNS esegue nuovi tentativi di consegna dei messaggi quando si verificano errori sul lato server (quando il sistema che ospita l'endpoint sottoscritto diventa non disponibile). Quando la politica di spedizione è esaurita, Amazon SNS smette di tentare nuovamente la consegna e scarterà il messaggio, a meno che all'abbonamento non sia allegata una coda di lettere non recapitate. Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS.

Argomenti

- Protocolli e policy di consegna
- Fasi della policy di consegna
- Creazione di una policy di consegna HTTP/S

Protocolli e policy di consegna

> **Note**
>
> - Ad eccezione di HTTP/S, non è possibile modificare le policy di consegna definite da Amazon SNS. Solo HTTP/S supporta policy personalizzate. Per informazioni, consulta Creazione di una policy di consegna HTTP/S.

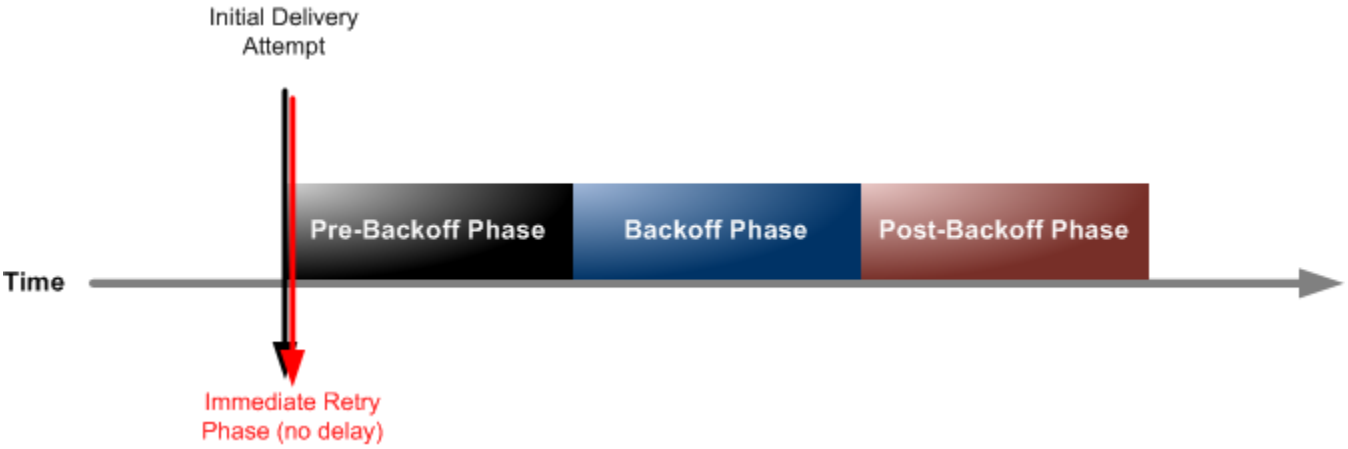
- Amazon SNS applica il jittering ai tentativi di consegna. Per ulteriori informazioni, vedere il post Backoff esponenziale e Jitter in Blog architettura AWS .
- Il tempo totale di ripetizione delle policy per un endpoint HTTP/S non può essere superiore a 3.600 secondi. Questo è un limite fissato che non può essere modificato.

Tipo di endpoint	Protocolli di consegna	Fase dei nuovi tentativi immediati (nessun ritardo)	Fase di prebackoff	Fase di backoff	Fase di postbackoff	Totale tentativi
AWS endpoint gestiti	Amazon Data Firehose[1]	3 volte, senza ritardo	2 volte, 1 secondo di distanza	10 volte, con backoff esponenziale, da 1 secondo a 20 secondi	100.000 volte, 20 secondi di distanza	100.015 volte, oltre 23 giorni
	AWS Lambda					
	Amazon SQS					
Endpoint gestiti dal cliente	SMTP	0 volte, senza ritardo	2 volte, 10 secondi di distanza	10 volte, con backoff esponenziale, da 10 secondi a 600 secondi (10 minuti)	38 volte, 600 secondi (10 minuti) di distanza	50 tentativi, oltre 6 ore
	SMS					
	Push per dispositivi mobili					

[1] Per gli errori di limitazione con il protocollo Firehose, Amazon SNS utilizza la stessa politica di distribuzione degli endpoint gestiti dai clienti.

Fasi della policy di consegna

Il diagramma seguente mostra le fasi di una policy di consegna.



Ogni policy di consegna si compone di quattro fasi.

1. Fase di riprova immediata (Nessun ritardo) – Questa fase si verifica immediatamente dopo il tentativo di consegna iniziale. Non c'è ritardo tra i nuovi tentativi in questa fase.
2. Fase di prebackoff – Questa fase segue la fase dei nuovi tentativi senza ritardo. Amazon SNS utilizza questa fase per effettuare una serie di tentativi prima di applicare una funzione di backoff. Questa fase specifica il numero di tentativi e la quantità di ritardo tra di loro.
3. Fase di backoff – Questa fase controlla il ritardo tra tentativi utilizzando la funzione retry-backoff. Questa fase imposta un ritardo minimo, un ritardo massimo e una funzione retry-backoff che definisce la velocità con cui il ritardo aumenta da minimo a massimo. La funzione di backoff può essere aritmetica, esponenziale, geometrica o lineare.
4. Fase di postbackoff – Questa fase segue la fase di backoff. Specifica un certo numero di tentativi e la quantità di ritardo tra di loro. Questa è la fase finale.

Creazione di una policy di consegna HTTP/S

Puoi utilizzare una policy di consegna e le sue quattro fasi per definire la modalità in cui Amazon SNS effettua nuovi tentativi di consegna dei messaggi agli endpoint HTTP/S. Amazon SNS ti consente di ignorare la policy predefinita per i nuovi tentativi per gli endpoint HTTP quando, ad esempio, desideri personalizzare la policy in base alla capacità del server HTTP.

È possibile impostare le policy di consegna HTTP/S come oggetto JSON a livello di sottoscrizione o argomento. Quando si definisce la policy a livello di argomento, essa si applica a tutte le

sottoscrizioni HTTP/S associate all'argomento. Per impostare la policy di distribuzione a livello di abbonamento, puoi utilizzare l'azione API `Subscribe` o `SetSubscriptionAttributes`. Per impostare la policy di distribuzione a livello di argomento, puoi utilizzare l'azione API `CreateTopic` o `SetTopicAttributes`. In alternativa, puoi anche utilizzare la risorsa nei tuoi modelli. `AWS::SNS::Subscription` AWS CloudFormation

È consigliabile personalizzare le policy di consegna in base alla capacità del server HTTP/S. È possibile impostare la policy come attributo argomento o attributo di sottoscrizione. Se tutte le sottoscrizioni HTTP/S nell'argomento sono destinate allo stesso server HTTP/S, si consiglia di impostare la policy di consegna come attributo argomento, in modo che rimanga valida per tutte le sottoscrizioni HTTP/S nell'argomento. In caso contrario, è necessario comporre una policy di consegna per ogni sottoscrizione HTTP/S nell'argomento, in base alla capacità del server HTTP/S cui la policy è destinata.

Inoltre, puoi impostare l'intestazione Content-Type nella policy di richiesta per specificare il tipo di supporto della notifica. Per impostazione predefinita, Amazon SNS invia tutte le notifiche agli endpoint HTTP/S con il tipo di contenuto impostato su `text/plain; charset=UTF-8`. Amazon SNS ti consente di ignorare la policy di richiesta predefinita. Per `headerContentType` supportata e i vincoli, consulta la tabella sottostante.

L'oggetto JSON seguente rappresenta una policy di consegna che indica a Amazon SNS di ripetere un tentativo di consegna HTTP/S non riuscito, come segue:

1. 3 volte immediatamente nella fase nessun ritardo
2. 2 volte (1 secondo di distanza) nella fase di pre-backoff
3. 10 volte (con backoff esponenziale da 1 secondo a 60 secondi)
4. 35 volte (60 secondi di distanza) nella fase di pre-backoff

In questa policy di consegna di esempio, Amazon SNS esegue un totale di 50 tentativi prima di eliminare il messaggio. Per mantenere il messaggio dopo che i tentativi specificati nelle policy di consegna sono esauriti, configurare la sottoscrizione per spostare i messaggi non recapitabili in una coda di lettere non recapitabili (DLQ). Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS.

> **Note**
>
> Questa policy di consegna indica inoltre a Amazon SNS di limitare le consegne a non più di 10 al secondo, usando la `maxReceivesPerSecond` proprietà. Questa velocità di limitazione

automatica potrebbe comportare la pubblicazione di un numero maggiore di messaggi (traffico in entrata) rispetto a quelli recapitati (traffico in uscita). Quando c'è più traffico in ingresso rispetto a quello in uscita, la sottoscrizione può accumulare un backlog di messaggi di grandi dimensioni, che potrebbe causare un'elevata latenza di recapito dei messaggi. Nelle policy di consegna, assicurati di specificare un valore per `maxReceivesPerSecond` che non influisce negativamente sul carico di lavoro.

Note

Questa policy di distribuzione sostituisce il tipo di contenuto predefinito per la notifica HTTP/S in `application/json`.

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

La policy di distribuzione è composta da una policy per nuovi tentativi, una policy di limitazione (della larghezza di banda della rete) e una policy di richiesta. In totale, sono disponibili nove attributi in una policy di distribuzione.

Policy	Descrizione	Vincolo
`minDelayTarget`	Il ritardo minimo per un nuovo tentativo. Unità: secondi	1 al massimo ritardo Di default: 20
`maxDelayTarget`	Il ritardo massimo per un nuovo tentativo. Unità: secondi	Ritardo minimo su 3.600 Di default: 20
`numRetries`	Il numero totale di tentativi, inclusi tentativi immediati, pre-backoff, backoff e post-back off.	Da 0 a 100 Di default: 3
`numNoDelayRetries`	Il numero di tentativi da effettuare immediatamente, senza ritardi tra essi.	Uguale o maggiore di 0 Di deafult: 0
`numMinDelayRetries`	Il numero di tentativi nella fase di pre-backoff, con il ritardo minimo specificato tra essi.	Uguale o maggiore di 0 Di default: 0
`numMaxDelayRetries`	Il numero di tentativi nella fase post-backoff, con il massimo ritardo tra essi.	Uguale o maggiore di 0 Di default: 0
`backoffFunction`	Il modello per il backoff tra i nuovi tentativi.	Una delle quattro opzioni: • Aritmetica • Esponenziale • Geometrica • Lineare Di default: lineare

Policy	Descrizione	Vincolo
`maxReceivesPerSecond`	Numero massimo di consegne al secondo, per sottoscrizione.	Uguale o maggiore di 1 Di default: nessuna limitazione

Policy	Descrizione	Vincolo
`headerContentType`	Il tipo di contenuto della notifica inviato agli endpoint HTTP/S.	Se la policy di richiesta non è definita, il tipo di contenuto è preimpostato su `text/plain; charset=UTF-8` . Quando la distribuzione di messaggi non elaborati è disabilitata per una sottoscrizione (impostazione predefinita) o quando la policy di distribuzione è definita a livello di argomento, i tipi di contenuto dell'intestazione supportati sono `application/json` e `text/plain` . Quando la distribuzione di messaggi non elaborati è abilitata per una sottoscrizione, sono supportati i seguenti tipi di contenuto: • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • applicazione/ x-www-form-urlencoded

Policy	Descrizione	Vincolo
		• application/xhtml+xml • application/xml

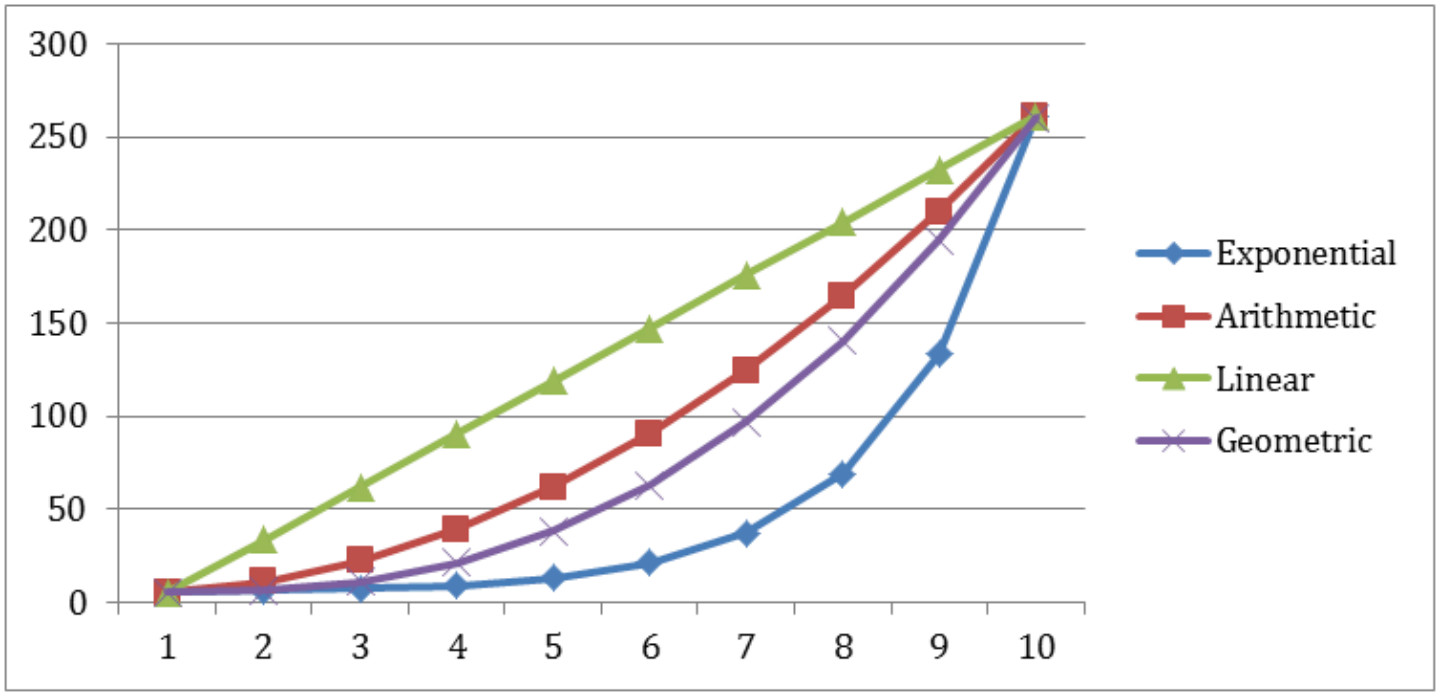
Amazon SNS utilizza la formula seguente per calcolare il numero di nuovi tentativi nella fase di backoff:

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

È possibile utilizzare tre parametri per controllare la frequenza dei nuovi tentativi nella fase di backoff.

- `minDelayTarget` – Definisce il ritardo associato al primo tentativo nella fase di backoff.
- `maxDelayTarget` – Definisce il ritardo associato al nuovo tentativo finale nella fase di backoff.
- `backoffFunction` – Definisce l'algoritmo utilizzato da Amazon SNS per calcolare i ritardi associati a tutti i nuovi tentativi eseguiti tra il primo e l'ultimo nella fase di backoff. È possibile utilizzare una delle quattro funzioni retry-backoff.

Il diagramma seguente mostra come ogni funzione di backoff di nuovi tentativi influisce sul ritardo associato ai tentativi durante la fase di backoff: una policy di consegna con il numero totale di tentativi impostato su 10, il ritardo minimo impostato su 5 secondi e il ritardo massimo impostato su 260 secondi. L'asse verticale rappresenta il ritardo in secondi associato a ciascuno dei 10 tentativi. L'asse orizzontale rappresenta il numero di tentativi, dal primo al decimo tentativo.



Code DLQ (DLQ) Amazon SNS

Una coda DLQ è una coda Amazon SQS a cui un abbonamento Amazon SNS può mirare per i messaggi che non possono essere recapitati correttamente agli abbonati. I messaggi che non possono essere recapitati a causa di errori client o errori server vengono mantenuti nella coda DLQ per ulteriori analisi o elaborazione. Per ulteriori informazioni, consulta Configurazione di una coda Amazon SNS dead-letter per una sottoscrizione e Tentativi di consegna dei messaggi di Amazon SNS.

Note

- L'abbonamento Amazon SNS e la coda Amazon SQS devono essere sotto lo stesso account AWS e regione.
- Per un Argomento FIFO, puoi utilizzare una coda di Amazon SQS come una coda DLQ per la sottoscrizione ad Amazon SNS. Le sottoscrizioni agli argomenti FIFO utilizzano le code FIFO e le sottoscrizioni agli argomenti standard utilizzano le code standard.
- Per utilizzare una coda Amazon SQS crittografata come coda DLQ, devi utilizzare una chiave KMS personalizzata con una policy delle chiavi che conceda al principale di servizio Amazon SNS l'accesso a operazioni API AWS KMS. Per ulteriori informazioni, consulta

Crittografia a riposo in questa guida e Protezione dei dati Amazon SQS con la crittografia lato server (SSE) e AWS KMS nella Guida per sviluppatori Amazon Simple Queue Service.

Argomenti

Perché le consegne dei messaggi non riescono?

In generale, la consegna dei messaggi non riesce quando Amazon SNS non può accedere a un endpoint sottoscritto a causa di un errore lato client o lato server. Quando Amazon SNS riceve un errore sul lato client o continua a ricevere un errore sul lato server per un messaggio che supera il numero di tentativi specificato dalla policy di ripetizione dei tentativi corrispondente, Amazon SNS ignora il messaggio, a meno che non sia allegata una coda DLQ all'abbonamento. Le consegne non riuscite non modificano lo stato delle sottoscrizioni. Per ulteriori informazioni, consulta Tentativi di consegna dei messaggi di Amazon SNS.

Errori lato client

Gli errori lato client possono verificarsi quando Amazon SNS dispone di metadati di sottoscrizione obsoleti. Questi errori si verificano in genere quando un proprietario elimina l'endpoint (ad esempio, una funzione Lambda sottoscritta a un argomento Amazon SNS) o quando un proprietario modifica la policy associata all'endpoint sottoscritto in modo da impedire ad Amazon SNS la policy di consegna dei messaggi all'endpoint. Amazon SNS non riprova il recapito dei messaggi che non riesce a causa di un errore sul lato client.

Errori lato server

Gli errori lato server possono verificarsi quando il sistema responsabile dell'endpoint sottoscritto diventa non disponibile o restituisce un'eccezione che indica che non è in grado di elaborare una richiesta valida da Amazon SNS. Quando si verificano errori lato server, Amazon SNS ripropone le consegne non riuscite utilizzando una funzione di backoff lineare o esponenziale. Per errori sul lato

server causati da endpoint AWS gestiti supportati da Amazon SQS o AWS Lambda, Amazon SNS ritira la consegna fino a 100.015 volte, in 23 giorni.

Gli endpoint gestiti dal cliente (ad esempio HTTP, SMTP, SMS o push mobile) possono anche causare errori lato server. Amazon SNS tenta di eseguire nuovamente la consegna anche a questi tipi di endpoint. Mentre gli endpoint HTTP supportano le policy di nuovi tentativi definite dal cliente, Amazon SNS imposta una policy di nuovi tentativi di consegna interna su 50 volte nell'arco di 6 ore per gli endpoint push SMTP, SMS e dispositivi mobili.

Come funzionano le code DLQ?

Una coda dead-letter è allegata a una sottoscrizione Amazon SNS (anziché a un argomento) perché la consegna dei messaggi avviene a livello di sottoscrizione. In questo modo è possibile identificare più facilmente l'endpoint di destinazione originale per ogni messaggio.

Una coda dead-letter associata a una sottoscrizione Amazon SNS è una coda Amazon SQS ordinaria. Per ulteriori informazioni sul periodo di conservazione dei messaggi, consulta Quote correlate ai messaggi nella Guida per sviluppatori Amazon Simple Queue Service. È possibile modificare il periodo di conservazione dei messaggi utilizzando l'operazione API Amazon SQS `SetQueueAttributes`. Per rendere le applicazioni più resilienti, è consigliabile impostare il periodo massimo di conservazione per le code dead-letter a 14 giorni.

Come vengono spostati i messaggi in una coda DLQ?

I messaggi vengono spostati in una coda dead-letter utilizzando una policy di redrive. Una policy di redrive è un oggetto JSON che fa riferimento all'ARN della coda dead-letter. L'attributo `deadLetterTargetArn` specifica l'ARN. L'ARN deve puntare a una coda Amazon SQS nello stesso Account AWS e nella stessa regione della sottoscrizione Amazon SNS. Per ulteriori informazioni, consulta Configurazione di una coda Amazon SNS dead-letter per una sottoscrizione.

L'oggetto JSON seguente è una policy di redrive di esempio, allegato a una sottoscrizione SNS.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Come posso spostare i messaggi fuori da una coda DLQ?

È possibile spostare i messaggi fuori da una coda dead-letter in due modi:

- Evitare di scrivere logica consumer Amazon SQS - Impostare la coda dead-letter come origine evento sulla funzione Lambda per drenare la coda dead-letter.
- Evitare di scrivere la logica consumer Amazon SQS - Utilizzare l'API Amazon SQS, SDK AWS o AWS CLI per scrivere la logica consumer personalizzata per il polling, l'elaborazione e l'eliminazione dei messaggi nella coda dead-letter.

Come posso monitorare e registrare code DLQ?

È possibile utilizzare i parametri Amazon CloudWatch per monitorare le code dead-letter associate alle sottoscrizioni Amazon SNS. Tutte le code Amazon SQS emettono le metriche CloudWatch a intervalli di un minuto. Per ulteriori informazioni, consulta Metriche disponibili CloudWatch per Amazon SQS nella Guida per sviluppatori Amazon Simple Queue Service. Anche tutte le sottoscrizioni Amazon SNS con code DLQ emettono parametri CloudWatch. Per ulteriori informazioni, consulta Monitoraggio di argomenti Amazon SNS tramite CloudWatch.

Per ricevere una notifica dell'attività nelle code dead-letter, è possibile utilizzare allarmi e parametri CloudWatch. Ad esempio, quando si prevede che la coda dead-letter sia sempre vuota, è possibile creare un avviso CloudWatch per il parametro `NumberOfMessagesSent`. È possibile impostare la soglia di allarme su 0 e specificare un argomento Amazon SNS da notificare quando l'avviso si spegne. In questo argomento Amazon SNS è possibile inviare la notifica di allarme a qualsiasi tipo di endpoint (ad esempio un indirizzo e-mail, un numero di telefono o un'app per il cercapersone mobile).

È possibile utilizzare CloudWatch Logs per esaminare le eccezioni che causano qualsiasi esito negativo delle consegne Amazon SNS e dell'invio di messaggi alle code dead-letter. Amazon SNS può registrare sia le consegne riuscite che quelle non riuscite in CloudWatch. Per ulteriori informazioni, consulta Stato di consegna dei messaggi Amazon SNS.

Configurazione di una coda Amazon SNS dead-letter per una sottoscrizione

Una coda DLQ è una coda Amazon SQS a cui un abbonamento Amazon SNS può mirare per i messaggi che non possono essere recapitati correttamente agli abbonati. I messaggi che non possono essere recapitati a causa di errori client o errori server vengono mantenuti nella coda DLQ per ulteriori analisi o elaborazione. Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS e Tentativi di consegna dei messaggi di Amazon SNS.

Questa pagina mostra come utilizzare AWS Management Console, un AWS SDK AWS CLI, e configurare una coda di lettere non scritte AWS CloudFormation per un abbonamento Amazon SNS.

> **Note**
>
> Per un Argomento FIFO, puoi utilizzare una coda di Amazon SQS come una coda DLQ per la sottoscrizione ad Amazon SNS. Le sottoscrizioni agli argomenti FIFO utilizzano le code FIFO e le sottoscrizioni agli argomenti standard utilizzano le code standard.

Prerequisiti

Prima di configurare una coda DLQ, completare i seguenti prerequisiti:

1. Creare un argomento Amazon SNS, denominato `MyTopic`.
2. Crear una coda Amazon SQS denominata `MyEndpoint`, da utilizzare come endpoint per l'abbonamento Amazon SNS.
3. (Salta per AWS CloudFormation) Iscriviti alla coda per accedere all'argomento.
4. Creare un'altra coda Amazon SQS denominata `MyDeadLetterQueue`, da utilizzare come coda dead-letter per la sottoscrizione Amazon SNS.
5. Per concedere l'accesso Amazon SNS principale all'azione Amazon SQS API, impostare la policy della coda seguente per `MyDeadLetterQueue`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
    }
  }]
}
```

Argomenti

- Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando il AWS Management Console
- Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando un SDK AWS
- Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando il AWS CLI
- Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando AWS CloudFormation

Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando il AWS Management Console

Prima di iniziare questo tutorial, completare i prerequisiti descritti di seguito.

1. Accedere alla console Amazon SQS.
2. Creare una coda Amazon SQS o utilizzare una coda esistente e prendere nota dell'ARN della coda nella scheda Details (Dettagli) della coda, ad esempio:

   ```
   arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
   ```

3. Accedi alla console Amazon SNS.
4. Nel riquadro di navigazione, scegli Sottoscrizioni.
5. Sulla pagina Subscriptions (Abbonamenti), selezionare una sottoscrizione esistente, quindi scegliere Edit (Modifica).
6. Nella pagina Modificare ***1234a567-bc89-012d-3e45-6fg7h890123i***, espandere la pagina Policy di redrive (coda DLQ) e procedere come segue:

   a. Scegli Enabled (Abilitato).

   b. Specificare l'ARN di una coda Amazon SQS.

7. Seleziona Save changes (Salva modifiche).

   La sottoscrizione è configurata per l'utilizzo di una coda dead-letter.

Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando un SDK AWS

Prima di eseguire questo esempio, completare i prerequisiti.

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

Il seguente esempio di codice mostra come utilizzare. `SetSubscriptionAttributesRedrivePolicy`

Java

SDK per Java 1.x

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando il AWS CLI

Prima di iniziare questo tutorial, completare i prerequisiti descritti di seguito.

1. Istalla e configura la AWS CLI. Per ulteriori informazioni, consulta la AWS Command Line Interface Guida per l'utente.
2. Utilizza il seguente comando.

```
aws sns set-subscription-attributes \
--subscription-arn arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i
--attribute-name RedrivePolicy
--attribute-value "{\"deadLetterTargetArn\": \"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}"
```

Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando AWS CloudFormation

Prima di iniziare questo tutorial, completare i prerequisiti.

1. Copia il seguente codice JSON in un file denominato `MyDeadLetterQueue.json`.

```
{
  "Resources": {
    "mySubscription": {
      "Type" : "AWS::SNS::Subscription",
      "Properties" : {
        "Protocol": "sqs",
        "Endpoint": "arn:aws:sqs:us-east-2:123456789012:MyEndpoint",
        "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
        "RedrivePolicy": {
          "deadLetterTargetArn":
            "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
        }
      }
    }
  }
}
```

2. Accedi alla console AWS CloudFormation.
3. Nella pagina Select Template (Seleziona modello) scegliere Upload a template to Amazon S3 (Carica un modello in Amazon S3), selezionare il file `MyDeadLetterQueue.json`, quindi scegliere Next (Avanti).
4. Nella pagina Specify Details (Specifica dettagli), digitare `MyDeadLetterQueue` per Stack Name (Nome stack), quindi scegliere Next (Avanti).
5. Nella pagina Opzioni, scegli Next (Avanti).
6. Nella pagina Revisione scegli Create (Crea).

   AWS CloudFormation inizia a creare lo **`MyDeadLetterQueue`** stack e visualizza lo stato CREATE_IN_PROGRESS. Quando il processo è completo, visualizza lo stato CREATE_COMPLETE. AWS CloudFormation

Archiviazione, riproduzione e analisi dei messaggi di Amazon SNS

Gli argomenti standard di Amazon SNS supportano l'archiviazione dei messaggi tramite Amazon Data Firehose. È possibile inviare notifiche ai flussi di distribuzione di Firehose, che consentono di inviare notifiche alle destinazioni di archiviazione e analisi supportate da Firehose, tra cui Amazon Simple Storage Service (Amazon S3), Amazon Redshift e altre.

Gli argomenti FIFO di Amazon SNS supportano un archivio di messaggi locale e senza codice che consente ai proprietari degli argomenti di archiviare i messaggi pubblicati su un argomento per un massimo di 365 giorni. Per gli argomenti con una `ArchivePolicy` attiva, gli abbonati possono quindi creare una `ReplayPolicy` per recuperare (o riprodurre) i messaggi archiviati su un endpoint sottoscritto. Per ulteriori informazioni su questa funzionalità, consulta Archiviazione e riproduzione dei messaggi per argomenti FIFO.

Funzionalità	Argomenti standard	Argomenti FIFO
Archiviazione di messaggi	Flussi di distribuzione da Fanout a Firehose	Archiviazione dei messaggi per i proprietari di argomenti FIFO
Riproduzione di messaggi	La riproduzione per argomenti standard non è una funzional ità integrata. Molti clienti ne creano una propria in base al proprio archivio di messaggi.	Riproduzione dei messaggi per gli abbonati all'argomento FIFO

Using Amazon SNS per la messaggistica da applicazione a applicazione (A2A)

Questa sezione fornisce informazioni sull'utilizzo di Amazon SNS per la messaggistica applicazione-applicazione con gli abbonati.

Argomenti

Flussi di distribuzione da Fanout a Firehose

Puoi abbonare i flussi di distribuzione di Amazon Data Firehose agli argomenti di Amazon SNS, il che ti consente di inviare notifiche a endpoint di storage e analisi aggiuntivi. I messaggi pubblicati su un argomento di Amazon SNS vengono inviati al flusso di distribuzione Firehose sottoscritto e consegnati alla destinazione come configurato in Firehose. Il titolare di un abbonamento può abbonare fino a cinque stream di distribuzione Firehose a un argomento Amazon SNS. Ogni flusso di distribuzione Firehose ha una quota predefinita per le richieste e la velocità effettiva al secondo. Questa limitazione potrebbe comportare un numero maggiore di messaggi pubblicati (traffico in entrata) rispetto a quelli consegnati (traffico in uscita). Quando c'è più traffico in ingresso rispetto a quello in uscita, la sottoscrizione può accumulare un backlog di messaggi di grandi dimensioni, che potrebbe causare un'elevata latenza di recapito dei messaggi. Puoi richiedere un aumento della quota in base alla percentuale di pubblicazione per evitare impatti negativi sul carico di lavoro.

Tramite i flussi di distribuzione Firehose, puoi inviare le notifiche di Amazon SNS ad Amazon Simple Storage Service (Amazon S3), Amazon Redshift, OpenSearch Amazon Service (Service) e a fornitori di servizi di terze parti come Datadog, New Relic, MongoDB e Splunk. OpenSearch

Ad esempio, puoi utilizzare questa funzionalità per archiviare in modo permanente i messaggi inviati a un argomento in un bucket Amazon S3 a fini di conformità, archiviazione o altri scopi. A tale scopo, crea un flusso di distribuzione Firehose con una destinazione per bucket S3 e sottoscrivi tale flusso

di distribuzione all'argomento Amazon SNS. Come altro esempio, per eseguire analisi sui messaggi inviati a un argomento Amazon SNS, crea un flusso di distribuzione con una destinazione dell'indice di OpenSearch servizio. Puoi quindi sottoscrivere lo stream di distribuzione di Firehose all'argomento Amazon SNS.

Amazon SNS supporta anche la registrazione dello stato di consegna dei messaggi per le notifiche inviate agli endpoint Firehose. Per ulteriori informazioni, consulta Stato di consegna dei messaggi Amazon SNS.

Argomenti

Prerequisiti per la sottoscrizione dei flussi di distribuzione di Firehose agli argomenti di Amazon SNS

Per sottoscrivere uno stream di distribuzione di Amazon Data Firehose a un argomento SNS, devi avereAccount AWS:

- Un argomento SNS standard. Per ulteriori informazioni, consulta Creare un argomento Amazon SNS.
- Un flusso di distribuzione Firehose. Per ulteriori informazioni, consulta Creare un flusso di distribuzione Amazon Data Firehose e concedere all'applicazione l'accesso alle risorse Firehose nella Amazon Data Firehose Developer Guide.
- Un ruolo (IAM) AWS Identity and Access Management che considera attendibile l'entità servizio Amazon SNS e dispone dell'autorizzazione a scrivere nel flusso di distribuzione. Inserisci l'Amazon Resource Name (ARN) di questo ruolo come `SubscriptionRoleARN` quando crei la sottoscrizione. Amazon SNS assume questo ruolo, che consente ad Amazon SNS di inserire record nel flusso di distribuzione Firehose.

  Di seguito viene illustrato un esempio di policy con autorizzazioni suggerite:

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Action": [
        "firehose:DescribeDeliveryStream",
        "firehose:ListDeliveryStreams",
        "firehose:ListTagsForDeliveryStream",
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:us-east-1:111111111111:deliverystream/firehose-sns-
delivery-stream"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Per fornire l'autorizzazione completa all'utilizzo di Firehose, puoi anche utilizzare la policy AWS gestita. `AmazonKinesisFirehoseFullAccess` In alternativa, per fornire autorizzazioni più rigorose per l'utilizzo di Firehose, è possibile creare una politica personalizzata. Come minimo, la policy deve dare l'autorizzazione per eseguire l'operazione `PutRecord` su un flusso di consegna specifico.

In tutti i casi, bisogna anche modificare la relazione di fiducia per includere il servizio principale Amazon SNS. Per esempio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Per ulteriori informazioni sulla creazione di un ruolo, consulta Creazione di un ruolo per delegare le autorizzazioni a un servizio AWS nella Guida per l'utente di IAM.

Dopo aver completato questi requisiti, puoi sottoscrivere il flusso di consegna all'argomento SNS.

Sottoscrizione di uno stream di distribuzione Firehose a un argomento di Amazon SNS

Per inviare le notifiche di Amazon SNS ai flussi di distribuzione di Amazon Data Firehose, assicurati innanzitutto di aver soddisfatto tutti i prerequisiti. Per un elenco degli endpoint supportati, consulta la sezione Endpoint e quote di Amazon Data Firehose nel. Riferimenti generali di Amazon Web Services

Per abbonare uno stream di distribuzione di Firehose a un argomento

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione scegli Subscriptions (Sottoscrizioni).
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
    a. Per Topic ARN (ARN argomento) scegli l'Amazon Resource Name (ARN) di un argomento standard.
    b. Per Protocollo, scegliete Firehose.
    c. Per Endpoint, scegli l'ARN di un flusso di distribuzione Firehose in grado di ricevere notifiche da Amazon SNS.
    d. Per il ruolo di sottoscrizione ARN, specificate l'ARN del ruolo AWS Identity and Access Management (IAM) creato per la scrittura nei flussi di distribuzione Firehose. Per ulteriori informazioni, consulta Prerequisiti per la sottoscrizione dei flussi di distribuzione di Firehose agli argomenti di Amazon SNS.
    e. (Facoltativo) Per rimuovere i metadati di Amazon SNS dai messaggi pubblicati, scegli Enable raw message delivery (Abilitaza il recapito di messaggi). Per ulteriori informazioni, consulta Consegna di messaggi non elaborati Amazon SNS.
5. (Facoltativo) Per configurare un criterio di filtro, espandere la sezione policy di filtro della sottoscrizione. Per ulteriori informazioni, consulta Policy di filtro degli abbonamenti Amazon SNS.

6. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS.
7. Scegli Create Subscription (Crea sottoscrizione).

La console crea la sottoscrizione e apre la pagina dettagli della sottoscrizione.

Utilizzo delle destinazioni del flusso di distribuzione

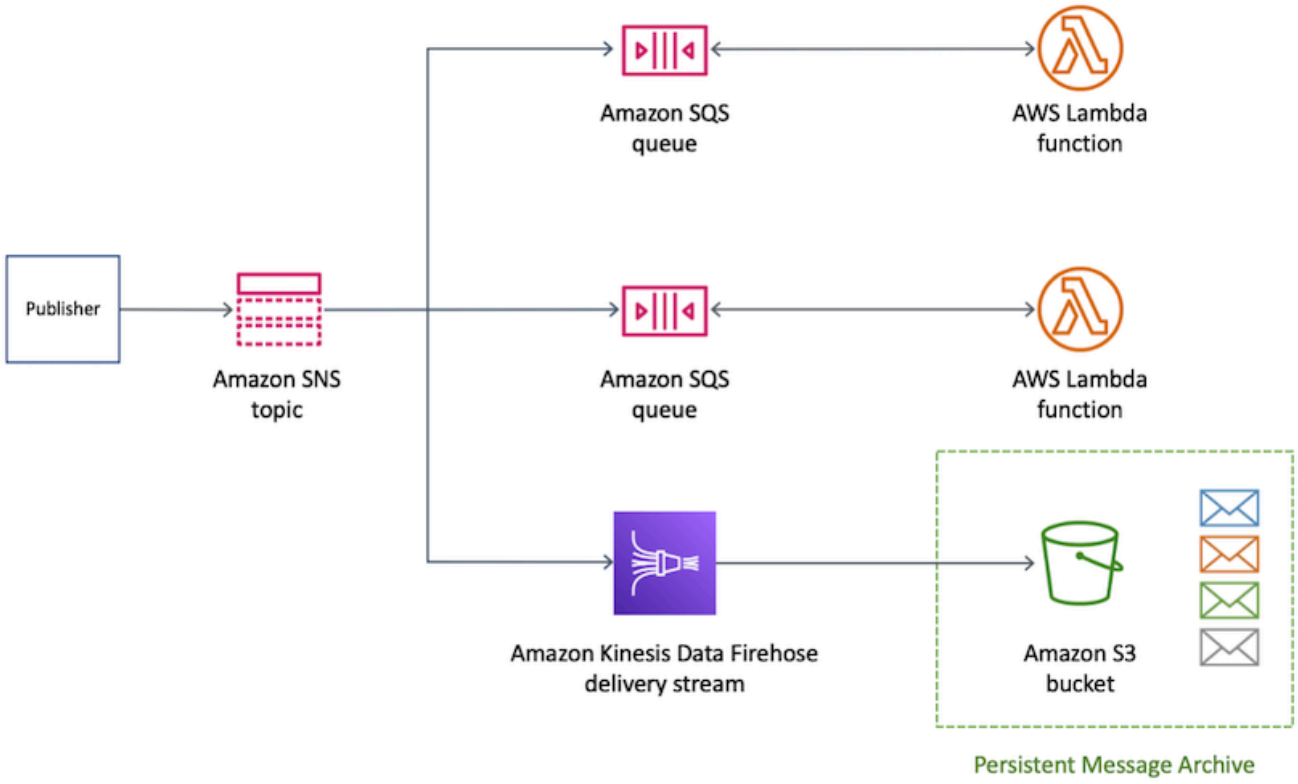
Tramite i flussi di distribuzione di Amazon Data Firehose, puoi inviare messaggi a endpoint aggiuntivi. In questa sezione viene descritto come lavorare con le destinazioni supportate.

Argomenti

- Destinazioni di Amazon S3
- OpenSearch Destinazioni di servizio
- Destinazioni Amazon Redshift
- Destinations HTTP

Destinazioni di Amazon S3

Questa sezione fornisce informazioni sui flussi di distribuzione di Amazon Data Firehose che pubblicano dati su Amazon Simple Storage Service (Amazon S3).



Argomenti

- Formato dei messaggi archiviati per le destinazioni Amazon S3
- Analisi dei messaggi per le destinazioni Amazon S3

Formato dei messaggi archiviati per le destinazioni Amazon S3

L'esempio seguente illustra una notifica Amazon SNS inviata a un bucket Amazon Simple Storage Service (Amazon S3), utilizzando i rientri per la leggibilità.

> **Note**
>
> In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito dei messaggi non elaborati è diasbilitato, Amazon SNS aggiunge i metadati JSON al messaggio, incluse le seguenti proprietà:
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- `Subject`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Per ulteriori informazioni sulla distribuzione non elaborata, consulta Consegna di messaggi non elaborati Amazon SNS.

```
{
    "Type": "Notification",
    "MessageId": "719a6bbf-f51b-5320-920f-3385b5e9aa56",
    "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-kinesis-test-topic",
    "Subject": "My 1st subject",
    "Message": "My 1st body",
    "Timestamp": "2020-11-26T23:48:02.032Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:333333333333:my-kinesis-test-
topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5",
    "MessageAttributes": {
        "myKey1": {
            "Type": "String",
            "Value": "myValue1"
        },
        "myKey2": {
            "Type": "String",
            "Value": "myValue2"
        }
    }
 }
```

L'esempio seguente mostra tre messaggi SNS inviati tramite un flusso di distribuzione di Amazon Data Firehose allo stesso bucket Amazon S3. Il buffer viene preso in considerazione e le interruzioni di riga separano i messaggi.

```
{"Type":"Notification","MessageId":"d7d2513e-6126-5d77-
bbe2-09042bd0a03a","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 1st subject","Message":"My 1st
 body","Timestamp":"2020-11-27T00:30:46.100Z","UnsubscribeURL":"https://
```

```
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":
{"Type":"String","Value":"myValue1"},"myKey2":{"Type":"String","Value":"myValue2"}}}
{"Type":"Notification","MessageId":"0c0696ab-7733-5bfb-b6db-
ce913c294d56","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 2nd subject","Message":"My 2nd
 body","Timestamp":"2020-11-27T00:31:22.151Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":{"Type":"String","Value":"myValue1"}}}
{"Type":"Notification","MessageId":"816cd54d-8cfa-58ad-91c9-8d77c7d173aa","TopicArn":"arn:aws:s
east-1:333333333333:my-kinesis-test-topic","Subject":"My 3rd subject","Message":"My
 3rd body","Timestamp":"2020-11-27T00:31:39.755Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5"}
```


Analisi dei messaggi per le destinazioni Amazon S3

Questa pagina descrive come analizzare i messaggi di Amazon SNS inviati tramite i flussi di distribuzione di Amazon Data Firehose verso destinazioni Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3).

Per analizzare i messaggi SNS inviati tramite i flussi di distribuzione Firehose verso destinazioni Amazon S3

1. Configura le risorse Amazon S3. Per istruzioni, consulta Creazione di un bucket nella Guida per l'utente di Amazon Simple Storage Service e Utilizzo dei bucket Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.
2. Configura il flusso di consegna. Per istruzioni, consulta Scegli Amazon S3 per la tua destinazione nella Amazon Data Firehose Developer Guide.
3. Utilizza Amazon Athena per eseguire query sugli oggetti Amazon S3 utilizzando SQL standard. Per ulteriori informazioni, consulta l'argomento relativo alle nozioni di base nella Guida per l'utente di Amazon Athena.

Query di esempio

Per questa query di esempio, supponiamo quanto segue:

- I messaggi vengono archiviati nella tabella `notifications` nello schema `default`.

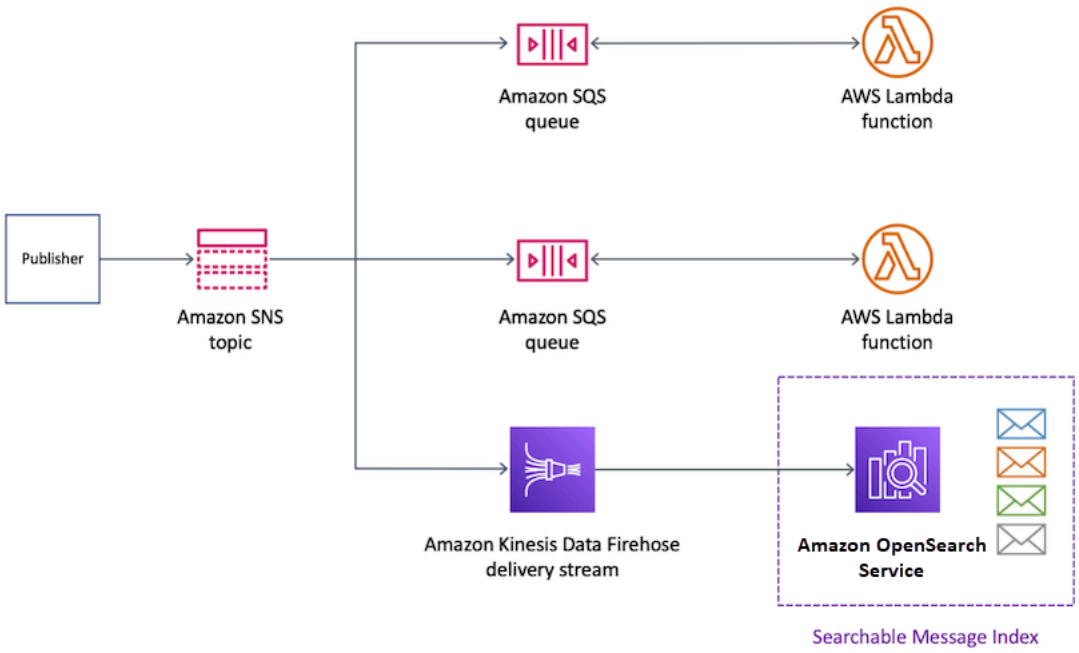
- La tabella `notifications` include una colonna `timestamp` con un tipo di `string`.

La query seguente restituisce tutti i messaggi SNS ricevuti nell'intervallo di date specificato:

```
SELECT *
FROM default.notifications
WHERE from_iso8601_timestamp(timestamp) BETWEEN TIMESTAMP '2020-12-01 00:00:00' AND
 TIMESTAMP '2020-12-02 00:00:00';
```

OpenSearch Destinazioni di servizio

Questa sezione fornisce informazioni sui flussi di distribuzione di Amazon Data Firehose che pubblicano dati su Amazon OpenSearch Service (Service)OpenSearch .



Argomenti

- Formato dei messaggi archiviati negli indici OpenSearch Service
- Analisi dei messaggi per le destinazioni dei OpenSearch servizi

Formato dei messaggi archiviati negli indici OpenSearch Service

L'esempio seguente illustra una notifica Amazon SNS inviata a un indice Amazon OpenSearch Service (OpenSearch Service) denominato `my-index`. Questo indice ha un campo filtro tempo nella scheda `Timestamp`. La notifica SNS viene inserita nella proprietà `_source` del payload.

Note

In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito dei messaggi non elaborati è diasbilitato, Amazon SNS aggiunge i metadati JSON al messaggio, incluse le seguenti proprietà:

- `Type`
- `MessageId`
- `TopicArn`
- `Subject`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Per ulteriori informazioni sulla distribuzione non elaborata, consulta Consegna di messaggi non elaborati Amazon SNS.

```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "49613100963111323203250405402193283794773886550985932802.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
```

```
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
  ]
}
```

Analisi dei messaggi per le destinazioni dei OpenSearch servizi

Questa pagina descrive come analizzare i messaggi Amazon SNS inviati tramite i flussi di consegna di Amazon Data Firehose verso destinazioni Amazon OpenSearch Service (Service). OpenSearch

Per analizzare i messaggi SNS inviati tramite i flussi OpenSearch di consegna Firehose verso le destinazioni del servizio

1. Configura le risorse del tuo servizio. OpenSearch Per istruzioni, consulta la sezione Guida introduttiva ad Amazon OpenSearch Service nella Amazon OpenSearch Service Developer Guide.
2. Configura il flusso di consegna. Per istruzioni, consulta Scegli il OpenSearch servizio per la tua destinazione nella Amazon Data Firehose Developer Guide.
3. Esegui una query utilizzando OpenSearch Service queries e Kibana. Per ulteriori informazioni, consulta Step 3: Search Documents in an OpenSearch Service Domain e Kibana nella Amazon OpenSearch Service Developer Guide.

Query di esempio

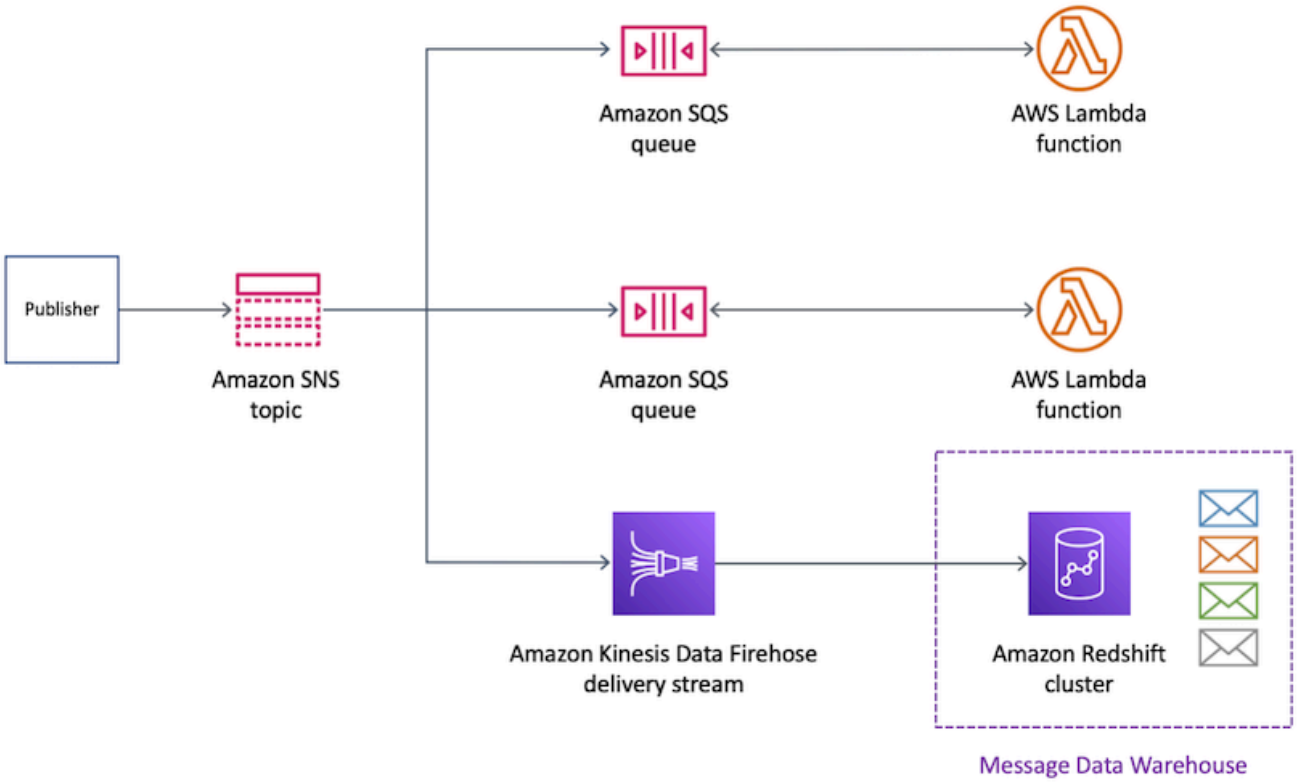
Nell'esempio seguente viene eseguita una query sull'indice `my-index` per tutti i messaggi SNS ricevuti nell'intervallo di date specificato:

```
POST https://search-my-domain.us-east-1.es.amazonaws.com/my-index/_search
{
  "query": {
    "bool": {
      "filter": [
```

```
        {
          "range": {
            "Timestamp": {
              "gte": "2020-12-08T00:00:00.000Z",
              "lte": "2020-12-09T00:00:00.000Z",
              "format": "strict_date_optional_time"
            }
          }
        }
      ]
    }
  }
}
```

Destinazioni Amazon Redshift

Questa sezione descrive come estendere le notifiche di Amazon SNS a un flusso di distribuzione di Amazon Data Firehose che pubblica dati su Amazon Redshift. Con questa configurazione, puoi connetterti al database Amazon Redshift e utilizzare uno strumento di query SQL per eseguire query sul database per ottenere messaggi Amazon SNS che soddisfano determinati criteri.



Argomenti

Struttura della tabella di archiviazione per le destinazioni Amazon Redshift

Per gli endpoint Amazon Redshift, i messaggi Amazon SNS pubblicati vengono archiviati come righe in una tabella. Di seguito è riportato un esempio.

> **Note**
>
> In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito dei messaggi non elaborati è diasbilitato, Amazon SNS aggiunge i metadati JSON al messaggio, incluse le seguenti proprietà:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Message`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Per ulteriori informazioni sulla distribuzione non elaborata, consulta Consegna di messaggi non elaborati Amazon SNS.
> Sebbene Amazon SNS aggiunga proprietà al messaggio utilizzando le maiuscole mostrate in questo elenco, i nomi delle colonne nelle tabelle Amazon Redshift vengono visualizzati con caratteri minuscoli. Per trasformare i metadati JSON per l'endpoint Amazon Redshift, puoi utilizzare il comando SQL `COPY`. Per ulteriori informazioni, consulta Esempi di copia da JSON e Caricamento da dati JSON utilizzando l'opzione "auto ignorecase" nella Guida per gli sviluppatori di database di Amazon Redshift.

type	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageattributes
Notification	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Oggetto di esempio	Messaggio di esempio	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}
Notification	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:11111	Oggetto di esempio 2	Messaggio di esempio 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/	{\"my_attribute2\":{\"Type\":\"String\",\"Val

type	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageattributes
		111111111:my-topic				?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	ue\":\"my_value\"}}

type	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageattributes
Notification	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Oggetto di esempio 3	Messaggio di esempio 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Per maggiori informazioni sull'invio di notifiche agli endpoint Amazon Redshift, consulta Destinazioni Amazon Redshift.

Analisi dei messaggi per le destinazioni Amazon Redshift

Questa pagina descrive come analizzare i messaggi Amazon SNS inviati tramite i flussi di distribuzione di Amazon Data Firehose verso destinazioni Amazon Redshift.

Per analizzare i messaggi SNS inviati tramite i flussi di distribuzione Firehose verso destinazioni Amazon Redshift

1. Configura le risorse Amazon Redshift. Per istruzioni, consulta Nozioni di base su Amazon Redshift nella Guida alle operazioni di Amazon Redshift.
2. Configura il flusso di consegna. Per istruzioni, consulta Scegli Amazon Redshift per la tua destinazione nella Amazon Data Firehose Developer Guide.
3. Eseguire una query. Per ulteriori informazioni, consulta Esecuzione di query su un database con l'editor di query nella Guida alla gestione di Amazon Redshift.

Query di esempio

Per questa query di esempio, supponiamo quanto segue:

- I messaggi vengono archiviati nella tabella `notifications` nello schema predefinito `public`.
- La proprietà `Timestamp` del messaggio SNS viene memorizzata nella colonna `timestamp` della tabella con un tipo `timestamptz` di dati colonna.

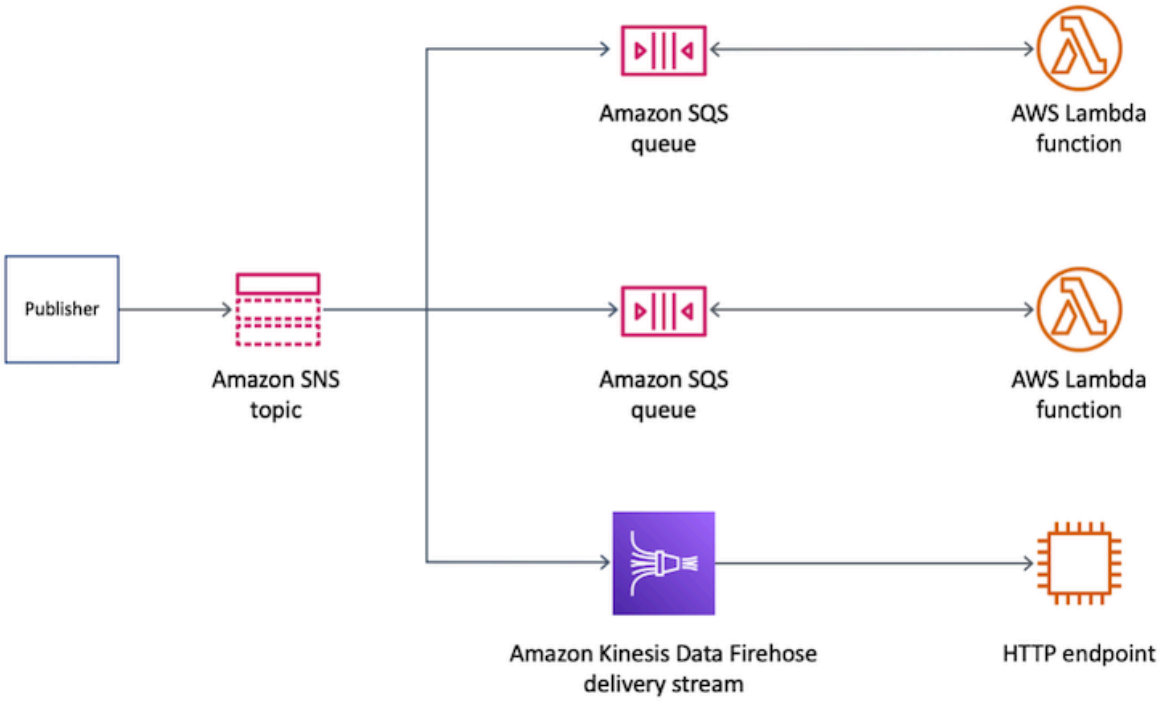
> **Note**
>
> Per trasformare i metadati JSON per l'endpoint Amazon Redshift, puoi utilizzare il comando SQL `COPY`. Per ulteriori informazioni, consulta Esempi di copia da JSON e Caricamento da dati JSON utilizzando l'opzione "auto ignorecasee" nella Guida per sviluppatori di Amazon Redshift.

La query seguente restituisce tutti i messaggi SNS ricevuti nell'intervallo di date specificato:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

Destinations HTTP

Questa sezione fornisce informazioni sui flussi di distribuzione di Amazon Data Firehose che pubblicano dati su endpoint HTTP.



Argomenti

- Formato dei messaggi recapitati per le destinazioni HTTP

Formato dei messaggi recapitati per le destinazioni HTTP

Di seguito è riportato un esempio di corpo di richiesta HTTP POST di Amazon SNS che un flusso di distribuzione di Amazon Data Firehose può inviare all'endpoint HTTP. La notifica SNS è codificata come payload base64 nella proprietà `records`.

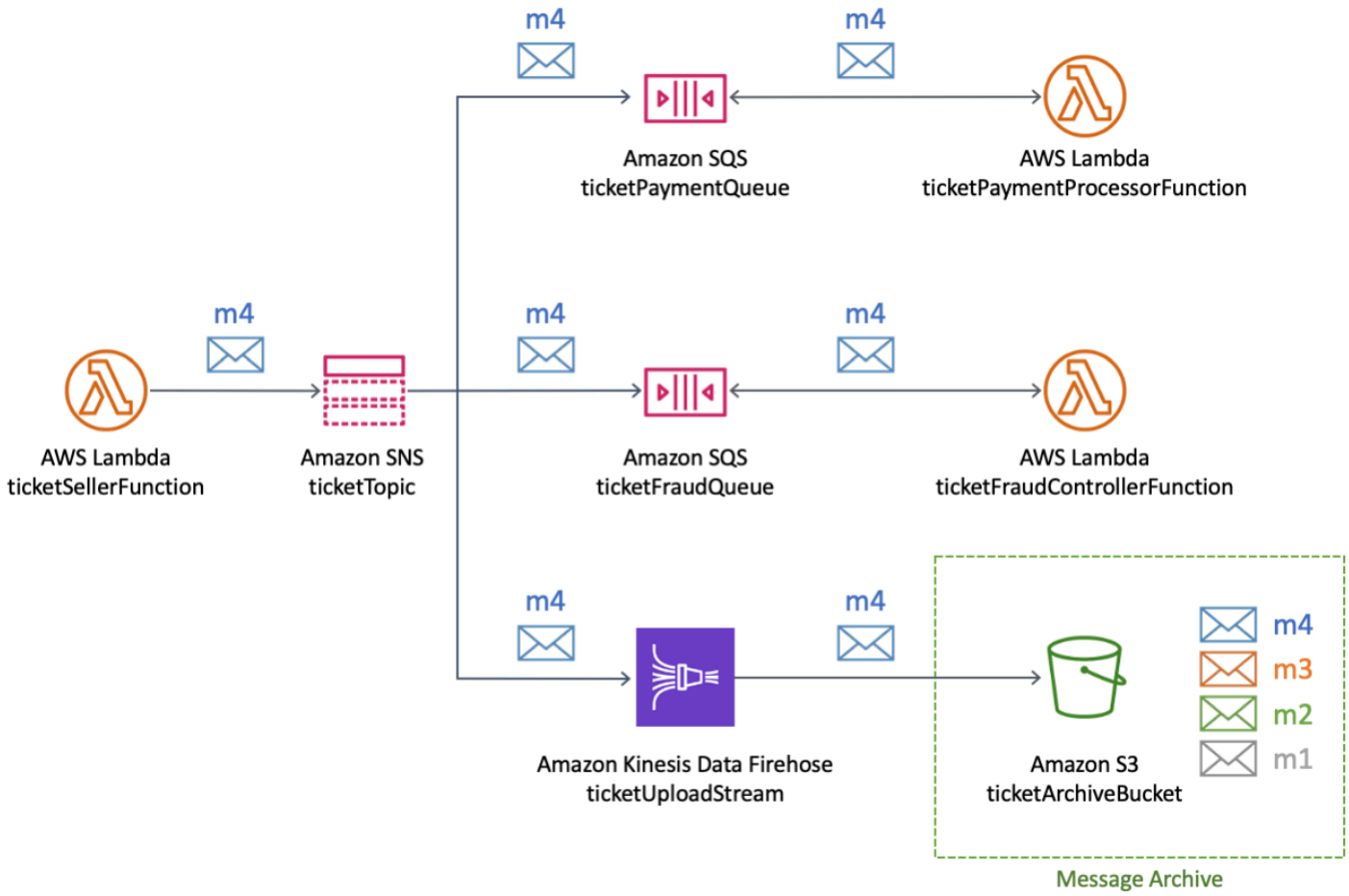
> **Note**
>
> In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Per ulteriori informazioni sulla distribuzione non elaborata, consulta Consegna di messaggi non elaborati Amazon SNS.

```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
        "data":
 "eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```


Esempio di utilizzo per l'archiviazione e l'analisi dei messaggi

Questa sezione fornisce un'esercitazione su un caso d'uso comune per l'archiviazione e l'analisi dei messaggi Amazon SNS.

L'impostazione di questo caso d'uso è una piattaforma di biglietteria aerea che opera in un ambiente regolamentato. La piattaforma è soggetta a un framework di conformità che richiede all'azienda di archiviare tutte le vendite dei biglietti per almeno cinque anni. Per raggiungere l'obiettivo di conformità relativo alla conservazione dei dati, l'azienda sottoscrive uno stream di distribuzione Amazon Data Firehose su un argomento SNS esistente. La destinazione per il flusso di distribuzione è un bucket Amazon Simple Storage Service (Amazon S3). Con questa configurazione, tutti gli eventi pubblicati nell'argomento SNS vengono archiviati nel bucket Amazon S3. Nel diagramma seguente viene illustrata l'architettura di questa configurazione.



Per eseguire analisi e ottenere informazioni dettagliate sulle vendite dei biglietti, l'azienda esegue le query SQL utilizzando Amazon Athena. Ad esempio, l'azienda può eseguire query per conoscere le destinazioni più popolari e i volantini più frequenti.

Per creare risorse AWS per questo caso d'uso, è possibile utilizzare AWS Management Console o un modello AWS CloudFormation.

Argomenti

- Creazione delle risorse iniziali
- Creazione del flusso di distribuzione Firehose
- Sottoscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS
- Testare e interrogare la configurazione
- Utilizzo del modello AWS CloudFormation

Creazione delle risorse iniziali

In questa pagina viene descritto come creare le seguenti risorse per l'esempio di utilizzo dell'archiviazione dei messaggi e dell'analisi:

- Un bucket Amazon Simple Storage Service (Amazon S3).
- Due code di Amazon Simple Queue Service (Amazon SQS)
- Un argomento Amazon SNS
- Due abbonamenti Amazon SQS all'argomento Amazon SNS

Per creare le risorse iniziali

1. Creato il bucket Amazon S3:

   a. Aprire la console Amazon S3.
   b. Scegliere Create bucket (Crea bucket).
   c. In Bucket name (Nome bucket), immettere un nome univoco globale. Mantenere gli altri campi come valori predefiniti.
   d. Scegliere Create bucket (Crea bucket).

   Per ulteriori informazioni sui bucket Amazon S3, consulta Creazione di un bucket nella Guida per l'utente di Amazon Simple Storage Service e Utilizzo dei bucket Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.
2. Crea le due code Amazon SQS:

   a. Apri la console Amazon SQS.
   b. Scegliere Create queue (Crea coda).
   c. Per Tipo, scegliere Standard.
   d. In Name (Nome), inserire **`ticketPaymentQueue`**.
   e. Su Policy di accesso, per Scegli il metodo, scegliere Avanzato.
   f. Nella casella Criteri JSON, incollare la policy seguente:

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:ticketTopic"
        }
      }
    }
  ]
}
```

In questa policy di accesso, sostituire il numero Account AWS (*123456789012*) con il tuo, e cambiare la regione AWS(*us-east-1*) di conseguenza.

g. Scegliere Create queue (Crea coda).

h. Ripetere questi passaggi per creare una seconda coda SQS denominata **ticketFraudQueue**.

Per ulteriori informazioni sulla creazione delle code SQS, consulta Creazione di una coda Amazon SQS (console) nella Giuda per sviluppatori Amazon Simple Queue Service.

3. Creazione dell'argomento SNS:

   a. Aprire la pagina Topics (Argomenti) nella console Amazon SNS.

   b. Scegliere Create topic (Crea argomento).

   c. Su Details (Dettagli), per Type (Tipo), scegliere Standard.

   d. In Name (Nome), inserire **ticketTopic**.

   e. Scegliere Create topic (Crea argomento).

   Per ulteriori informazioni sulla creazione di argomenti SNS, consulta Creare un argomento Amazon SNS.

4. Sottoscrizione della coda SQS all'argomento SNS:

   a. Nella Console Amazon SNS, nella pagina ticketTopic dei dettagli dell'argomento, scegli Creazione di una sottoscrizione.

b. Su Details (Dettagli), per Protocol (Protocollo), scegliere Amazon SQS.

c. Per Endpoint scegli l'Amazon Resource Name (ARN) della coda TicketPaymentQueue.

d. Scegli Create Subscription (Crea sottoscrizione).

e. Ripetere queste fasi per creare una seconda sottoscrizione utilizzando l'ARN della coda TicketFraudCoda.

   Per ulteriori informazioni su come sottoscrivere argomenti SNS, consulta Iscrizione a un argomento Amazon SNS. Puoi anche sottoscrivere code SQS agli argomenti SNS dalla console Amazon SQS. Per ulteriori informazioni, consulta Sottoscrizione di una coda Amazon SQS a un argomento Amazon SNS (console) nella Guida per sviluppatori di Amazon Simple Queue Service.

Sono state create le risorse iniziali per questo caso d'uso di esempio. Per continuare, consulta Creazione del flusso di distribuzione Firehose.

Creazione del flusso di distribuzione Firehose

Questa pagina descrive come creare il flusso di distribuzione di Amazon Data Firehose per il caso d'uso di esempio di archiviazione e analisi dei messaggi.

Per creare il flusso di distribuzione di Firehose

1. Apri la console del servizio Amazon Kinesis.
2. Scegli Firehose, quindi scegli Crea flusso di distribuzione.
3. Nella pagina Nuovo flusso di distribuzione, per Nome del flusso di distribuzione, immettere **`ticketUploadStream`**, quindi scegliere Successivo.
4. Nella pagina Elaborazione di record, scegliere Successivo.
5. Nella pagina Seleziona destinazione, procedere come segue:

   a. Per Destination (Destinazione), scegliere Amazon S3.

   b. Su Destinazione S3, per Bucket S3, scegli il bucket S3 creato inizialmente.

   c. Seleziona Successivo.
6. Nella pagina Configurare le impostazioni, per Condizioni buffer S3 effettua le seguenti operazioni:

   - Per Dimensione del buffer, immettere **1**.

- Per Intervallo buffer, immettere **60**.

L'utilizzo di questi valori per il buffer Amazon S3 consente di testare rapidamente la configurazione. La prima condizione è soddisfatta attiva la distribuzione dei dati al bucket S3.

7. Nella pagina Configurare le impostazioni, per Autorizzazioni, scegliere di creare un ruolo (IAM) AWS Identity and Access Management con le autorizzazioni necessarie assegnate automaticamente. Quindi scegli Successivo.
8. Nella pagina Review (Revisione), scegliere Create delivery stream (Creare flusso di distribuzione).
9. Dalla pagina dei flussi di distribuzione di Kinesis Data Firehose, scegli il flusso di distribuzione che hai appena creato (). ticketUploadStream Nella tabella Dettagli, annota l'Amazon Resource Name (ARN) del relativo flusso per un momento successivo.

Per ulteriori informazioni sulla creazione di flussi di distribuzione, consulta Creating an Amazon Data Firehose Delivery Stream nella Amazon Data Firehose Developer Guide. Per ulteriori informazioni sulla creazione di un ruolo IAM, consulta Creazione di un ruolo per delegare le autorizzazioni a un servizio AWS nella Guida per l'utente di IAM.

Hai creato il flusso di distribuzione di Firehose con le autorizzazioni richieste. Per continuare, consulta Sottoscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS.

Sottoscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS

In questa pagina viene descritto come creare quanto segue per l'esempio di utilizzo dell'archiviazione dei messaggi e dell'analisi:

- Il ruolo AWS Identity and Access Management (IAM) che consente all'abbonamento Amazon SNS di inserire record nel flusso di distribuzione di Amazon Data Firehose
- L'abbonamento Firehose Delivery Stream all'argomento SNS

Per creare il ruolo IAM per l'abbonamento Amazon SNS

1. Aprire la pagina Roles (Ruoli) della console IAM.
2. Scegliere Create role (Crea ruolo).
3. Per Select type of trusted entity (Seleziona tipo di entità attendibile), seleziona AWS service (Servizio AWS).

4. Per Scegli un caso d'uso, scegliere SNS. Quindi scegliere Next: Permissions (Successivo: Autorizzazioni).
5. Scegliere Next: Tags (Successivo: Tag).
6. Scegliere Next:Review (Successivo: Rivedi).
7. Nella pagina Review (Rivedi), per Role name (Nome ruolo), immettere **`ticketUploadStreamSubscriptionRole`**. Quindi seleziona Create role (Crea ruolo).
8. Quando il ruolo viene creato, scegli il suo nome () ticketUploadStreamSubscriptionRole.
9. Nrlla pagina Riepilogo del ruolo, scegliere Aggiunta di policy inline.
10. Alla pagina Create policy (Crea policy), selezionare la scheda JSON, poi copiare la policy nella casella di testo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams",
                "firehose:ListTagsForDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ticketUploadStream"
            ],
            "Effect": "Allow"
        }
    ]
}
```

    In questa policy, sostituire il numero Account AWS (*`123456789012`*) con il tuo, e cambiare la Regione AWS (*`us-east-1`*) di conseguenza.
11. Scegliere Review policy (Esamina policy).
12. Nella pagina Review policy (Esamina policy), per Name (Nome), immettere **`FirehoseSnsPolicy`**. Quindi scegliere Create policy (Crea policy).
13. Nella pagina Riepilogo del ruolo, prendere nota dell'ARN ruolo per dopo.

Per ulteriori informazioni sulla creazione di un ruolo IAM, consulta Creazione di un ruolo per delegare le autorizzazioni a un servizio AWS nella Guida per l'utente di IAM.

Per sottoscrivere lo stream di distribuzione di Firehose all'argomento SNS

1. Aprire la pagina Topics (Argomenti) nella console Amazon SNS.
2. Nella tabella Subscriptions (Sottoscrizioni) scegliere Create subscription (Crea sottoscrizione).
3. In Dettagli, per Protocollo, scegli Amazon Data Firehose.
4. Per Endpoint, inserisci l'Amazon Resource Name (ARN) ticketUploadStreamdel flusso di distribuzione creato in precedenza. Ad esempio, specifica **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Per ARN del ruolo di sottoscrizione, inserisci l'ARN del ruolo ticketUploadStreamSubscriptionRoleIAM che hai creato in precedenza. Ad esempio, specifica **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Selezionare l'Abilitazione del recapito di messaggi.
7. Scegli Create Subscription (Crea sottoscrizione).

È stato creato il ruolo IAM e la sottoscrizione all'argomento SNS. Per continuare, consulta Testare e interrogare la configurazione.

Testare e interrogare la configurazione

In questa pagina viene descritto come testare l'esempio di utilizzo dell'archiviazione dei messaggi e dell'analisi pubblicando un messaggio nell'argomento Amazon SNS. Le istruzioni includono una query di esempio che è possibile eseguire e adattare alle proprie esigenze.

Per testare la configurazione

1. Aprire la pagina Topics (Argomenti) nella console Amazon SNS.
2. Seleziona l'argomento **`ticketTopic`**.
3. Seleziona Publish message (Pubblica messaggio).
4. Nella pagina Pubblica il messaggio nell'argomento inserisci quanto segue per il corpo del messaggio. Aggiungi un carattere di nuova riga alla fine del messaggio.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

Mantenere tutte le altre opzioni come valori predefiniti.

5. Seleziona Publish message (Pubblica messaggio).

   Per ulteriori informazioni sulla pubblicazione dei messaggi, consulta Pubblicazione messaggi di Amazon SNS.

6. Dopo l'intervallo di flusso di consegna di 60 secondi, aprire la finestra Console Amazon Simple Storage Service (Amazon S3) e scegliere il bucket Amazon S3 creato inizialmente.

   Il messaggio pubblicato appare nel bucket.

Per eseguire una query sui dati

1. Aprire la console Amazon Athena.
2. Eseguire una query.

   Ad esempio, si supponga che la tabella `notifications` nello schema `default` contenga i seguenti dati:

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 11:30:15","Destination":"Miami","FlyingFrom":"Omaha","TicketNumber":"efgh5678"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 3:30:10","Destination":"Miami","FlyingFrom":"NewYork","TicketNumber":"ijkl9012"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 12:30:05","Destination":"Delhi","FlyingFrom":"Omaha","TicketNumber":"mnop3456"}
```

   Per trovare la destinazione principale, eseguire la seguente query:

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

   Per eseguire una query per i ticket venduti in un intervallo di data e ora specifico, eseguire una query simile alla seguente:

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

È possibile adattare entrambe le query di esempio per le proprie esigenze. Per ulteriori informazioni sull'utilizzo di Athena per eseguire query, consulta Nozioni di base nellaGuida per l'utente di Amazon Athena.

Pulizia

Per evitare di incorrere in costi di utilizzo dopo aver terminato il test, eliminare le seguenti risorse create durante l'esercitazione:

- Abbonamenti Amazon SNS
- Argomento Amazon SNS
- Code di Amazon Simple Queue Service (Amazon SQS)
- Bucket di Amazon S3
- Flusso di distribuzione di Amazon Data Firehose
- AWS Identity and Access Management ruoli e criteri (IAM)

Utilizzo del modello AWS CloudFormation

Per automatizzare l'implementazione caso d'uso d'esempio di archiviazione dei messaggi e analisi dei dati di Amazon SNS, è possibile utilizzare il seguente modello YAML:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Template for creating an SNS archiving use case
Resources:
  ticketUploadStream:
    DependsOn:
    - ticketUploadStreamRolePolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN: !Sub 'arn:${AWS::Partition}:s3:::${ticketArchiveBucket}'
```

```
      BufferingHints:
        IntervalInSeconds: 60
        SizeInMBs: 1
      CompressionFormat: UNCOMPRESSED
      RoleARN: !GetAtt ticketUploadStreamRole.Arn
ticketArchiveBucket:
  Type: AWS::S3::Bucket
ticketTopic:
  Type: AWS::SNS::Topic
ticketPaymentQueue:
  Type: AWS::SQS::Queue
ticketFraudQueue:
  Type: AWS::SQS::Queue
ticketQueuePolicy:
  Type: AWS::SQS::QueuePolicy
  Properties:
    PolicyDocument:
      Statement:
        Effect: Allow
        Principal:
          Service: sns.amazonaws.com
        Action:
          - sqs:SendMessage
        Resource: '*'
        Condition:
          ArnEquals:
            aws:SourceArn: !Ref ticketTopic
    Queues:
      - !Ref ticketPaymentQueue
      - !Ref ticketFraudQueue
ticketUploadStreamSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketUploadStream.Arn
    Protocol: firehose
    SubscriptionRoleArn: !GetAtt ticketUploadStreamSubscriptionRole.Arn
ticketPaymentQueueSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketPaymentQueue.Arn
    Protocol: sqs
ticketFraudQueueSubscription:
```

```
    Type: AWS::SNS::Subscription
    Properties:
      TopicArn: !Ref ticketTopic
      Endpoint: !GetAtt ticketFraudQueue.Arn
      Protocol: sqs
  ticketUploadStreamRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Sid: ''
          Effect: Allow
          Principal:
            Service: firehose.amazonaws.com
          Action: sts:AssumeRole
  ticketUploadStreamRolePolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: FirehoseticketUploadStreamRolePolicy
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - s3:AbortMultipartUpload
          - s3:GetBucketLocation
          - s3:GetObject
          - s3:ListBucket
          - s3:ListBucketMultipartUploads
          - s3:PutObject
          Resource:
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}'
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}/*'
      Roles:
      - !Ref ticketUploadStreamRole
  ticketUploadStreamSubscriptionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
```

```
            Service:
            - sns.amazonaws.com
          Action:
          - sts:AssumeRole
      Policies:
      - PolicyName: SNSKinesisFirehoseAccessPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Action:
            - firehose:DescribeDeliveryStream
            - firehose:ListDeliveryStreams
            - firehose:ListTagsForDeliveryStream
            - firehose:PutRecord
            - firehose:PutRecordBatch
            Effect: Allow
            Resource:
            - !GetAtt ticketUploadStream.Arn
```

Funzioni da Fanout a Lambda

Amazon SNS e AWS Lambda sono integrati di modo che sia possibile richiamare funzioni Lambda utilizzando notifiche Amazon SNS. Quando un messaggio viene pubblicato in un argomento SNS a cui una funzione Lambda ha effettuato la sottoscrizione, la funzione Lambda viene richiamata con il payload del messaggio pubblicato. La funzione Lambda riceve il payload del messaggio come parametro di input e può manipolare le informazioni nel messaggio, pubblicare il messaggio in altri argomenti SNS o inviare il messaggio ad altri servizi AWS.

Inoltre, Amazon SNS supporta gli attributi relativi allo stato di consegna dei messaggi per le notifiche dei messaggi inviate agli endpoint Lambda. Per ulteriori informazioni, consulta Stato di consegna dei messaggi Amazon SNS.

Prerequisites

Per richiamare le funzioni Lambda utilizzando le notifiche Amazon SNS, è necessario quanto segue:

- valida e completa
- Argomento Amazon SNS

Per informazioni sulla creazione di una funzione Lambda da da utilizzare con Amazon SNS, consulta Uso di Lambda con Amazon SNS. Per ulteriori informazioni sulla creazione di un argomento Amazon SNS, consultare Creazione di un argomento.

Quando si utilizza Amazon SNS per recapitare i messaggi dalle regioni di opt-in alle regioni abilitate per impostazione predefinita, è necessario modificare la policy creata nella funzione AWS Lambda sostituendo l'entità `sns.amazonaws.com` con `sns.<opt-in-region>.amazonaws.com`.

Ad esempio, se si desidera sottoscrivere una funzione Lambda negli Stati Uniti orientali (Virginia settentrionale) a un argomento SNS in Asia Pacifico (Hong Kong), modificare il criterio funzione AWS Lambda in `sns.ap-east-1.amazonaws.com`. Le regioni opt-in includono tutte le regioni lanciate dopo il 20 marzo 2019 e includono Asia Pacifico (Hong Kong), Medio Oriente (Bahrein), UE (Milano) e Africa (Città del Capo). Le regioni lanciate prima del 20 marzo 2019 sono abilitate per impostazione predefinita.

> **Note**
>
> AWS non supporta la consegna tra regioni ad AWS Lambda da una regione abilitata per impostazione predefinita a una regione opt-in. Inoltre, l'inoltro tra regioni di messaggi SNS da regioni opt-in ad altre regioni opt-in non è supportato.

Sottoscrizione di una funzione a un argomento

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, selezionare Topics (Argomenti).
3. Nella pagina Topics (Argomenti) scegliere un argomento.
4. Nella sezione Subscriptions (Sottoscrizioni) scegliere Create subscription (Crea sottoscrizione).
5. Nella pagina Create subscription (Crea sottoscrizione), nella sezione Details (Dettagli), eseguire queste operazioni:
   a. Verificare il valore Topic ARN (ARN argomento) scelto.
   b. Per Protocol (Protocollo) scegliere AWS Lambda.
   c. Per Endpoint immettere l'ARN di una funzione.
   d. Scegli Create Subscription (Crea sottoscrizione).

Quando un messaggio viene pubblicato in un argomento SNS a cui una funzione Lambda ha effettuato la sottoscrizione, la funzione Lambda viene richiamata con il payload del messaggio pubblicato. Per informazioni su come utilizzare AWS Lambda con Amazon SNS, incluso un tutorial, consulta Utilizzo di AWS Lambda con Amazon SNS.

Fan-out a code Amazon SQS

Amazon SNS funziona a stretto contatto con Amazon Simple Queue Service (Amazon SQS). Questi servizi offrono numerosi vantaggi agli sviluppatori. Amazon SNS consente alle applicazioni di inviare messaggi con vincoli tempistici a più sottoscrittori grazie a un meccanismo "push", eliminando la necessità di cercare periodicamente gli aggiornamenti o di "eseguirne il polling". Amazon SQS; è un servizio di accodamento di messaggi utilizzato dalle applicazioni distribuite per scambiare messaggi mediante un modello di polling. Può essere utilizzato per separare i componenti di invio e ricezione, senza richiedere la disponibilità simultanea di tutti i componenti. Grazie all'integrazione di Amazon SNS con Amazon SQS, è possibile recapitare i messaggi alle applicazioni che richiedono la notifica immediata di un evento e renderli persistenti in una coda Amazon SQS in modo da essere elaborati in seguito da altre applicazioni.

Quando esegui la sottoscrizione di una coda Amazon SQS; a un argomento Amazon SNS, puoi pubblicare un messaggio nell'argomento e Amazon SNS invia un messaggio Amazon SQS alla coda per la quale è stata eseguita la sottoscrizione. Il messaggio Amazon SQS contiene l'oggetto e il messaggio pubblicati nell'argomento insieme ai metadati relativi al messaggio in un documento JSON. Il messaggio Amazon SQS sarà simile al documento JSON seguente.

```
{
   "Type" : "Notification",
   "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
   "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
   "Subject" : "Testing publish to subscribed queues",
   "Message" : "Hello world!",
   "Timestamp" : "2012-03-29T05:12:16.901Z",
   "SignatureVersion" : "1",
   "Signature" : "EXAMPLEnTrFPa3...",
   "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:MyTopic:c7fe3a54-
ab0e-4ec2-88e0-db410a0f2bee"
}
```

Iscrizione di una coda Amazon SQS a un argomento Amazon SNS

Per attivare un argomento Amazon SNS e inviare messaggi a una coda Amazon SQS, procedi in uno dei seguenti modi:

- Utilizzo del Console Amazon SQS, che semplifica il processo. Per ulteriori informazioni, consulta Tutorial: Sottoscrizione di una coda Amazon SQS a un argomento Amazon SNS nella Guida per sviluppatori di Amazon Simple Queue Service.
- Completare la procedura riportata di seguito.
  1. Ottieni l'Amazon Resource Name (ARN) della coda a cui desideri inviare i messaggi e l'argomento a cui intendi sottoscrivere la coda.
  2. Concedi l'autorizzazione `sqs:SendMessage` all'argomento Amazon SNS in modo che possa inviare messaggi alla coda.
  3. Sottoscrizione della coda all'argomento Amazon SNS.
  4. Concedi agli utenti IAM o agli Account AWS le autorizzazioni appropriate per pubblicare nell'argomento Amazon SNS e leggere i messaggi nella coda Amazon SQS.
  5. Verifica la procedura pubblicando un messaggio nell'argomento e leggendo il messaggio dalla coda.

Per informazioni sulla configurazione di un argomento per l'invio di messaggi a una coda che si trova in un account AWS differente, consultare Invio di messaggi Amazon SNS a una coda Amazon SQS; in un altro account.

Per visualizzare un modello AWS CloudFormation che crea un argomento che invia messaggi a due code, consulta Utilizzo di un modello di AWS CloudFormation per creare un argomento che invia messaggi alle code Amazon SQS.

Fase 1: ottenere l'ARN della coda e dell'argomento

Quando esegui la sottoscrizione di una coda al tuo argomento, necessiti di una copia dell'ARN della coda. Analogamente, quando autorizzi l'argomento a inviare messaggi, necessiti di una copia dell'ARN dell'argomento.

Per ottenere l'ARN della coda, puoi utilizzare la console Amazon SQS o l'operazione API GetQueueAttributes.

Ottenimento dell'ARN della coda dalla console Amazon SQS

1. Accedere alla AWS Management Console e aprire la console Amazon SQS all'indirizzo https://console.aws.amazon.com/sqs/.
2. Seleziona la casella per la coda di cui intendi ottenere l'ARN.
3. Nella scheda Details (Dettagli), copia il valore dell'ARN in modo da utilizzarlo per la sottoscrizione all'argomento Amazon SNS.

Per ottenere l'ARN dell'argomento, puoi utilizzare la console Amazon SNS, il comando `sns-get-topic-attributes` o l'operazione API `GetQueueAttributes`.

Ottenimento dell'ARN dell'argomento dalla console Amazon SNS

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, scegliere l'argomento di cui si desidera ottenere l'ARN.
3. Nella sezione Details (Dettagli), copiare il valore di ARN in modo da poterlo utilizzare per autorizzare l'argomento Amazon SNS a inviare messaggi alla coda.

Fase 2: Concedere all'argomento Amazon SNS l'autorizzazione a inviare messaggi alla coda Amazon SQS

Per consentire a un argomento Amazon SNS di inviare messaggi a una coda, devi definire una policy sulla coda che consenta all'argomento Amazon SNS di eseguire l'operazione `sqs:SendMessage`.

Prima di eseguire la sottoscrizione di una coda a un argomento, devi creare un argomento e una coda. Se non lo hai già fatto, creali adesso. Per ulteriori informazioni, consultare Creazione di un argomento e Creare una coda nella Guida per sviluppatori di Amazon Simple Queue Service.

Per impostare una policy su una coda, puoi utilizzare la console Amazon SQS oppure l'operazione API SetQueueAttributes. Prima di iniziare, assicurati di avere l'ARN dell'argomento a cui intendi concedere l'autorizzazione a inviare messaggi alla coda. Se stai sottoscrivendo una coda a più argomenti, la policy deve contenere un elemento `Statement` per ogni argomento.

Impostazione di una policy SendMessage su una coda utilizzando la console Amazon SQS

1. Accedere alla AWS Management Console e aprire la console Amazon SQS all'indirizzo https://console.aws.amazon.com/sqs/.

2. Seleziona la casella della coda per la quale intendi impostare la policy, scegli la scheda Policy di accesso, quindi scegli Modifica.
3. Nella Policy di accesso, definire chi può accedere alla coda.

   - Aggiungi una condizione che autorizza l'operazione per l'argomento.
   - Impostare `Principal` come servizio Amazon SNS, come mostrato nell'esempio seguente.
   - Utilizzare le chiavi di condizione globali `aws:SourceArn` o `aws:SourceAccount` per proteggersi dallo scenario Confused deputy. Per utilizzare queste chiavi di condizione, impostare il valore sull'ARN del proprio argomento. Se la coda è sottoscritta a più argomenti, è possibile usare invece `aws:SourceAccount`.

   Ad esempio, la policy seguente consente a MioArgomento di inviare messaggi a MiaCoda.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
      }
    }
  ]
}
```

Fase 3: eseguire la sottoscrizione della coda all'argomento Amazon SNS

Per inviare messaggi a una coda tramite un argomento, devi eseguire la sottoscrizione della coda all'argomento Amazon SNS. La coda viene specificata mediante il relativo ARN. Per effettuare la sottoscrizione a un argomento, puoi utilizzare la console Amazon SNS, il comando della CLI `sns-subscribe` o l'operazione API `Subscribe`. Prima di iniziare, assicurati di avere l'ARN della coda per la quale intendi eseguire la sottoscrizione.

1. Accedi a Amazon SNS console
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti) scegliere un argomento.
4. Nella pagina ***MioArgomento***, nella sezione Subscriptions (Sottoscrizioni), scegliere Create subscription (Crea sottoscrizione).
5. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:

   a. Verificare il valore Topic ARN (ARN argomento).

   b. Per Protocollo, scegliere Amazon SQS.

   c. Per Endpoint immettere l'ARN di una coda Amazon SQS.

   d. Scegli Create Subscription (Crea sottoscrizione).

   Dopo la conferma della sottoscrizione, il campo Subscription ID (ID sottoscrizione) della nuova sottoscrizione visualizza il relativo ID. Se il proprietario della coda crea la sottoscrizione, questa viene automaticamente confermata e dovrebbe essere attiva quasi immediatamente.

   In genere, esegui la sottoscrizione della tua coda al tuo argomento nel tuo account. Tuttavia, puoi anche eseguire la sottoscrizione di una coda in un altro account al tuo argomento. Se l'utente che crea la sottoscrizione non è il proprietario della coda (ad esempio, se un utente dell'account A esegue la sottoscrizione di una coda nell'account B a un argomento nell'account A), la sottoscrizione deve essere confermata. Per ulteriori informazioni sulla sottoscrizione di una coda in un account differente e sulla conferma della sottoscrizione, consulta Invio di messaggi Amazon SNS a una coda Amazon SQS; in un altro account.

Fase 4: concedere agli utenti le autorizzazioni per le operazioni appropriate su argomenti e code

Per autorizzare solo gli utenti appropriati a pubblicare nell'argomento Amazon SNS e a leggere/eliminare i messaggi dalla coda Amazon SQS, devi utilizzare AWS Identity and Access Management (IAM). Per ulteriori informazioni sul controllo delle operazioni sugli argomenti e le code per gli utenti IAM, consulta Utilizzo di policy basate su identità con Amazon SNS, e Identity and Access Management in Amazon SQS nella Guida per sviluppatori di Amazon Simple Queue Service.

Esistono due modi di controllare l'accesso a un argomento o a una coda:

- Aggiungere una policy a un utente o gruppo IAM. Il modo più semplice di concedere agli utenti le autorizzazioni per argomenti o code è di creare un gruppo e aggiungere a quel gruppo dapprima la policy appropriata e quindi gli utenti. È molto più semplice aggiungere e rimuovere utenti da un gruppo anziché tenere traccia delle policy impostate su singoli utenti.
- Aggiungere una policy a un argomento o a una coda. Se intendi concedere delle autorizzazioni per un argomento o una coda a un altro account AWS, l'unica soluzione è aggiungere una policy che ha come principale Account AWS a cui intendi concedere le autorizzazioni.

Il primo metodo deve essere utilizzato nella maggior parte dei casi (applicare policy a gruppi e gestire le autorizzazioni per gli utenti aggiungendo o rimuovendo gli utenti appropriati ai gruppi). Se invece hai la necessità di concedere delle autorizzazioni a un utente in un altro account, devi utilizzare il secondo metodo.

Aggiunta di una policy a un utente o gruppo IAM

Se aggiungessi la policy seguente a un utente o gruppo IAM, autorizzeresti quell'utente o i membri di quel gruppo a eseguire l'operazione `sns:Publish` sull'argomento MyTopic.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Se aggiungessi la policy seguente a un utente o gruppo IAM, autorizzeresti quell'utente o i membri di quel gruppo a eseguire le operazioni `sqs:ReceiveMessage` e `sqs:DeleteMessage` sulle code MyQueue1 e MyQueue2.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
```

```
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

Aggiunta di una policy a un argomento o a una coda

Gli esempi di policy seguenti mostrano come concedere autorizzazioni per un argomento e una coda a un altro account.

> **Note**
>
> Quando concedi a un altro Account AWS l'accesso a una risorsa nel tuo account, lo concedi anche agli utenti IAM che dispongono di autorizzazioni di accesso di livello amministratore (accesso generico). L'accesso alla risorsa viene automaticamente negato a tutti gli altri utenti IAM nell'altro account. Se intendi concedere l'accesso alla risorsa a specifici utenti IAM in quel Account AWS, l'account o un utente IAM con accesso di livello amministratore deve delegare le autorizzazioni per la risorsa a quegli utenti IAM. Per ulteriori informazioni sulla delega multiaccount, consulta la sezione relativa all'abilitazione dell'accesso multiaccount nella Guida all'uso di IAM.

Se aggiungi la policy seguente a un argomento MioArgomento nell'account 123456789012, autorizzi l'account 111122223333 a eseguire l'operazione `sns:Publish` su quell'argomento.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Se aggiungi la policy seguente a una coda MyQueue nell'account 123456789012, autorizzi l'account 111122223333 a eseguire le azioni `sqs:ReceiveMessage` e `sqs:DeleteMessage` su quella coda.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Fase 5: eseguire la verifica delle sottoscrizioni della coda all'argomento

Puoi eseguire la verifica delle sottoscrizioni di una coda a un argomento pubblicando nell'argomento e visualizzando il messaggio che l'argomento invia alla coda.

Per pubblicare un argomento utilizzando la console Amazon SNS

1. Utilizzando le credenziali del Account AWS o di un utente IAM con l'autorizzazione da pubblicare nell'argomento, accedi alla AWS Management Console e apri la console Amazon SNS a https://console.aws.amazon.com/sns/.
2. Nel riquadro di navigazione, seleziona l'argomento e scegli Publish to Topic (Pubblica nell'argomento).
3. Nella casella Subject (Oggetto), immettere un oggetto (ad esempio, **`Testing publish to queue`**) nella casella Message (Messaggio), immettere del testo (ad esempio, **`Hello world!`**) e selezionare Publish Message (Pubblica messaggio). Viene visualizzato il messaggio "Your message has been successfully published" (Il messaggio è stato pubblicato).

Visualizzazione del messaggio dall'argomento utilizzando la console Amazon SQS

1. Utilizzando le credenziali del Account AWS o dell'utente IAM autorizzato a visualizzare messaggi nella coda, accedi alla AWS Management Console e apri la console Amazon SQS a https://console.aws.amazon.com/sqs/.
2. Scegli una coda iscritta all'argomento.
3. Scegli Send and receive messages (Invia e ricevi messaggi), quindi seleziona Poll for messages (Polling per i messaggi). Viene visualizzato un messaggio di tipo notifica.
4. Nella colonna Body (Corpo), scegli More Details (Altri dettagli). La casella Message Details (Dettagli messaggio) contiene un documento JSON con l'oggetto e il messaggio che hai pubblicato nell'argomento. Il messaggio risulta simile al documento JSON seguente.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-db410a0f2bee"
}
```

5. Scegliere Close (Chiudi). La pubblicazione in un argomento che invia messaggi di notifica a una coda è completata.

Utilizzo di un modello di AWS CloudFormation per creare un argomento che invia messaggi alle code Amazon SQS

AWS CloudFormation consente di utilizzare un file di modello per creare e configurare una raccolta di risorse AWS come una singola unità. Questa sezione include un modello di esempio in grado di semplificare la distribuzione di argomenti che effettuano pubblicazioni nelle code. I modelli eseguono automaticamente la procedura di configurazione creando due code e un argomento con sottoscrizioni

alle code, aggiungendo una policy alle code affinché l'argomento possa inviare messaggi alle code e creando utenti e gruppi IAM per controllare l'accesso a tali risorse.

Per ulteriori informazioni sulla distribuzione delle risorse AWS mediante un modello AWS CloudFormation, consulta Operazioni di base nella AWS CloudFormation Guida per l'utente.

Utilizzo di un modello di AWS CloudFormation per configurare argomenti e code in un Account AWS

Il modello di esempio crea un argomento Amazon SNS in grado di inviare messaggi a due code Amazon SQS con autorizzazioni appropriate per consentire ai membri di un gruppo IAM di pubblicare nell'argomento e a un altro gruppo di leggere messaggi dalle code. Il modello crea inoltre utenti IAM che vengono aggiunti a ogni gruppo.

Copiare il contenuto del modello in un file. È inoltre possibile scaricare il modello dalla AWS pagina Modelli di CloudFormation. Nella pagina dei modelli scegliere Sfoglia i modelli di esempio AWS servizio e quindi scegliere Amazon Simple Queue Service.

MySNSTopic è configurato per eseguire pubblicazioni in due endpoint con sottoscrizione, ovvero due code Amazon SQS (MyQueue1 e MyQueue2). MyPublishTopicGroup è un gruppo IAM i cui membri sono autorizzati a pubblicare in MySNSTopic utilizzando l'operazione API Publish o il comando sns-publish. Il modello crea gli utenti IAM MyPublishUser e MyQueueUser e fornisce loro profili e chiavi di accesso. L'utente che crea uno stack con questo modello specifica le password per i profili di accesso come parametri di input. Il modello crea chiavi di accesso per i due utenti IAM con MyPublishUserKey e MyQueueUserKey. AddUserToMyPublishTopicGroup aggiunge MyPublishUser a MyPublishTopicGroup di modo che l'utente disponga delle autorizzazioni assegnate al gruppo.

MyRDMessageQueueGroup è un gruppo IAM i cui membri sono autorizzati a leggere ed eliminare messaggi dalle due code Amazon SQS utilizzando le azioni API ReceiveMessage e DeleteMessage. AddUserToMyQueueGroup aggiunge MyQueueUser a MyRDMessageQueueGroup di modo che l'utente disponga delle autorizzazioni assegnate al gruppo. MyQueuePolicy autorizza MySNSTopic a pubblicare le relative notifiche nelle due code.

L'elenco seguente mostra il contenuto del modello AWS CloudFormation.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template SNSToSQS: This Template creates
 an SNS topic that can send messages to
```

```
 two SQS queues with appropriate permissions for one IAM user to publish to the topic
and another to read messages from the queues.
 MySNSTopic is set up to publish to two subscribed endpoints, which are two SQS queues
(MyQueue1 and MyQueue2). MyPublishUser is an IAM user
 that can publish to MySNSTopic using the Publish API. MyTopicPolicy assigns that
permission to MyPublishUser. MyQueueUser is an IAM user
 that can read messages from the two SQS queues. MyQueuePolicy assigns those
permissions to MyQueueUser. It also assigns permission for
 MySNSTopic to publish its notifications to the two queues. The template creates
access keys for the two IAM users with MyPublishUserKey
 and MyQueueUserKey. ***Warning*** you will be billed for the AWS resources used if
you create a stack from this template.",

 "Parameters": {
   "MyPublishUserPassword": {
     "NoEcho": "true",
     "Type": "String",
     "Description": "Password for the IAM user MyPublishUser",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern": "[a-zA-Z0-9]*",
     "ConstraintDescription": "must contain only alphanumeric characters."
   },
   "MyQueueUserPassword": {
     "NoEcho": "true",
     "Type": "String",
     "Description": "Password for the IAM user MyQueueUser",
     "MinLength": "1",
     "MaxLength": "41",
     "AllowedPattern": "[a-zA-Z0-9]*",
     "ConstraintDescription": "must contain only alphanumeric characters."
   }
 },


 "Resources": {
   "MySNSTopic": {
     "Type": "AWS::SNS::Topic",
     "Properties": {
       "Subscription": [{
           "Endpoint": {
             "Fn::GetAtt": ["MyQueue1", "Arn"]
           },
           "Protocol": "sqs"
```

```
        },
        {
          "Endpoint": {
            "Fn::GetAtt": ["MyQueue2", "Arn"]
          },
          "Protocol": "sqs"
        }
      ]
    }
  },
  "MyQueue1": {
    "Type": "AWS::SQS::Queue"
  },
  "MyQueue2": {
    "Type": "AWS::SQS::Queue"
  },
  "MyPublishUser": {
    "Type": "AWS::IAM::User",
    "Properties": {
      "LoginProfile": {
        "Password": {
          "Ref": "MyPublishUserPassword"
        }
      }
    }
  },
  "MyPublishUserKey": {
    "Type": "AWS::IAM::AccessKey",
    "Properties": {
      "UserName": {
        "Ref": "MyPublishUser"
      }
    }
  },
  "MyPublishTopicGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
        "PolicyName": "MyTopicGroupPolicy",
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Action": [
              "sns:Publish"
```

```
            ],
            "Resource": {
              "Ref": "MySNSTopic"
            }
          }]
        }
      }]
    }
  },
  "AddUserToMyPublishTopicGroup": {
    "Type": "AWS::IAM::UserToGroupAddition",
    "Properties": {
      "GroupName": {
        "Ref": "MyPublishTopicGroup"
      },
      "Users": [{
        "Ref": "MyPublishUser"
      }]
    }
  },
  "MyQueueUser": {
    "Type": "AWS::IAM::User",
    "Properties": {
      "LoginProfile": {
        "Password": {
          "Ref": "MyQueueUserPassword"
        }
      }
    }
  },
  "MyQueueUserKey": {
    "Type": "AWS::IAM::AccessKey",
    "Properties": {
      "UserName": {
        "Ref": "MyQueueUser"
      }
    }
  },
  "MyRDMessageQueueGroup": {
    "Type": "AWS::IAM::Group",
    "Properties": {
      "Policies": [{
        "PolicyName": "MyQueueGroupPolicy",
        "PolicyDocument": {
```

```
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
              ],
              "Resource": [{
                  "Fn::GetAtt": ["MyQueue1", "Arn"]
                },
                {
                  "Fn::GetAtt": ["MyQueue2", "Arn"]
                }
              ]
            }]
          }
        }]
      }
    },
    "AddUserToMyQueueGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
          "Ref": "MyRDMessageQueueGroup"
        },
        "Users": [{
          "Ref": "MyQueueUser"
        }]
      }
    },
    "MyQueuePolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": {
              "Service": "sns.amazonaws.com"
            },
            "Action": ["sqs:SendMessage"],
            "Resource": "*",
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": {
                  "Ref": "MySNSTopic"
```

```
              }
            }
          }
        }]
      },
      "Queues": [{
        "Ref": "MyQueue1"
      }, {
        "Ref": "MyQueue2"
      }]
    }
  }
},
"Outputs": {
  "MySNSTopicTopicARN": {
    "Value": {
      "Ref": "MySNSTopic"
    }
  },
  "MyQueue1Info": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyQueue1", "Arn"]
          },
          "URL:",
          {
            "Ref": "MyQueue1"
          }
        ]
      ]
    }
  },
  "MyQueue2Info": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyQueue2", "Arn"]
```

```
          },
          "URL:",
          {
            "Ref": "MyQueue2"
          }
        ]
      ]
    }
  },
  "MyPublishUserInfo": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyPublishUser", "Arn"]
          },
          "Access Key:",
          {
            "Ref": "MyPublishUserKey"
          },
          "Secret Key:",
          {
            "Fn::GetAtt": ["MyPublishUserKey", "SecretAccessKey"]
          }
        ]
      ]
    }
  },
  "MyQueueUserInfo": {
    "Value": {
      "Fn::Join": [
        " ",
        [
          "ARN:",
          {
            "Fn::GetAtt": ["MyQueueUser", "Arn"]
          },
          "Access Key:",
          {
            "Ref": "MyQueueUserKey"
          },
          "Secret Key:",
```

```
            {
              "Fn::GetAtt": ["MyQueueUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    }
  }
}
```

Fanout agli endpoint HTTP(S)

Puoi utilizzare Amazon SNS per inviare messaggi di notifica a uno o più endpoint HTTP o HTTPS. Quando effettui la sottoscrizione di un endpoint a un argomento, puoi pubblicare una notifica nell'argomento e Amazon SNS provvederà a inviare una richiesta HTTP POST per consegnare il contenuto della notifica all'endpoint dotato di sottoscrizione. Quando effettui la sottoscrizione dell'endpoint, scegli se Amazon SNS deve utilizzare HTTP o HTTPS per inviare la richiesta POST all'endpoint. Se utilizzi HTTPS, puoi sfruttare il supporto di Amazon SNS per quanto segue:

- Server Name Indication (SNI): consente a Amazon SNS di supportare gli endpoint HTTPS che richiedono l'estensione di protocollo SNI, ad esempio un server che richiede più certificati per ospitare più domini. Per ulteriori informazioni sull'uso di SNI, consulta Server Name Indication (SNI).

- Autenticazione di accesso di base e digest— Questo consente di specificare un nome utente e una password nell'URL HTTPS della richiesta HTTP POST, come `https://user:password@domain.com` o `https://user@domain.com` Il nome utente e la password vengono crittografati nella connessione SSL stabilita quando usi HTTPS. Solo il nome di dominio viene inviato come testo normale. Per ulteriori informazioni su Basic e Digest Access Authentication, consulta RFC-2617.

> **Important**
>
> Amazon SNS attualmente non supporta endpoint HTTP(S) privati.
> Gli URL HTTPS sono recuperabili solo dall'operazione API di Amazon SNS `GetSubscriptionAttributes`, per le entità a cui è stato concesso l'accesso alle API.

> **Note**
>
> Il servizio client deve essere in grado di supportare la risposta con intestazione `HTTP/1.1 401 Unauthorized`.

La richiesta contiene l'oggetto e il messaggio pubblicati nell'argomento insieme ai metadati relativi alla notifica in un documento JSON. La richiesta risulterà simile alla richiesta HTTP POST seguente. Per i dettagli sull'intestazione HTTP e il formato JSON del corpo della richiesta, consulta Intestazioni HTTP/HTTPS e Notifica HTTP/HTTPS in formato JSON.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
    Content-Length: 761
    Content-Type: text/plain; charset=UTF-8
    Host: ec2-50-17-44-49.compute-1.amazonaws.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" :
 "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

Argomenti

Iscrizione di un endpoint HTTP/S a un argomento

Le pagine di questa sezione descrivono come iscrivere gli endpoint HTTP/S agli argomenti Amazon SNS.

Argomenti

Fase 1: Verifica della capacità dell'endpoint di elaborare messaggi Amazon SNS

Prima di effettuare la sottoscrizione dell'endpoint HTTP o HTTPS a un argomento, devi verificare che l'endpoint sia in grado di gestire le richieste HTTP POST che Amazon SNS usa per inviare la conferma della sottoscrizione e i messaggi di notifica. In genere questo significa creare e distribuire un'applicazione Web (ad esempio, un servlet Java se l'host dell'endpoint esegue Linux con Apache e Tomcat) che elabori le richieste HTTP di Amazon SNS. Quando effettui la sottoscrizione di un endpoint HTTP, Amazon SNS invia una richiesta di conferma della sottoscrizione. L'endpoint deve essere preparato a ricevere ed elaborare la richiesta quando crei la sottoscrizione perché Amazon SNS la invia in quel momento. Amazon SNS non invierà notifiche all'endpoint fino alla conferma della sottoscrizione. Una volta confermata la sottoscrizione, Amazon SNS invierà le notifiche all'endpoint quando verrà eseguita un'operazione di pubblicazione sull'argomento sottoscritto.

Per impostare l'endpoint in modo che elabori i messaggi di conferma della sottoscrizione e di notifica

1. Il tuo codice deve leggere le intestazioni HTTP delle richieste HTTP POST che Amazon SNS invia all'endpoint, Il codice deve cercare il campo di intestazione `x-amz-sns-message-`

`type` che indica il tipo di messaggio inviato da Amazon SNS. In base all'intestazione puoi determinare il tipo di messaggio senza dover analizzare il corpo della richiesta HTTP. Sono due i tipi che devi gestire: `SubscriptionConfirmation` e `Notification`. Il messaggio `UnsubscribeConfirmation` viene utilizzato solo quando la sottoscrizione viene eliminata dall'argomento.

Per i dettagli sull'intestazione HTTP, consulta Intestazioni HTTP/HTTPS. La seguente richiesta HTTP POST è un esempio di messaggio di conferma della sottoscrizione.

```
POST / HTTP/1.1
    x-amz-sns-message-type: SubscriptionConfirmation
    x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    Content-Length: 1336
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37f...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH+...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

2. Il codice deve analizzare il documento JSON nel corpo della richiesta HTTP POST e il tipo di contenuti di testo per leggere le coppie nome/valore che costituiscono il messaggio Amazon SNS. Usa un parser JSON in grado di gestire la conversione della rappresentazione con escape dei caratteri di controllo nei corrispondenti valori ASCII (ad esempio, la conversione di \n in un carattere di nuova riga). Puoi utilizzare un parser JSON esistente come il processore

Jackson JSON o scriverne uno personalizzato. Per inviare il testo nei campi dell'oggetto e del messaggio come codice JSON valido, Amazon SNS deve convertire alcuni caratteri di controllo in rappresentazioni con escape che possono essere incluse nel documento JSON. Quando ricevi il documento JSON nel corpo della richiesta POST inviata all'endpoint, devi riconvertire i caratteri con escape nei valori originali se vuoi una rappresentazione esatta dell'oggetto e dei messaggi originali pubblicati nell'argomento. Questo è un aspetto critico se desideri verificare la firma di una notifica, perché la firma usa il messaggio e l'oggetto nel formato originale come parte della stringa di firma.

3. Il codice deve verificare l'autenticità di un messaggio di notifica, di conferma della sottoscrizione o di conferma di annullamento della sottoscrizione inviato da Amazon SNS. Utilizzando le informazioni contenute nel messaggio Amazon SNS, l'endpoint può ricreare la firma per consentirti di verificare i contenuti del messaggio mettendo in corrispondenza la tua firma con la firma che Amazon SNS ha inviato con il messaggio. Per ulteriori informazioni sulla verifica della firma di un messaggio, consulta Verifica delle firme dei messaggi Amazon SNS.

4. In base al tipo specificato dal campo di intestazione `x-amz-sns-message-type`, il codice deve leggere il documento JSON contenuto nel corpo della richiesta HTTP ed elaborare il messaggio. Di seguito sono riportate le linee guida per gestire i due tipi di messaggi principali:

   SubscriptionConfirmation

   Leggi il valore di `SubscribeURL` e visita l'URL. Per confermare la sottoscrizione e iniziare a ricevere notifiche presso l'endpoint, devi visitare l'`SubscribeURL`URL (ad esempio inviando all'URL una richiesta HTTP GET). Fai riferimento all'esempio di richiesta HTTP nella fase precedente per vedere l'aspetto di `SubscribeURL`. Per ulteriori informazioni sul formato del messaggio `SubscriptionConfirmation`, consulta Conferma sottoscrizione HTTP/HTTPS in formato JSON. Quando visiterai l'URL, otterrai una risposta simile al documento XML seguente. Il documento restituisce l'ARN della sottoscrizione dell'endpoint nell'elemento `ConfirmSubscriptionResult`.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55</
SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
   </ResponseMetadata>
```

```
</ConfirmSubscriptionResponse>
```

Anziché visitare `SubscribeURL`, puoi confermare la sottoscrizione utilizzando l'operazione ConfirmSubscription con il valore `Token` impostato sul corrispondente valore nel messaggio `SubscriptionConfirmation`. Se desideri che solo il proprietario dell'argomento e il proprietario della sottoscrizione possano annullare la sottoscrizione dell'endpoint, chiama l'operazione `ConfirmSubscription` con una firma AWS.

Notifica

Leggi i valori di `Subject` e `Message` per ottenere le informazioni sulla notifica pubblicate nell'argomento.

Per i dettagli sul formato del messaggio `Notification`, consulta Intestazioni HTTP/HTTPS. La richiesta HTTP POST seguente è un esempio di messaggio di notifica inviato all'endpoint example.com.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
    Content-Length: 773
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
```

```
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

5. Assicurati che l'endpoint risponda al messaggio HTTP POST ricevuto da Amazon SNS con il codice di stato appropriato. La connessione scade dopo 15 secondi. Se l'endpoint non risponde prima del timeout della connessione o restituisce un codice di stato non compreso nell'intervallo 200–4xx, Amazon SNS considera la consegna del messaggio come tentativo non riuscito.
6. Assicurati che il codice possa gestire tentativi ripetuti di consegna del messaggio da parte di Amazon SNS. Se Amazon SNS non riceve una risposta positiva dall'endpoint, tenta di consegnare nuovamente il messaggio. Ciò vale per tutti i messaggi, incluso il messaggio di conferma della sottoscrizione. Per default, se la consegna iniziale del messaggio non riesce, Amazon SNS ritenta fino a tre volte, con un intervallo di 20 secondi tra tentativi non riusciti.

   > **Note**
   >
   > La richiesta del messaggio scade dopo 15 secondi. Questo significa che, se la mancata consegna del messaggio è stata causata da un timeout, Amazon SNS riprova circa 35 secondi dopo il precedente tentativo non riuscito. È possibile impostare una policy di consegna diversa per l'endpoint.

   Amazon SNS utilizza il campo di intestazione `x-amz-sns-message-id` per identificare in modo univoco ogni messaggio pubblicato su un argomento Amazon SNS. Tramite il confronto degli ID dei messaggi elaborati con i messaggi in entrata, puoi determinare se il messaggio è un tentativo ripetuto.
7. Se sottoscrivi un endpoint HTTPS, assicurati che l'endpoint disponga di un certificato server di un'autorità di certificazione (CA) attendibile. Amazon SNS invia messaggi solo agli endpoint HTTPS che dispongono di un certificato server firmato da una CA che Amazon SNS ritiene attendibile.
8. Distribuisci il codice creato per ricevere i messaggi Amazon SNS. Quando effettui la sottoscrizione dell'endpoint, questo deve essere pronto a ricevere almeno il messaggio di conferma della sottoscrizione.

Fase 2: Sottoscrizione dell'endpoint HTTP/HTTPS all'argomento Amazon SNS

Per inviare messaggi a un endpoint HTTP o HTTPS tramite un argomento, devi effettuare la sottoscrizione dell'endpoint all'argomento Amazon SNS. L'endpoint deve essere specificato tramite il relativo URL. Per effettuare la sottoscrizione a un argomento, puoi utilizzare la console Amazon SNS, il comando sns-subscribe o l'operazione API Subscribe. Prima di iniziare, verifica di disporre dell'URL dell'endpoint di cui effettuare la sottoscrizione e che l'endpoint sia pronto a ricevere i messaggi di conferma e notifica, come descritto nella fase 1.

Per effettuare la sottoscrizione di un endpoint HTTP o HTTPS a un argomento utilizzando la console Amazon SNS

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Scegli l'opzione Create Subscription (Crea sottoscrizione).
4. Nell'elenco a discesa Protocol (Protocollo) seleziona HTTP o HTTPS.
5. Nella casella Endpoint incolla l'URL dell'endpoint a cui l'argomento invierà messaggi e scegli Create subscription (Crea sottoscrizione).
6. Viene visualizzato il messaggio di conferma. Scegli Close (Chiudi).

   Il campo Subscription ID (ID sottoscrizione) della nuova sottoscrizione mostra PendingConfirmation. Dopo la conferma della sottoscrizione, il campo Subscription ID (ID sottoscrizione) visualizzerà il relativo ID.

Fase 3: conferma della sottoscrizione

A seguito della sottoscrizione dell'endpoint, Amazon SNS invierà a tale endpoint un messaggio di conferma della sottoscrizione Devi avere già distribuito nell'endpoint il codice che esegue le operazioni descritte nella fase 1. Specificamente, il codice nell'endpoint deve recuperare il valore `SubscribeURL` dal messaggio di conferma della sottoscrizione e visitare automaticamente la posizione specificata da `SubscribeURL` oppure renderla disponibile in modo che tu possa visitare manualmente `SubscribeURL`, ad esempio con un Web browser. Amazon SNS non invierà messaggi all'endpoint fino alla conferma della sottoscrizione. Quando visiterai `SubscribeURL`, otterrai una risposta con un documento XML contenente un elemento `SubscriptionArn` che specifica l'ARN della sottoscrizione. Puoi anche utilizzare la console Amazon SNS per verificare che la sottoscrizione sia stata confermata: Il Subscription ID (ID sottoscrizione) mostrerà l'ARN della sottoscrizione invece del valore `PendingConfirmation` visualizzato al momento dell'aggiunta della sottoscrizione.

Fase 4: impostazione della policy di distribuzione per la sottoscrizione (opzionale)

Per default, se la consegna iniziale del messaggio non riesce, Amazon SNS ritenta fino a tre volte, con un intervallo di 20 secondi tra tentativi non riusciti. Come descritto nella fase 1, l'endpoint deve disporre di codice in grado di gestire i tentativi ripetuti di consegna dei messaggi. Impostando la policy di consegna per un argomento o una sottoscrizione puoi determinare la frequenza e l'intervallo tra i tentativi di consegna non riusciti di Amazon SNS. Puoi anche specificare il tipo di contenuto per le notifiche HTTP/S in `DeliveryPolicy`. Per ulteriori informazioni, consulta Creazione di una policy di consegna HTTP/S.

Fase 5: concessione delle autorizzazioni di pubblicazione nell'argomento agli utenti (opzionale)

Per default il proprietario dell'argomento dispone delle autorizzazioni per pubblicare nell'argomento. Per consentire ad altri utenti o applicazioni di pubblicare nell'argomento, ti consigliamo di utilizzare AWS Identity and Access Management (IAM) per concedere l'autorizzazione di pubblicazione nell'argomento. Per informazioni sull'assegnazione di autorizzazioni per le operazioni Amazon SNS agli utenti IAM, consulta Utilizzo di policy basate su identità con Amazon SNS.

Vi sono due modi di controllare l'accesso a un argomento:

- Aggiungere una policy a un utente o gruppo IAM. Il modo più semplice di concedere agli utenti le autorizzazioni per gli argomenti è creare un gruppo e aggiungere la policy appropriata al gruppo e quindi aggiungere gli utenti a quel gruppo. È molto più semplice aggiungere e rimuovere utenti da un gruppo anziché tenere traccia delle policy impostate su singoli utenti.
- Aggiungere una policy all'argomento. Se desideri concedere le autorizzazioni per un argomento a un altro account AWS, l'unico modo consiste nell'aggiungere una policy che abbia come principale il Account AWS a cui concedere le autorizzazioni.

Il primo metodo deve essere utilizzato nella maggior parte dei casi (applicare policy a gruppi e gestire le autorizzazioni per gli utenti aggiungendo o rimuovendo gli utenti appropriati ai gruppi). Se hai la necessità di concedere autorizzazioni a un utente in un altro account, utilizza il secondo metodo.

Se aggiungessi la policy seguente a un utente o gruppo IAM autorizzeresti quell'utente o i membri di quel gruppo a eseguire l'operazione `sns:Publish` sull'argomento MyTopic.

```
{
  "Statement":[{
```

```
    "Sid":"AllowPublishToMyTopic",
    "Effect":"Allow",
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

La policy di esempio seguente mostra come concedere a un altro account le autorizzazioni per un argomento.

> **Note**
>
> Quando concedi a un altro Account AWS l'accesso a una risorsa nel tuo account, lo concedi anche agli utenti IAM che dispongono di autorizzazioni di accesso di livello amministratore (accesso generico). L'accesso alla risorsa viene automaticamente negato a tutti gli altri utenti IAM nell'altro account. Se intendi concedere l'accesso alla risorsa a specifici utenti IAM in quel Account AWS, l'account o un utente IAM con accesso di livello amministratore deve delegare le autorizzazioni per la risorsa a quegli utenti IAM. Per ulteriori informazioni sulla delega multiaccount, consulta la sezione relativa all'abilitazione dell'accesso multiaccount nella Guida all'uso di IAM.

Se aggiungi la policy seguente a un argomento MioArgomento nell'account 123456789012, autorizzi l'account 111122223333 a eseguire l'operazione `sns:Publish` su quell'argomento.

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Fase 6: invio di messaggi all'endpoint HTTP/HTTPS

Puoi inviare un messaggio alle sottoscrizioni di un argomento pubblicando nell'argomento. Per pubblicare in un argomento, puoi utilizzare la console Amazon SNS, il comando della CLI `sns-publish` o l'operazione API `Publish`.

Se hai seguito le istruzioni nella fase 1, il codice distribuito all'endpoint è in grado di elaborare la notifica.

Pubblicazione di un argomento utilizzando la console Amazon SNS

1. Utilizzando le credenziali del Account AWS o di un utente IAM con l'autorizzazione da pubblicare nell'argomento, accedi alla AWS Management Console e apri la console Amazon SNS all'indirizzo https://console.aws.amazon.com/sns/.
2. Nel riquadro di navigazione, selezionare Topics (Argomenti) e scegli un argomento.
3. Scegliere il pulsante Publish message (Pubblica messaggio).
4. Nella casella Subject (Oggetto), immetti un oggetto (ad esempio **`Testing publish to my endpoint`**).
5. Nella casella Message (Messaggio), immetti del testo (ad esempio, **`Hello world!`**) e scegli Publish message (Pubblica messaggio).

   Viene visualizzato il messaggio "Your message has been successfully published" (Il messaggio è stato pubblicato).

Verifica delle firme dei messaggi Amazon SNS

Per verificare l'autenticità di un messaggio inviato all'endpoint HTTP da Amazon SNS, puoi verificare la firma del messaggio. Esistono due casi in cui si consiglia di verificare l'autenticità del messaggio. Innanzitutto, quando Amazon SNS invia un messaggio all'endpoint HTTP indicante che ti sei iscritto a un argomento. In secondo luogo, quando Amazon SNS invia un messaggio di conferma all'endpoint HTTP al momento dell'esecuzione delle operazioni API `Subscribe` o `Unsubscribe`.

Per verificare i messaggi inviati da Amazon SNS, è consigliabile procedere come segue:

- Usa sempre il protocollo HTTPS per recuperare il certificato da Amazon SNS.
- Convalida l'autenticità del certificato.
- Verifica che il certificato provenga da Amazon SNS.

- Quando possibile, utilizza uno degli SDK AWS per Amazon SNS supportati per convalidare e verificare i messaggi.
- Verifica che i messaggi Amazon SNS siano stati ricevuti dal `TopicArn` desiderato.

Amazon SNS supporta due versioni di firma dei messaggi:

- `SignatureVersion`1: Amazon SNS crea la firma in base all'hash SHA1 del messaggio.
- `SignatureVersion`2: Amazon SNS crea la firma in base all'hash SHA256 del messaggio.

Per configurare la versione della firma dei messaggi sugli argomenti di Amazon SNS

Per impostazione predefinita, gli argomenti di Amazon SNS utilizzano `SignatureVersion` 1. Per scegliere l'algoritmo di hash sull'argomento Amazon SNS, ovvero `SignatureVersion` 1 (SHA1) o `SignatureVersion` 2 (SHA256), puoi usare l'operazione API `SetTopicAttributes`.

L'esempio di codice seguente mostra come impostare l'attributo `SignatureVersion` dell'argomento mediante la AWS CLI:

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

Per verificare la firma di un messaggio Amazon SNS quando utilizzi richieste basate su query HTTP

1. Estrai le coppie nome/valore dal documento JSON nel corpo della richiesta HTTP POST che Amazon SNS ha inviato all'endpoint. Utilizzerai i valori di alcune delle coppie nome/valore per creare la stringa di firma. Quando verifichi la firma di un messaggio Amazon SNS, è essenziale convertire i caratteri di controllo con escape nelle rappresentazioni di carattere originali nei valori `Message` e `Subject`. Questi valori devono avere il formato originale quando li utilizzi come parte della stringa di firma. Per informazioni su come analizzare il documento JSON, consulta Fase 1: Verifica della capacità dell'endpoint di elaborare messaggi Amazon SNS.

   `SignatureVersion` indica la versione della firma utilizzata da Amazon SNS per generare la firma del messaggio. che ti consente di determinare i requisiti per la generazione della firma. Per le notifiche, Amazon SNS attualmente supporta la versione di firma 1 e 2. Questa sezione contiene la procedura per verificare una firma usando qieste versioni.

2. Ottieni il certificato X509 che Amazon SNS ha utilizzato per firmare il messaggio. Il valore `SigningCertURL` punta alla posizione del certificato X509 utilizzato per creare la firma digitale del messaggio. Recupera il certificato da questa posizione.
3. Estrai la chiave pubblica dal certificato. La chiave pubblica del certificato specificato da `SigningCertURL` viene utilizzata per verificare l'autenticità e l'integrità del messaggio.
4. Determina il tipo di messaggio. Il formato della stringa di firma dipende dal tipo di messaggio, specificato dal valore `Type`.
5. Crea la stringa di firma. La stringa di firma è un elenco delimitato da caratteri di nuova riga contenente coppie nome/valore tratte dal messaggio. Ogni coppia nome/valore è rappresentata con il nome per primo, seguito da un carattere di nuova riga, seguito dal valore e terminante con un carattere di nuova riga. Le coppie nome/valore devono essere elencate in ordine di byte.

   In base al tipo di messaggio, la stringa di firma deve contenere le coppie nome/valore seguenti.

   Notifica

   I messaggi di notifica devono contenere le coppie nome/valore seguenti:

   ```
   Message
   MessageId
   Subject (if included in the message)
   Timestamp
   TopicArn
   Type
   ```

   Di seguito è riportata una stringa di firma di esempio per un messaggio di tipo `Notification`.

   ```
   Message
   My Test Message
   MessageId
   4d4dc071-ddbf-465d-bba8-08f81c89da64
   Subject
   My subject
   Timestamp
   2019-01-31T04:37:04.321Z
   TopicArn
   arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
   Type
   ```

```
Notification
```

SubscriptionConfirmation e UnsubscribeConfirmation

I messaggi `SubscriptionConfirmation` e `UnsubscribeConfirmation` devono contenere le coppie nome/valore seguenti:

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

Di seguito è riportata una stringa di firma di esempio per un messaggio di tipo `SubscriptionConfirmation`.

```
Message
My Test Message
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-2:123456789012:s4-
MySNSTopic-1G1WEFCOXTC0P&Token=233...
Timestamp
2019-01-31T19:25:13.719Z
Token
233...
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

6. Decodifica il valore `Signature` dal formato Base64. Il messaggio consegna la firma nel valore `Signature`, codificato come Base64. Prima di confrontare il valore della firma con la firma calcolata, assicurati di decodificare il valore `Signature` da Base64 per effettuare il confronto utilizzando lo stesso formato.
7. Genera il valore hash derivato del messaggio Amazon SNS. Invia il messaggio Amazon SNS, in formato canonico, alla stessa funzione di hash utilizzata per generare la firma.

a. Se `SignatureVersion` è impostata su 1, utilizza SHA1 come algoritmo hash.

b. Se `SignatureVersion` è impostata su 2, utilizza SHA256 come algoritmo hash.

8. Genera il valore hash dichiarato del messaggio Amazon SNS. Il valore hash dichiarato è il risultato dell'uso del valore della chiave pubblica (fase 3) per decrittografare la firma consegnata con il messaggio Amazon SNS.
9. Verifica l'autenticità e l'integrità del messaggio Amazon SNS. Confronta il valore hash derivato (fase 7) con il valore hash dichiarato (fase 8). Se i valori sono identici, il ricevitore ha la certezza che il messaggio non è stato modificato in transito e che è stato originato da Amazon SNS. Se i valori non sono identici, il messaggio non può essere considerato attendibile dal ricevitore.

Analisi dei formati di messaggi

Amazon SNS utilizza i seguenti formati.

Argomenti

Intestazioni HTTP/HTTPS

Quando Amazon SNS manda un messaggio di conferma o annullamento dell'iscrizione o di notifica agli endpoint HTTP/HTTPS, invia un messaggio POST con un numero di valori d'intestazione specifici di Amazon SNS. Puoi utilizzare i valori d'intestazione per attività quali l'identificazione del tipo di messaggio senza dover analizzare il corpo del messaggio JSON per leggere il valore `Type`. Per impostazione predefinita, Amazon SNS invia tutte le notifiche agli endpoint HTTP/S con `Content-Type` impostato su `text/plain; charset=UTF-8`. Per scegliere un `Content-Type` diverso da text/plain (impostazione predefinita), consultare `headerContentType` in Creazione di una policy di consegna HTTP/S.

`x-amz-sns-message-type`

Il tipo di messaggio. I valori possibili sono `SubscriptionConfirmation`, `Notification` e `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per una notifica che Amazon SNS reinvia durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

`x-amz-sns-topic-arn`

L'Amazon Resource Name (ARN) per l'argomento in cui questo messaggio è stato pubblicato.

`x-amz-sns-subscription-arn`

L'ARN per la sottoscrizione a questo endpoint.

L'intestazione HTTP POST seguente è un esempio di intestazione per un messaggio `Notification` inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

Conferma sottoscrizione HTTP/HTTPS in formato JSON

Una volta effettuata la sottoscrizione all'endpoint HTTP/HTTPS, Amazon SNS invia a esso un messaggio di conferma. Tale messaggio contiene un valore `SubscribeURL` da selezionare per confermare la sottoscrizione. In alternativa, puoi utilizzare il valore `Token` con `ConfirmSubscription`.

> **Note**
>
> Amazon SNS non invierà notifiche a questo endpoint fino alla conferma della sottoscrizione

Il messaggio di conferma della sottoscrizione è un messaggio POST con un corpo che contiene un documento JSON con le seguenti coppie nome/valore.

`Type`

Il tipo di messaggio. Per confermare la sottoscrizione, il tipo è `SubscriptionConfirmation`.

`MessageId`

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per un messaggio che Amazon SNS invia di nuovo durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

`Token`

Un valore che puoi utilizzare con l'operazione `ConfirmSubscription` per confermare la sottoscrizione. In alternativa, puoi selezionare `SubscribeURL`.

`TopicArn`

L'Amazon Resource Name (ARN) per l'argomento a cui questo endpoint è sottoscritto.

`Message`

Una stringa che descrive il messaggio. Per una conferma di sottoscrizione, la stringa deve avere il seguente aspetto:

```
You have chosen to subscribe to the topic arn:aws:sns:us-
east-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.
```

`SubscribeURL`

L'URL da selezionare per confermare la sottoscrizione. In alternativa, puoi utilizzare il `Token` con l'operazione `ConfirmSubscription` per confermare la sottoscrizione.

`Timestamp`

L'ora (GMT) in cui è stata inviata la conferma della sottoscrizione.

`SignatureVersion`

La versione della firma Amazon SNS utilizzata.

- Se `SignatureVersion` è 1, `Signature` è una firma `SHA1withRSA` con codifica Base64 dei valori `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.
- Se `SignatureVersion` è 2, `Signature` è una firma `SHA256withRSA` con codifica Base64 dei valori `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.

`Signature`

Firma `SHA1withRSA` o `SHA256withRSA` con codifica Base64 dei valori `Message`, `MessageIdType`, `Timestamp` e `TopicArn`.

`SigningCertURL`

L'URL per il certificato utilizzato per firmare il messaggio.

Il seguente messaggio HTTP POST è un esempio di messaggio `SubscriptionConfirmation` inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
```

```
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

Notifica HTTP/HTTPS in formato JSON

Quando Amazon SNS invia una notifica a un endpoint HTTP o HTTPS sottoscritto, il corpo del messaggio POST inviato all'endpoint contiene un documento JSON con le seguenti coppie nome/valore.

`Type`

Il tipo di messaggio. Per una notifica, il tipo è `Notification`.

`MessageId`

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per una notifica che Amazon SNS reinvia durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

`TopicArn`

L'Amazon Resource Name (ARN) per l'argomento in cui questo messaggio è stato pubblicato.

`Subject`

Il parametro `Subject` specificato quando la notifica è stata pubblicata nell'argomento.

> **Note**
>
> Si tratta di un parametro facoltativo. Se non è stato specificato alcun `Subject`, questa coppia nome/valore non appare nel documento JSON.

`Message`

Il valore `Message` specificato nel momento in cui la notifica è stata pubblicata nell'argomento.

`Timestamp`

L'ora (GMT) di pubblicazione della notifica.

`SignatureVersion`

La versione della firma Amazon SNS utilizzata.

- Se `SignatureVersion` è 1, `Signature` è una firma `SHA1withRSA` con codifica Base64 dei valori `Message`, `MessageId`, `Subject` (se presente), `Type`, `Timestamp` e `TopicArn`.
- Se `SignatureVersion` è 2, `Signature` è una firma `SHA256withRSA` con codifica Base64 dei valori `Message`, `MessageId`, `Subject` (se presente), `Type`, `Timestamp` e `TopicArn`.

`Signature`

Firma `SHA1withRSA` o `SHA256withRSA` con codifica Base64 dei valori `Message`, `MessageIdSubject` (se presente), `Type`, `Timestamp` e `TopicArn`.

`SigningCertURL`

L'URL per il certificato utilizzato per firmare il messaggio.

`UnsubscribeURL`

Un URL che puoi utilizzare per annullare la sottoscrizione dell'endpoint da questo argomento. Se visiti questo URL, Amazon SNS annulla la sottoscrizione all'endpoint e interrompe l'invio di notifiche a esso.

Il seguente messaggio HTTP POST è un esempio di messaggio `Notification` inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
```

```
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
}
```

Conferma annullamento sottoscrizione HTTP/HTTPS in formato JSON

Dopo l'annullamento della sottoscrizione di un endpoint HTTP/HTTPS da un argomento, Amazon SNS invia a esso un messaggio di conferma dell'annullamento.

Il messaggio di conferma dell'annullamento della sottoscrizione è un messaggio POST con un corpo che contiene un documento JSON con le seguenti coppie nome/valore.

`Type`

Il tipo di messaggio. Per confermare l'annullamento della sottoscrizione, il tipo è `UnsubscribeConfirmation`.

`MessageId`

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per un messaggio che Amazon SNS invia di nuovo durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

`Token`

Un valore che puoi utilizzare con l'operazione `ConfirmSubscription` per confermare nuovamente la sottoscrizione. In alternativa, puoi selezionare `SubscribeURL`.

`TopicArn`

L'Amazon Resource Name (ARN) per l'argomento da cui è stato effettuato l'annullamento della sottoscrizione di questo endpoint.

`Message`

Una stringa che descrive il messaggio. La stringa per la conferma di annullamento della sottoscrizione ha il seguente aspetto:

```
You have chosen to deactivate subscription arn:aws:sns:us-
east-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
```

```
operation and restore the subscription, visit the SubscribeURL included in this
message.
```

`SubscribeURL`

L'URL da selezionare per confermare nuovamente la sottoscrizione. In alternativa, puoi utilizzare il `Token` con l'operazione `ConfirmSubscription` per confermare nuovamente la sottoscrizione.

`Timestamp`

L'ora (GMT) in cui è stata inviata la conferma di annullamento della sottoscrizione.

`SignatureVersion`

La versione della firma Amazon SNS utilizzata.

- Se `SignatureVersion` è 1, `Signature` è una firma `SHA1withRSA` con codifica Base64 dei valori `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.
- Se `SignatureVersion` è 2, `Signature` è una firma `SHA256withRSA` con codifica Base64 dei valori `Message`, `MessageId`, `Type`, `Timestamp` e `TopicArn`.

`Signature`

Firma `SHA1withRSA` o `SHA256withRSA` con codifica Base64 dei valori `Message`, `MessageIdType`, `Timestamp` e `TopicArn`.

`SigningCertURL`

L'URL per il certificato utilizzato per firmare il messaggio.

Il seguente messaggio HTTP POST è un esempio di messaggio `UnsubscribeConfirmation` inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

```
{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37fb6...",
  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJm...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

Policy di consegna di SetSubscriptionAttributes in formato JSON

Se invii una richiesta all'operazione `SetSubscriptionAttributes` e imposti il parametro `AttributeName` su un valore di `DeliveryPolicy`, il valore del parametro `AttributeValue` deve essere un oggetto JSON valido. Per esempio, il caso seguente imposta la policy di consegna su 5 tentativi totali.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetSubscriptionAttributes
&SubscriptionArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
%3A80289ba6-0fd4-4079-afb4-ce8c8260f0ca
&AttributeName=DeliveryPolicy
&AttributeValue={"healthyRetryPolicy":{"numRetries":5}}
...
```

Utilizza il seguente formato JSON per il valore del parametro `AttributeValue`.

```
{
    "healthyRetryPolicy" : {
        "minDelayTarget" :  int,
        "maxDelayTarget" : int,
        "numRetries" : int,
```

```
        "numMaxDelayRetries" : int,
        "backoffFunction" : "linear|arithmetic|geometric|exponential"
    },
    "throttlePolicy" : {
        "maxReceivesPerSecond" : int
    },
    "requestPolicy" : {
        "headerContentType" : "text/plain | application/json | application/xml"
    }
}
```

Per ulteriori informazioni sull'operazione `SetSubscriptionAttribute`, passa a SetTopicAttributes nella Documentazione di riferimento dell'API di Amazon Simple Notification Service. Per ulteriori informazioni sulle intestazioni dei tipi di contenuto HTTP supportate, consultare Creazione di una policy di consegna HTTP/S.

Policy di consegna SetTopicAttributes in formato JSON

Se invii una richiesta all'operazione `SetTopicAttributes` e imposti il parametro `AttributeName` su un valore di `DeliveryPolicy`, il valore del parametro `AttributeValue` deve essere un oggetto JSON valido. Per esempio, il caso seguente imposta la policy di consegna su 5 tentativi totali.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetTopicAttributes
&TopicArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
&AttributeName=DeliveryPolicy
&AttributeValue={"http":{"defaultHealthyRetryPolicy":{"numRetries":5}}}
...
```

Utilizza il seguente formato JSON per il valore del parametro `AttributeValue`.

```
{
    "http" : {
        "defaultHealthyRetryPolicy" : {
            "minDelayTarget":  int,
            "maxDelayTarget": int,
            "numRetries": int,
            "numMaxDelayRetries": int,
            "backoffFunction": "linear|arithmetic|geometric|exponential"
        },
        "disableSubscriptionOverrides" : Boolean,
        "defaultThrottlePolicy" : {
```

```
            "maxReceivesPerSecond" : int
        },
        "defaultRequestPolicy" : {
            "headerContentType" : "text/plain | application/json | application/xml"
        }
    }
}
```

Per ulteriori informazioni sulla `SetTopicAttribute` azione, vai a SetTopicAttributes nella Amazon Simple Notification Service API reference. Per ulteriori informazioni sulle intestazioni dei tipi di contenuto HTTP supportate, consultare Creazione di una policy di consegna HTTP/S.

Da Fanout a AWS Pipeline Even Fork

Per l'archiviazione e l'analisi degli eventi, Amazon SNS ora consiglia di utilizzare la sua integrazi one nativa con Amazon Data Firehose. Puoi abbonare i flussi di distribuzione Firehose agli argomenti SNS, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift, Amazon Service (Service) e altro ancora. OpenSearch OpenSearch L'uso di Amazon SNS con i flussi di distribuzione Firehose è una soluzione completamente gestita e priva di codice che non richiede l'utilizzo di funzioni. AWS Lambda Per ulteriori informazioni, consulta Flussi di distribuzione da Fanout a Firehose.

È possibile utilizzare Amazon SNS per creare applicazioni basate su eventi che utilizzano i servizi del sottoscrittore per eseguire automaticamente il lavoro in risposta a eventi attivati dai servizi del publisher. Questo modello di architettura può rendere i servizi più riutilizzabili, interoperabili e scalabili. Tuttavia, può essere impegnativo eseguire il forking dell'elaborazione di eventi in pipeline rivolte ai requisiti di gestione di eventi comuni, come l'archiviazione, il backup, la ricerca, l'analisi dei dati e la riproduzione.

Per accelerare lo sviluppo delle applicazioni basate sugli eventi, è possibile sottoscrivere pipeline di gestione degli eventi, basate su AWSPipeline di fork degli eventi: per gli argomenti Amazon SNS. AWS Event Fork Pipelines è una suite di applicazioni nidificate open source, basata su AWS Serverless Application Model (AWS SAM), che puoi implementare direttamente dalla suite AWS Event Fork Pipelines, scegliendo Show apps that create custom IAM roles or resource policies (Visualizza le app che creano ruoli IAM personalizzati o policy delle risorse), nel tuo account AWS.

Per un AWS caso d'uso delle pipeline di Event Fork, consulta Distribuzione e test di AWS Applicazione di esempio Pipeline Event Fork.

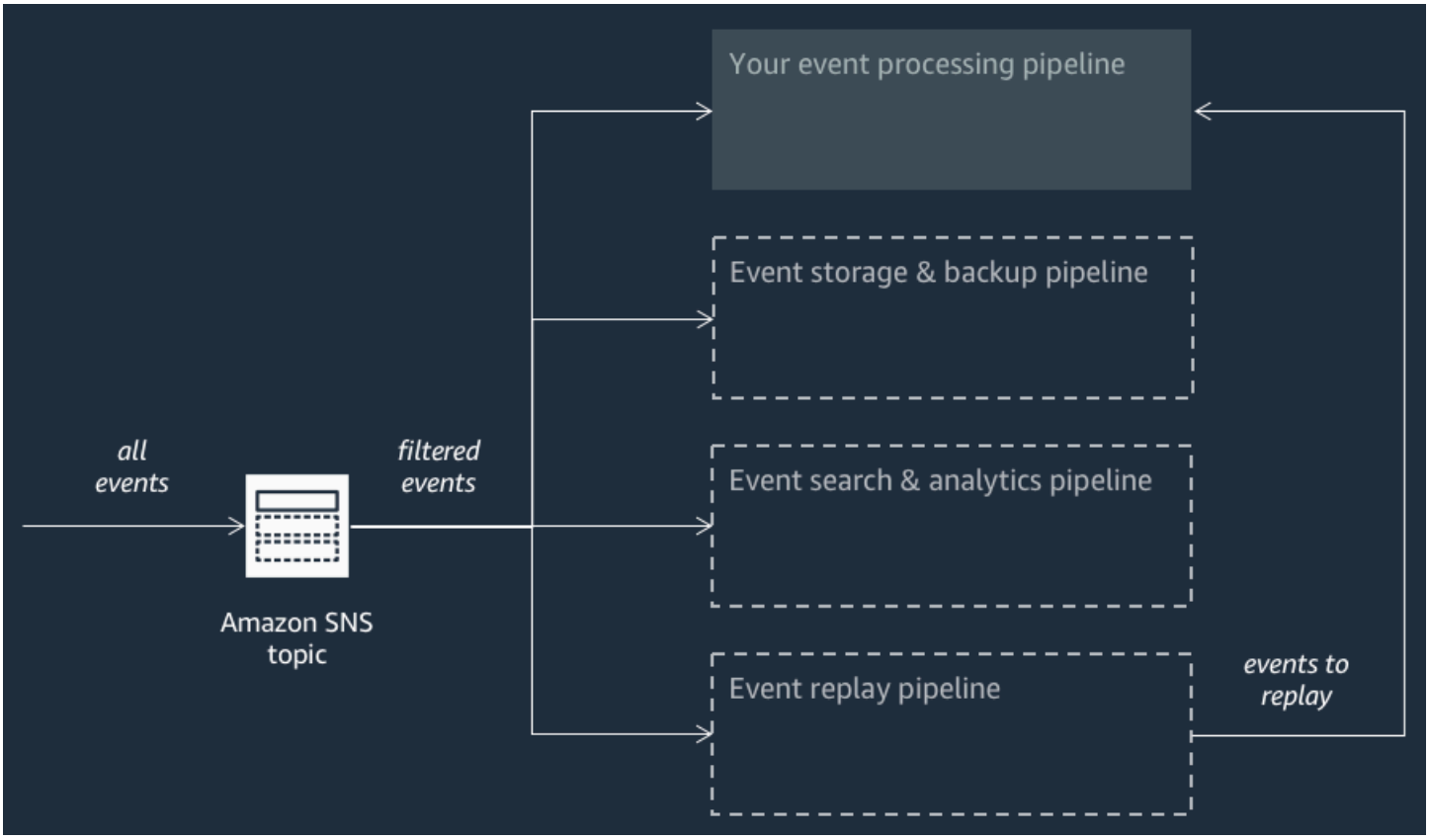
Argomenti

Come AWS funziona Event Fork Pipelines

AWS Event Fork Pipelines è un modello di progettazione serverless. Tuttavia, è anche una suite di applicazioni serverless nidificate basate su SAM AWS (che si può distribuire direttamente da AWS Serverless Application Repository, SAR AWS, al tuo Account AWS per arricchire le piattaforme basate su eventi). È possibile distribuire queste applicazioni nidificate individualmente, come richiesto dall'architettura.

Argomenti

Il seguente diagramma mostra un'applicazione Event Fork Pipelines AWS completata da tre applicazioni nidificate. È possibile distribuire una qualsiasi delle pipeline dalla AWS suite Event Fork Pipelines in AWS SAR in modo indipendente, come richiesto dall'architettura.



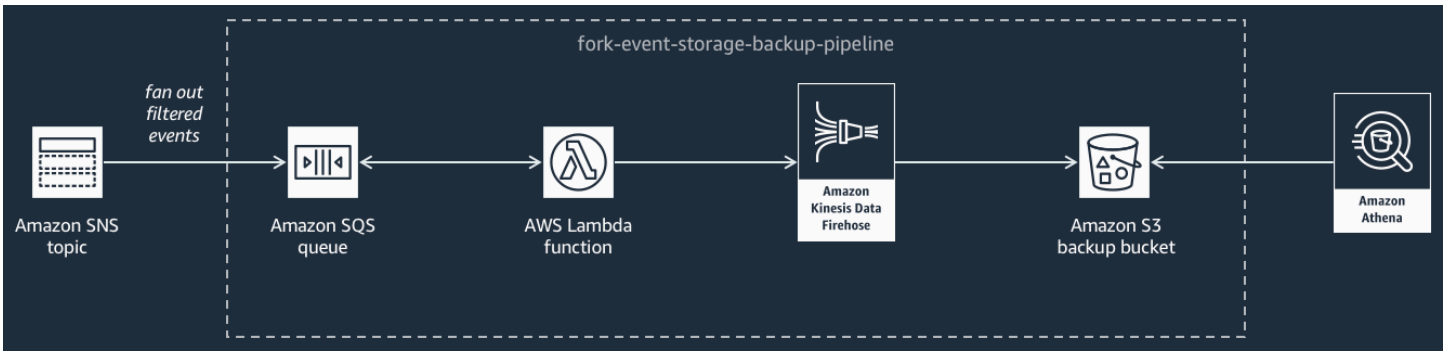
Ogni pipeline è sottoscritta allo stesso argomento Amazon SNS, facendo così in modo di elaborare eventi non appena vengono pubblicati nell'argomento. Ogni pipeline è indipendente ed è in grado di impostare la propria Policy di filtro per le sottoscrizioni. In questo modo una pipeline può elaborare solo un sottoinsieme di eventi a cui è interessata (invece che tutti gli eventi pubblicati nell'argomento).

> **Note**
>
> Poiché le tre AWS Event Fork Pipeline vengono posizionate accanto alle normali pipeline di elaborazione degli eventi (possibilmente già sottoscritte all'argomento Amazon SNS), non occorre modificare alcuna porzione del publisher del messaggio per sfruttare Event Fork Pipelines AWS nei flussi di lavoro esistenti.

La pipeline di archiviazione di eventi e di backup

Il seguente diagramma mostra la Pipeline di archiviazione di eventi e di backup. Puoi sottoscrivere questa pipeline al tuo argomento Amazon SNS per eseguire il backup automatico degli eventi che scorrono nel sistema.

Questa pipeline è composta da una coda Amazon SQS che memorizza nel buffer gli eventi forniti dall'argomento Amazon SNS, AWS Lambda una funzione che esegue automaticamente il polling di questi eventi nella coda e li inserisce in un flusso Amazon Data Firehose e un bucket Amazon S3 che esegue il backup durevole degli eventi caricati dallo stream.

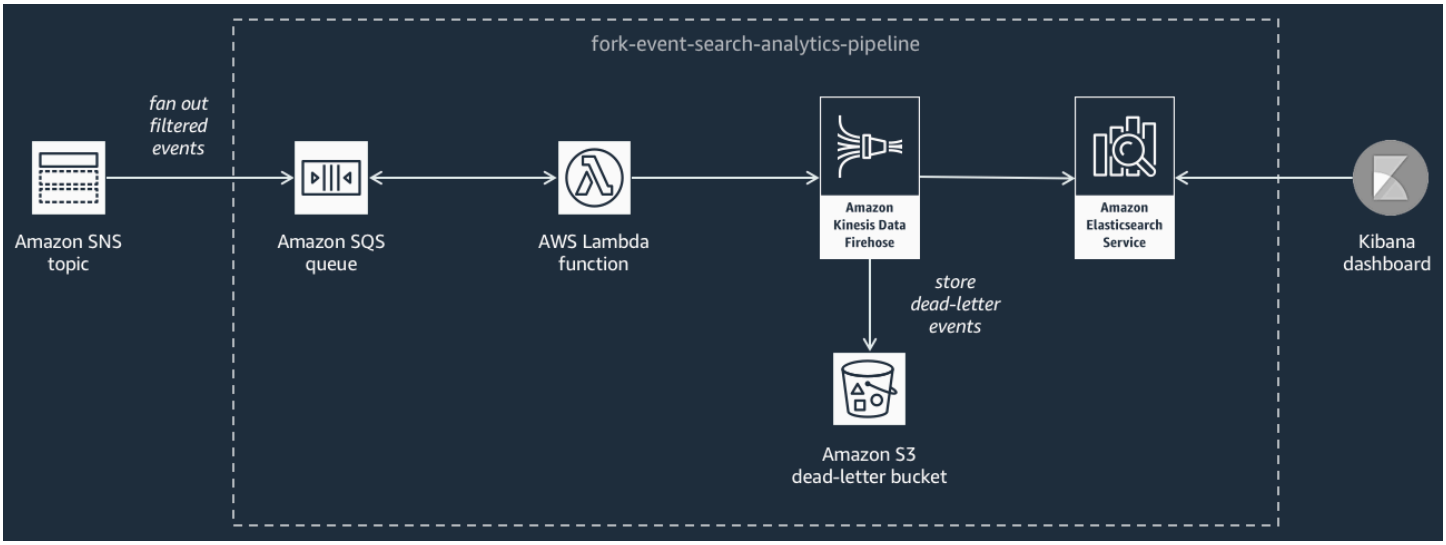


Per affinare il comportamento del flusso Firehose, puoi configurarlo in modo che trasformi, comprima ed esegua il buffering degli eventi prima di caricarli nel bucket. Man mano che gli eventi vengono caricati, puoi utilizzare Amazon Athena per eseguire query sul bucket utilizzando query SQL standard. Puoi anche configurare la pipeline in modo che riutilizzi un bucket Amazon S3 esistente o ne crei uno nuovo.

La pipeline di ricerca di eventi e di analisi dei dati

Il seguente diagramma mostra la Pipeline di ricerca di eventi e di analisi dei dati. Puoi sottoscrivere questa pipeline al tuo argomento Amazon SNS per indicizzare gli eventi che scorrono nel sistema in un dominio di ricerca e per eseguirvi le analisi dei dati.

Questa pipeline è composta da una coda Amazon SQS che memorizza nel buffer gli eventi forniti dall'argomento Amazon SNS, AWS Lambda una funzione che analizza gli eventi dalla coda e li inserisce in un flusso Amazon Data Firehose, un dominio Amazon OpenSearch Service che indicizza gli eventi caricati dal flusso Firehose e un bucket Amazon S3 che memorizza gli eventi con lettera morta che non può essere indicizzato nel dominio di ricerca.



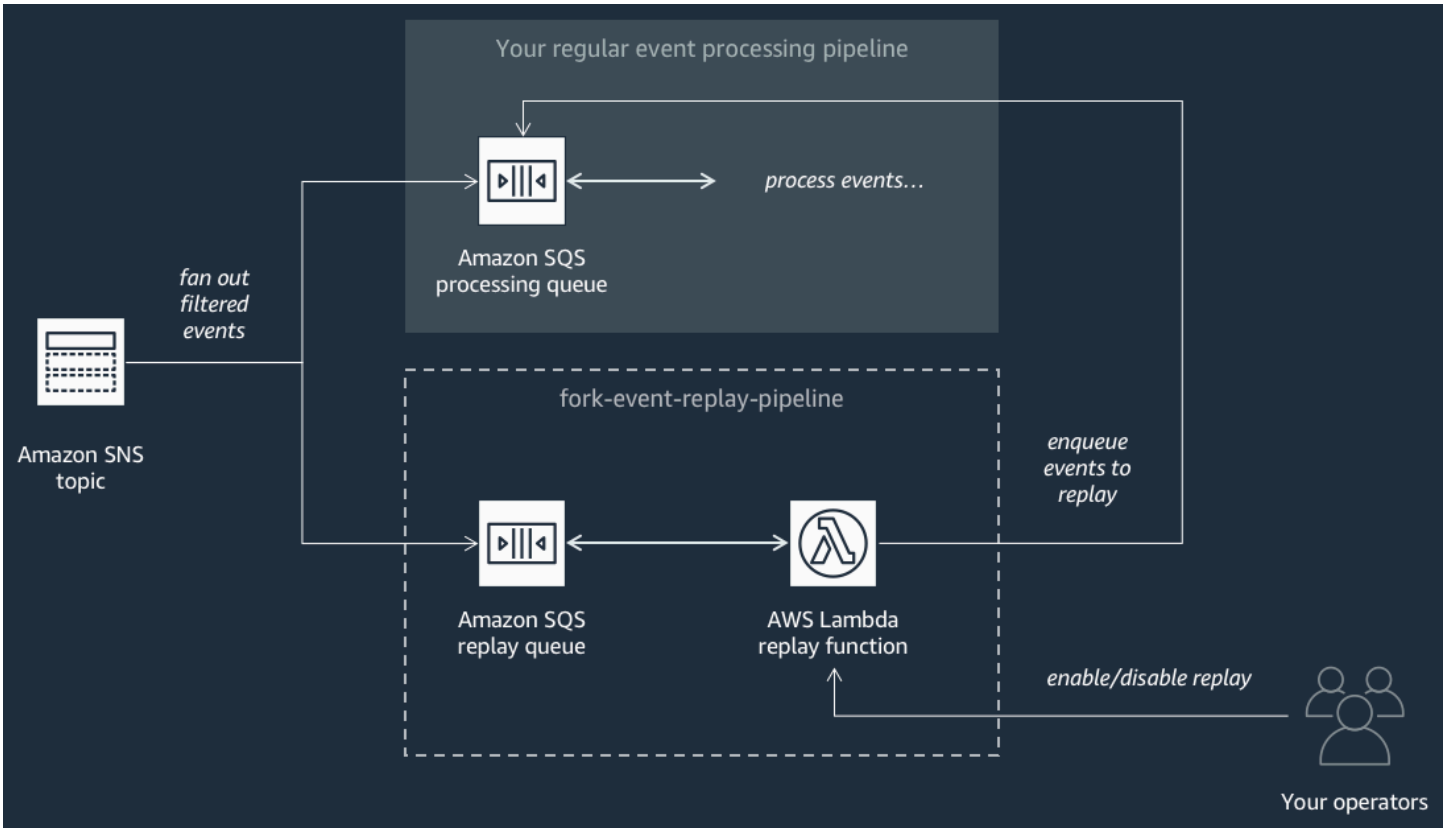
Per affinare il flusso Firehose in termini di buffering, trasformazione e compressione degli eventi, puoi configurare questa pipeline.

Puoi anche configurare se la pipeline deve riutilizzare un OpenSearch dominio esistente nel tuo Account AWS o crearne uno nuovo per te. Man mano che gli eventi vengono indicizzati nel dominio di ricerca, puoi utilizzare Kibana per eseguire l'analisi dei dati sugli eventi e aggiornare i pannelli di controllo visivi in tempo reale.

Pipeline di riproduzione eventi

Il seguente diagramma mostra la Pipeline di riproduzione eventi. Per registrare gli eventi elaborati dal sistema negli ultimi 14 giorni (ad esempio quando la piattaforma deve essere ripristinata in seguito a un guasto), puoi sottoscrivere questa pipeline all'argomento Amazon SNS e rielaborare gli eventi.

Questa pipeline è composta da una coda Amazon SQS che esegue il buffering degli eventi distribuiti dall'argomento Amazon SNS e da una funzione AWS Lambda che esegue il polling degli eventi dalla coda e li indirizza nella normale pipeline di elaborazione degli eventi, che è già sottoscritta all'argomento.



> **Note**
>
> Per impostazione predefinita, la funzione di riproduzione è disattivata e gli eventi non vengono reindirizzati. Se devi rielaborare gli eventi, devi abilitare la coda di riproduzione Amazon SQS come origine dell'evento per la funzione di riproduzione AWS Lambda.

Distribuzione di AWS Pipeline Event Fork

La AWS suite Event Fork Pipeline (seleziona Show apps that create custom IAM roles or resource policies (Mostra app in grado di creare ruoli IAM o policy delle risorse personalizzati)) è disponibile come gruppo di applicazioni pubbliche in AWS Serverless Application Repository, da cui puoi distribuirle e testarle manualmente utilizzando la AWS Lambda console. Per informazioni sulla distribuzione di pipeline tramite la console AWS Lambda, consulta Sottoscrizione ad un AWS Event Fork Pipeline per un argomento Amazon SNS.

In uno scenario di produzione, ti consigliamo di incorporare AWS Event Fork Pipeline nel modello generale SAM AWS dell'applicazione. La caratteristica di applicazione nidificata ti permette di eseguire questa operazione aggiungendo la risorsa `AWS::Serverless::Application` al

modello AWS SAM, facendo riferimento a AWS SAR `ApplicationId` e a `SemanticVersion` dell'applicazione nidificata.

Ad esempio, puoi utilizzare la pipeline di archiviazione degli eventi e di backup come applicazione nidificata aggiungendo il seguente frammento YAML alla sezione `Resources` del modello AWS SAM.

```
Backup:
  Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-2:123456789012:applications/fork-
event-storage-backup-pipeline
      SemanticVersion: 1.0.0
    Parameters:
      #The ARN of the Amazon SNS topic whose messages should be backed up to the Amazon
 S3 bucket.
      TopicArn: !Ref MySNSTopic
```

Quando specifichi i valori dei parametri, puoi utilizzare le funzioni intrinseche AWS CloudFormation per fare riferimento ad altre risorse nel modello. Ad esempio, nel frammento YAML precedente, il parametro `TopicArn` fa riferimento alla risorsa `AWS::SNS::Topic` `MySNSTopic`, definita altrove nel modello AWS SAM. Per ulteriori informazioni, consulta la Riferimento funzione intrinseca nella AWS CloudFormation Guida per l'utente.

> **Note**
>
> La pagina della console AWS Lambda per l'applicazione AWS SAR include il pulsante Copy as SAM Resource (Copia come risorsa SAM), che copia negli appunti il codice YAML necessario a nidificare un'applicazione AWS SAR.

Distribuzione e test di AWS Applicazione di esempio Pipeline Event Fork

Per accelerare lo sviluppo delle applicazioni basate sugli eventi, è possibile sottoscrivere pipeline di gestione degli eventi, basate su AWSPipeline Event Fork: per gli argomenti Amazon SNS. AWS Event Fork Pipelines è una suite di applicazioni nidificate open source, basata su AWS Serverless Application Model (AWS SAM), che puoi implementare direttamente dalla suite AWS Event Fork Pipelines, scegliendo Show apps that create custom IAM roles or resource policies (Visualizza le

app che creano ruoli IAM personalizzati o policy delle risorse), nel tuo account AWS. Per ulteriori informazioni, consulta Come AWS funziona Event Fork Pipelines.

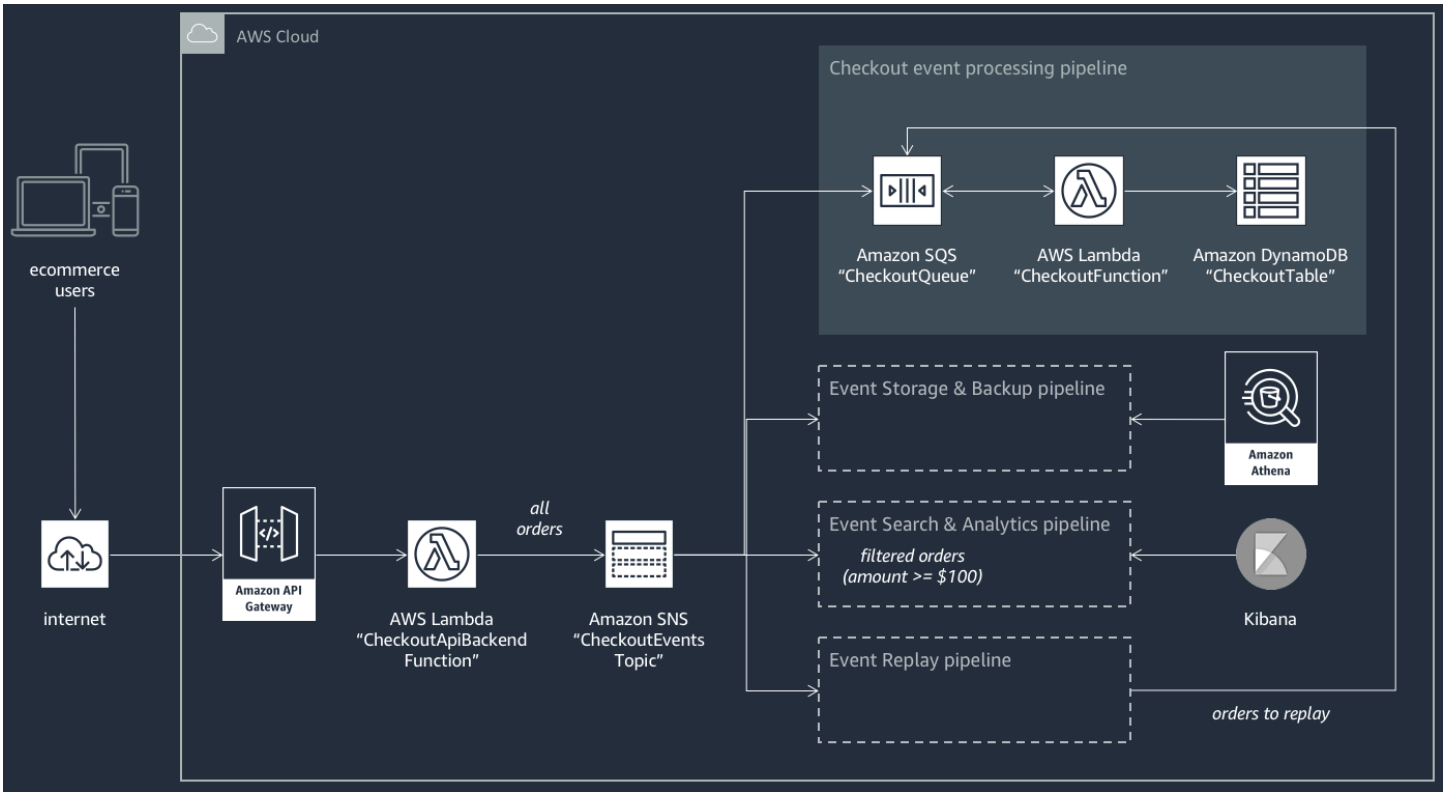
Questa pagina mostra come puoi utilizzare le AWS Management Console per distribuire e testare la AWSaApplicazione di esempio Event Fork Pipelines.

> **Important**
>
> Per evitare costi indesiderati, una volta completata la distribuzione dell'applicazione AWS Event Fork Pipelines di esempio, eliminane lo stack AWS CloudFormation. Per ulteriori informazioni, vedi Eliminazione di uno stack nella console AWS CloudFormation su AWS CloudFormation Guida dell'utente.

Argomenti

Esempio AWS caso d'uso delle Event Fork Pipeline

Lo scenario seguente descrive un'applicazione di E-Commerce serverless basata su eventi che utilizza AWS Event Fork Pipeline. È possibile utilizzare questo esempio di applicazione di e-commerce in AWS Serverless Application Repository e poi distribuirla Account AWS utilizzando la AWS Lambda console, dove è possibile testarla ed esaminarne il codice sorgente. GitHub



Questa applicazione di E-Commerce acquisisce gli ordini dei clienti tramite un'API RESTful ospitata da API Gateway e basata sulla funzione AWS Lambda `CheckoutApiBackendFunction`. La funzione pubblica tutti gli ordini ricevuti in un argomento Amazon SNS denominato `CheckoutEventsTopic` che, a sua volta, li distribuisce a quattro diverse pipeline.

La prima è la normale pipeline di elaborazione del checkout progettata e implementata dal proprietario dell'applicazione di E-Commerce. La pipeline include la coda `CheckoutQueue` Amazon SQS che aggiunge al buffer tutti gli ordini ricevuti, una funzione AWS Lambda denominata `CheckoutFunction` che esegue il polling della coda per l'elaborazione degli ordini e la tabella DynamoDB `CheckoutTable` che salva in modo sicuro tutti gli ordini effettuati.

Applicazione di AWS Event Fork Pipeline

La logica di business principale è gestita dai componenti dell'applicazione di E-Commerce, il cui proprietario deve comunque tenere conto anche dei seguenti fattori:

- Conformità—backup protetti e compressi crittografati a riposo e sanificazione delle informazioni sensibili
- Resilienza—riproduzione degli ordini più recenti in caso di interruzione del processo di evasione
- Searchability—esecuzione di analisi e generazione di metriche sugli ordini effettuati

Invece di implementare questa logica di elaborazione degli eventi, il proprietario dell'applicazione può iscrivere AWS Event Fork Pipelines all' `CheckoutEventsTopic` argomento Amazon SNS.

- La pipeline di archiviazione di eventi e di backup è configurata per trasformare i dati in modo da rimuovere i dettagli delle carte di credito, eseguire il buffer dei dati per 60 secondi, comprimerli usando GZIP e crittografarli tramite la chiave predefinita gestita dal cliente per Amazon S3. Tale chiave viene gestita da AWS con la tecnologia di AWS Key Management Service (AWS KMS).

  Per ulteriori informazioni, consulta Scegli Amazon S3 per la tua destinazione, Amazon Data Firehose Data Transformation e Configure Settings nella Amazon Data Firehose Developer Guide.
- La pipeline di ricerca di eventi e di analisi dei dati è configurata con un valore di 30 secondi per la durata dei nuovi tentativi di indice, un bucket per lo storage di ordini non indicizzati nel dominio di ricerca e una policy di filtro per limitare il set degli ordini indicizzati.

  Per ulteriori informazioni, consulta Scegli il OpenSearch servizio per la tua destinazione nella Amazon Data Firehose Developer Guide.
- Pipeline di riproduzione eventi è configurata con la coda Amazon SQS che fa parte della normale pipeline di elaborazione degli ordini progettata e implementata dal proprietario dell'applicazione di E-Commerce.

  Per ulteriori informazioni, consulta la sezione al Nome della coda e URL nella Guida per gli sviluppatori di Amazon Simple Queue Service.

Nella configurazione per la pipeline di ricerca di eventi e di analisi è impostata la seguente policy di filtro in formato JSON, che trova solo ordini in entrata il cui importo totale sia di 100 dollari o più. Per ulteriori informazioni, consulta Filtraggio messaggi di Amazon SNS.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

Utilizzando il modello AWS Event Fork Pipeline, il proprietario dell'applicazione di E-Commerce può evitare i costi di sviluppo spesso causati dalla codifica di una logica non differenziata per la gestione degli eventi Può invece distribuire AWS Event Fork Pipeline direttamente da AWS Serverless Application Repository nel suo Account AWS.

Fase 1: distribuzione dell'applicazione di esempio

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:
   a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
   b. Cercare `fork-example-ecommerce-checkout-api` e scegliere l'applicazione.
4. Nella pagina fork-example-ecommerce-checkout-api, procedi come segue:
   a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio `fork-example-ecommerce-my-app`).

      > **Note**
      >
      > - Per trovare con facilità le risorse in un secondo momento, mantenere il prefisso `fork-example-ecommerce`.
      > - Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se tale nome viene riutilizzato, la distribuzione si limiterà ad aggiornare lo stack AWS CloudFormation distribuito in precedenza (anziché crearne uno nuovo).

   b. (Facoltativo) Immettete una delle seguenti LogLevelimpostazioni per l'esecuzione della funzione Lambda dell'applicazione:
      - `DEBUG`
      - `ERROR`
      - `INFO` (predefinito)
      - `WARNING`
5. Scegliere I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.) e quindi, in fondo alla pagina, selezionare Deploy (Distribuisci).

Nella pagina Deployment status for fork-example-ecommerce - ***my-app***, Lambda visualizza lo stato La tua applicazione è in fase di distribuzione.

Nella sezione Resources (Risorse), AWS CloudFormation avvia la creazione dello stack e mostra lo stato CREATE_IN_PROGRESS per ciascuna risorsa. Al termine del processo, AWS CloudFormation mostra lo stato CREATE_COMPLETE.

> **Note**
>
> La distribuzione di tutte le risorse può richiedere 20-30 minuti.

Al termine della distribuzione, Lambda mostra lo stato Your application has been deployed (L'applicazione è stata distribuita).

Fase 2: esecuzione dell'applicazione di esempio

1. Nel riquadro di navigazione della console AWS Lambda, scegliere Applications (Applicazioni).
2. Nel campo di ricerca della pagina Applications (Applicazioni), cercare `serverlessrepo-fork-example-ecommerce-`*`my-app`* e quindi scegliere l'applicazione.
3. Nella sezione Resources (Risorse), procedere come segue:

   a. Per trovare la risorsa il cui tipo è ApiGatewayRestApi, ordina le risorse per Tipo, ad esempio`ServerlessRestApi`, e poi espandi la risorsa.

   b. Vengono visualizzate due risorse annidate, di tipo ApiGatewayDeployment e ApiGatewayStage.

   c. Copiare il link per Prod API endpoint (Endpoint API prod) e aggiungervi `/checkout`, ad esempio:

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. Copiare il JSON seguente in un file denominato `test_event.json`.

   ```
   {
      "id": 15311,
      "date": "2019-03-25T23:41:11-08:00",
      "status": "confirmed",
      "customer": {
         "id": 65144,
   ```

```
      "name": "John Doe",
      "email": "john.doe@example.com"
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
      "currency": "usd",
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. Per inviare una richiesta HTTPS al proprio endpoint API, passare il payload degli eventi di esempio come input eseguendo un comando `curl`, ad esempio:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

L'API restituisce la seguente risposta vuota, a indicare che l'esecuzione è riuscita:

```
{ }
```

Fase 3: verifica dell'esecuzione dell'applicazione di esempio e delle relative pipeline

Fase 1: verifica dell'esecuzione della pipeline di checkout di esempio

1. Accedi alla console Amazon DynamoDB.
2. Nel riquadro di navigazione, selezionare Tables (Tabelle).
3. Cercare `serverlessrepo-fork-example` e selezionare `CheckoutTable`.
4. Nella pagina dei dettagli della tabella, scegliere Items (Voci) e quindi scegliere la voce creata.

   Vengono mostrati gli attributi memorizzati.

Fase 2: verifica dell'esecuzione della pipeline di storage di eventi e di backup

1. Accedere alla console Amazon S3.
2. Nel riquadro di navigazione, scegliere Buckets (Bucket).
3. Cercare `serverlessrepo-fork-example` e quindi selezionare `CheckoutBucket`.
4. Esaminare la gerarchia della directory fino a trovare un file con l'estensione `.gz`.
5. Per scaricare il file, scegliere prima Actions (Azioni) e poi Open (Apri).
6. La pipeline è configurata con una funzione Lambda che sterilizza le informazioni sulle carte di credito per motivi di conformità.

   Per verificare che il payload JSON archiviato non contenga informazioni sulle carte di credito, decomprimere il file.

Fase 3: verifica dell'esecuzione della pipeline di ricerca di eventi e di analisi

1. Accedi alla console di OpenSearch servizio.
2. Nel riquadro di navigazione, in My domains (I miei domini), scegliere il dominio con prefisso `serverl-analyt`.

3. La pipeline viene configurata con una policy di filtro della sottoscrizione Amazon SNS che imposta una condizione di corrispondenza numerica.

   Per verificare che l'evento sia indicizzato perché si riferisce a un ordine il cui valore è superiore a 100 dollari, nella pagina serverl-analyt-***abcdefgh1ijk***, scegliere prima Indices (Indici) e poi checkout_events.

Fase 4: verifica dell'esecuzione della pipeline di riproduzione eventi

1. Accedere alla console Amazon SQS.
2. Nell'elenco delle code, cercare `serverlessrepo-fork-example` e scegliere `ReplayQueue`.
3. Scegli Invia e ricevi messaggi.
4. Nella finestra di dialogo Invia e ricevi messaggi in fork-example-ecommerce - ***my-app***... ReplayP- ReplayQueue - ***123ABCD4E5F6***, scegli Sondaggio per i messaggi.
5. Per verificare che l'evento sia inserito nella coda, scegliere More Details (Altri dettagli) accanto al messaggio visualizzato nella coda.

Fase 4: simulazione di un problema e riproduzione di eventi per il ripristino

Fase 1: abilitazione del problema simulato e invio di una seconda richiesta API

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, scegliere Functions (Funzioni).
3. Cercare `serverlessrepo-fork-example` e selezionare `CheckoutFunction`.
4. fork-example-ecommerceSu - *`my-app - - ABCDEF... CheckoutFunction`* pagina, nella sezione Variabili d'ambiente, imposta la variabile BUG_ENABLED su true e poi scegli Salva.
5. Copiare il JSON seguente in un file denominato `test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
```

```
    "payment": {
       "id": 3311,
       "amount": 75.00,
       "currency": "usd",
       "method": "credit",
       "card-network": "mastercard",
       "card-number": "1234 5678 9012 3456",
       "card-expiry": "12/2025",
       "card-owner": "Marcia Oliveira",
       "card-cvv": "321"
    },
    "shipping": {
       "id": 9900,
       "time": 20,
       "unit": "days",
       "method": "plane"
    },
    "items": [{
       "id": 9993,
       "product": 3120,
       "name": "Hockey Stick",
       "quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Per inviare una richiesta HTTPS al proprio endpoint API, passare il payload degli eventi di esempio come input eseguendo un comando `curl`, ad esempio:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

L'API restituisce la seguente risposta vuota, a indicare che l'esecuzione è riuscita:

```
{ }
```

Fase 2: verifica del danneggiamento simulato dei dati

1. Accedi alla console Amazon DynamoDB.
2. Nel riquadro di navigazione, selezionare Tables (Tabelle).

3. Cercare `serverlessrepo-fork-example` e selezionare `CheckoutTable`.
4. Nella pagina dei dettagli della tabella, scegliere Items (Voci) e quindi scegliere la voce creata.

   Vengono visualizzati gli attributi archiviati, alcuni contrassegnati come CORRUPTED! (Danneggiati).

Fase 3: disabilitazione del problema simulato

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, scegliere Functions (Funzioni).
3. Cercare `serverlessrepo-fork-example` e selezionare `CheckoutFunction`.
4. *Su* ***fork-example-ecommerce- my-app - - ABCDEF...*** *CheckoutFunction* pagina, nella sezione Variabili d'ambiente, imposta la variabile BUG_ENABLED su false e poi scegli Salva.

Fase 4: abilitazione della riproduzione per il ripristino dal problema

1. Nel riquadro di navigazione della console AWS Lambda, scegliere Functions (Funzioni).
2. Cercare `serverlessrepo-fork-example` e selezionare `ReplayFunction`.
3. Espandere la sezione Designer, scegliere il riquadro SQS e quindi, nella sezione SQS, scegliere Enabled (Abilitato).

   > **Note**
   >
   > L'abilitazione del trigger dell'origine evento Amazon SQS richiede circa un minuto.

4. Seleziona Salva.
5. Per visualizzare gli attributi ripristinati, tornare alla console Amazon DynamoDB.
6. Per disabilitare la riproduzione, tornare alla console AWS Lambda e disabilitare il trigger dell'origine evento Amazon SQS per `ReplayFunction`.

Sottoscrizione ad un AWS Event Fork Pipeline per un argomento Amazon SNS

Per accelerare lo sviluppo delle applicazioni basate sugli eventi, è possibile sottoscrivere pipeline di gestione degli eventi, basate su AWSPipeline Event Fork per gli argomenti Amazon SNS.AWS Event Fork Pipelines è una suite di applicazioni nidificate open source, basata su AWS Serverless Application Model (AWS SAM), che puoi implementare direttamente dalla suite AWS Event Fork Pipelines, scegliendo Show apps that create custom IAM roles or resource policies (Visualizza le app che creano ruoli IAM personalizzati o policy delle risorse), nel tuo account AWS. Per ulteriori informazioni, consulta Come AWS funziona Event Fork Pipelines.

In questa sezione viene illustrato come è possibile utilizzare le AWS Management Console per distribuire una pipeline e quindi sottoscrivere AWS Event Fork Pipeline per un argomento Amazon SNS. Prima di iniziare, crea un argomento Amazon SNS.

Per eliminare le risorse che compongono una pipeline, individua la pipeline nella pagina Applicazioni della AWS Lambda console, espandi la sezione dei modelli SAM, scegli CloudFormationstack, quindi scegli Altre azioni, Elimina stack.

Argomenti

- Distribuzione e iscrizione della pipeline di storage di eventi e di backup
- Distribuzione e iscrizione della pipeline di ricerca di eventi e di analisi
- Distribuzione e iscrizione della pipeline di riproduzione eventi

Distribuzione e iscrizione della pipeline di storage di eventi e di backup

Per l'archiviazione e l'analisi degli eventi, Amazon SNS ora consiglia di utilizzare la sua integrazi one nativa con Amazon Data Firehose. Puoi abbonare i flussi di distribuzione Firehose agli argomenti SNS, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift, Amazon Service (Service) e altro ancora. OpenSearch OpenSearch L'uso di Amazon SNS con i flussi di distribuzione Firehose è una soluzione completamente gestita e priva di codice che non richiede l'utilizzo di funzioni. AWS Lambda Per ulteriori informazioni, consulta Flussi di distribuzione da Fanout a Firehose.

Questo tutorial mostra come distribuire la pipeline di storage di eventi e di backup e iscriverla a un argomento Amazon SNS. Questo processo trasforma automaticamente il modello AWS SAM associato alla pipeline in uno stack AWS CloudFormation, che viene quindi distribuito nel tuo Account AWS. Il processo inoltre crea e configura il set di risorse che compongono la pipeline di storage di eventi e di backup, tra cui:

- Coda Amazon SQS
- Funzione Lambda
- Flussi di distribuzione Firehose
- S3 backup bucket Amazon S3

Per ulteriori informazioni sulla configurazione di uno stream con un bucket S3 come destinazione, consulta `S3DestinationConfiguration` Amazon Data Firehose API Reference.

Per ulteriori informazioni sulla trasformazione degli eventi e sulla configurazione del buffering, della compressione degli eventi e della crittografia degli eventi, consulta Creating an Amazon Data Firehose Delivery Stream nella Amazon Data Firehose Developer Guide.

Per ulteriori informazioni su come filtrare gli eventi, consulta Policy di filtro degli abbonamenti Amazon SNS in questa guida.

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:
    a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
    b. Cercare `fork-event-storage-backup-pipeline` e scegliere l'applicazione.
4. Nella pagina fork-event-storage-backup-pipeline, procedi come segue:
    a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio `my-app-backup`).

> **Note**
>
> - Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se tale nome viene riutilizzato, la distribuzione si limiterà ad aggiornare lo stack AWS CloudFormation distribuito in precedenza (anziché crearne uno nuovo).

b. (Facoltativo) Per BucketArn, inserisci l'ARN del bucket S3 in cui vengono caricati gli eventi in entrata. In assenza di valori immessi, viene creato un nuovo bucket S3 nell'account AWS.

c. (Facoltativo) Per DataTransformationFunctionArn, immettete l'ARN della funzione Lambda attraverso la quale vengono trasformati gli eventi in entrata. In assenza di valori immessi, la trasformazione dei dati è disabilitata.

d. (Facoltativo) Immettete una delle seguenti LogLevelimpostazioni per l'esecuzione della funzione Lambda dell'applicazione:

   - `DEBUG`
   - `ERROR`
   - `INFO` (predefinito)
   - `WARNING`

e. Per esempio TopicArn, inserisci l'ARN dell'argomento Amazon SNS a cui deve essere sottoscritta questa istanza della pipeline fork.

f. (Facoltativo) Per StreamBufferingIntervalInSecondse StreamBufferingSizeInMB, inserisci i valori per configurare il buffering degli eventi in arrivo. In assenza di valori immessi, vengono impostati 300 secondi e 5 MB.

g. (Facoltativo) Immettete una delle seguenti StreamCompressionFormatimpostazioni per la compressione degli eventi in arrivo:

   - `GZIP`
   - `SNAPPY`
   - `UNCOMPRESSED` (predefinito)
   - `ZIP`

h. (Facoltativo) Per StreamPrefix, inserisci il prefisso di stringa per denominare i file archiviati nel bucket di backup S3. In assenza di valori immessi, non viene utilizzato alcun prefisso.

i. (Facoltativo) Per SubscriptionFilterPolicy, inserisci la politica di filtro degli abbonamenti Amazon SNS, in formato JSON, da utilizzare per filtrare gli eventi in arrivo. La politica di filtro decide quali eventi sono indicizzati nell'indice del servizio. OpenSearch In assenza di valori immessi, non viene applicato alcun filtro (tutti gli eventi vengono indicizzati).

j. (Facoltativo) Per SubscriptionFilterPolicyScope, inserisci la stringa `MessageBody` o per `MessageAttributes` abilitare il filtraggio dei messaggi basato sul payload o sugli attributi.

k. Scegliere I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.), quindi selezionare Deploy (Distribuisci).

Nella pagina Deployment status for ***my-app (Stato distribuzione per my-app)***, Lambda mostra lo stato Your application is being deployed (La distribuzione dell'applicazione è in corso).

Nella sezione Resources (Risorse), AWS CloudFormation avvia la creazione dello stack e mostra lo stato CREATE_IN_PROGRESS per ciascuna risorsa. Al termine del processo, AWS CloudFormation mostra lo stato CREATE_COMPLETE.

Al termine della distribuzione, Lambda mostra lo stato Your application has been deployed (L'applicazione è stata distribuita).

I messaggi pubblicati nell'argomento Amazon SNS vengono archiviati nel bucket S3 di backup assegnato automaticamente dalla pipeline di storage di eventi e di backup.

Distribuzione e iscrizione della pipeline di ricerca di eventi e di analisi

Per l'archiviazione e l'analisi degli eventi, Amazon SNS ora consiglia di utilizzare la sua integrazione nativa con Amazon Data Firehose. Puoi abbonare i flussi di distribuzione Firehose agli argomenti SNS, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift, Amazon Service (Service) e altro ancora. OpenSearch OpenSearch L'uso di Amazon SNS con i flussi di distribuzione Firehose è una soluzione completamente gestita e priva di codice che non richiede l'utilizzo di funzioni. AWS Lambda Per ulteriori informazioni, consulta Flussi di distribuzione da Fanout a Firehose.

Questo tutorial mostra come distribuire la pipeline di ricerca di eventi e di analisi e iscriverla a un argomento Amazon SNS. Questo processo trasforma automaticamente il modello AWS SAM

associato alla pipeline in uno stack AWS CloudFormation, che viene quindi distribuito nel tuo Account AWS. Il processo inoltre crea e configura il set di risorse che compongono la pipeline di ricerca di eventi e di analisi, tra cui:

- Coda Amazon SQS
- Funzione Lambda
- Flussi di distribuzione Firehose
- Dominio Amazon OpenSearch Service
- Bucket di Amazon S3

Per ulteriori informazioni sulla configurazione di uno stream con un indice come destinazione, consulta `ElasticsearchDestinationConfiguration` Amazon Data Firehose API Reference.

Per ulteriori informazioni sulla trasformazione degli eventi e sulla configurazione del buffering, della compressione degli eventi e della crittografia degli eventi, consulta Creating an Amazon Data Firehose Delivery Stream nella Amazon Data Firehose Developer Guide.

Per ulteriori informazioni su come filtrare gli eventi, consulta Policy di filtro degli abbonamenti Amazon SNS in questa guida.

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:
   a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
   b. Cercare `fork-event-search-analytics-pipeline` e scegliere l'applicazione.
4. Nella pagina fork-event-search-analytics-pipeline, procedi come segue:
   a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio `my-app-search`).

> **Note**
>
> Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se tale nome viene riutilizzato, la distribuzione si limiterà ad aggiornare lo stack AWS CloudFormation distribuito in precedenza (anziché crearne uno nuovo).

b. (Facoltativo) Per DataTransformationFunctionArn, immettete l'ARN della funzione Lambda utilizzata per trasformare gli eventi in entrata. In assenza di valori immessi, la trasformazione dei dati è disabilitata.

c. (Facoltativo) Immettete una delle seguenti LogLevelimpostazioni per l'esecuzione della funzione Lambda dell'applicazione:

- `DEBUG`
- `ERROR`
- `INFO` (predefinito)
- `WARNING`

d. (Facoltativo) Per SearchDomainArn, inserisci l'ARN del dominio del OpenSearch servizio, un cluster che configura le funzionalità di elaborazione e archiviazione necessarie. In assenza di valori immessi, viene creato un nuovo dominio con la configurazione predefinita.

e. Per esempio TopicArn, inserisci l'ARN dell'argomento Amazon SNS a cui deve essere sottoscritta questa istanza della pipeline fork.

f. Per SearchIndexName, inserisci il nome dell'indice dei OpenSearch servizi per la ricerca e l'analisi degli eventi.

> **Note**
>
> I seguenti limiti si applicano ai nomi degli indici:
>
> - Non possono includere lettere maiuscole
> - Non possono includere i seguenti caratteri: `\ / * ? " < > | ` , #`
> - Non possono iniziare con i seguenti caratteri: `- + _`
> - Non possono corrispondere a: `. ..`
> - Non possono essere più lunghi di 80 caratteri
> - Non possono essere più lunghi di 255 byte

- Non può contenere due punti (da OpenSearch Service 7.0)

g. (Facoltativo) Immettete una delle seguenti SearchIndexRotationPeriodimpostazioni per il periodo di rotazione dell'indice di OpenSearch servizio:

- `NoRotation` (predefinito)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

La rotazione dell'indice aggiunge un timestamp al nome dell'indice, facilitando così la scadenza dei dati meno recenti.

h. Per SearchTypeName, inserisci il nome del tipo di OpenSearch servizio per l'organizzazione degli eventi in un indice.

> **Note**
>
> - OpenSearch I nomi dei tipi di servizio possono contenere qualsiasi carattere (eccetto i byte nulli) ma non possono iniziare con. _
> - Per OpenSearch Service 6.x, può esistere un solo tipo per indice. Se si specifica un nuovo tipo per un indice esistente che ha già un altro tipo, Firehose restituisce un errore di runtime.

i. (Facoltativo) Per StreamBufferingIntervalInSecondse StreamBufferingSizeInMB, inserite i valori per configurare il buffering degli eventi in entrata. In assenza di valori immessi, vengono impostati 300 secondi e 5 MB.

j. (Facoltativo) Immettete una delle seguenti StreamCompressionFormatimpostazioni per la compressione degli eventi in arrivo:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (predefinito)
- `ZIP`

k. (Facoltativo) Per StreamPrefix, inserisci il prefisso di stringa per assegnare un nome ai file archiviati nel bucket con lettere morte S3. In assenza di valori immessi, non viene utilizzato alcun prefisso.

l. (Facoltativo) Per StreamRetryDurationInSecons, inserire la durata dei nuovi tentativi nei casi in cui Firehose non è in grado di indicizzare gli eventi nell'indice OpenSearch dei servizi. In assenza di valori immessi, viene utilizzato un valore di 300 secondi.

m. (Facoltativo) Per SubscriptionFilterPolicy, inserisci la politica di filtro degli abbonamenti Amazon SNS, in formato JSON, da utilizzare per filtrare gli eventi in arrivo. La politica di filtro decide quali eventi sono indicizzati nell'indice del servizio. OpenSearch In assenza di valori immessi, non viene applicato alcun filtro (tutti gli eventi vengono indicizzati).

n. Scegliere I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.), quindi selezionare Deploy (Distribuisci).

Nella *my-app-search*pagina Stato di distribuzione per, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Resources (Risorse), AWS CloudFormation avvia la creazione dello stack e mostra lo stato CREATE_IN_PROGRESS per ciascuna risorsa. Al termine del processo, AWS CloudFormation mostra lo stato CREATE_COMPLETE.

Al termine della distribuzione, Lambda mostra lo stato Your application has been deployed (L'applicazione è stata distribuita).

I messaggi pubblicati sul tuo argomento Amazon SNS vengono indicizzati automaticamente nell'indice dei OpenSearch servizi fornito dalla pipeline Event Search and Analytics. Se la pipeline non riesce a indicizzare un evento, lo archivia in un bucket S3 per dead-letter.

Distribuzione e iscrizione della pipeline di riproduzione eventi

Questoa pagina mostra come distribuire la pipeline di riproduzione eventi e iscriverla a un argomento Amazon SNS. Questo processo trasforma automaticamente il modello AWS SAM associato alla pipeline in uno stack AWS CloudFormation, che viene quindi distribuito nel tuo Account AWS. Il processo inoltre crea e configura il set di risorse che compongono la pipeline di riproduzione eventi, tra cui una coda Amazon SQS e una funzione Lambda.

Per ulteriori informazioni su come filtrare gli eventi, consulta Policy di filtro degli abbonamenti Amazon SNS in questa guida.

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:

   a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).

   b. Cercare `fork-event-replay-pipeline` e scegliere l'applicazione.
4. Nella fork-event-replay-pipelinepagina, procedi come segue:

   a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio `my-app-replay`).

   > **Note**
   >
   > Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se tale nome viene riutilizzato, la distribuzione si limiterà ad aggiornare lo stack AWS CloudFormation distribuito in precedenza (anziché crearne uno nuovo).

   b. (Facoltativo) Immettete una delle seguenti LogLevelimpostazioni per l'esecuzione della funzione Lambda dell'applicazione:

      - `DEBUG`
      - `ERROR`
      - `INFO` (predefinito)
      - `WARNING`

   c. (Facoltativo) Per ReplayQueueRetentionPeriodInSeconds, inserisci il periodo di tempo, in secondi, per il quale la coda di replay di Amazon SQS conserva il messaggio. In assenza di valori immessi, viene utilizzato un valore di 1.209.600 secondi (14 giorni).

   d. Per esempio TopicArn, inserisci l'ARN dell'argomento Amazon SNS a cui deve essere sottoscritta questa istanza della pipeline fork.

   e. Per DestinationQueueName, inserisci il nome della coda Amazon SQS a cui la funzione di replay Lambda inoltra i messaggi.

   f. (Facoltativo) Per SubscriptionFilterPolicy, inserisci la politica di filtro degli abbonamenti Amazon SNS, in formato JSON, da utilizzare per filtrare gli eventi in arrivo. Tale policy

definisce quali eventi vengono aggiunti al buffer per la riproduzione. In assenza di valori immessi, non viene applicato alcun filtro (tutti gli eventi vengono aggiunti al buffer per la riproduzione).

g. Scegliere I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.), quindi selezionare Deploy (Distribuisci).

Nella *my-app-replay*pagina Stato di distribuzione per, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Resources (Risorse), AWS CloudFormation avvia la creazione dello stack e mostra lo stato CREATE_IN_PROGRESS per ciascuna risorsa. Al termine del processo, AWS CloudFormation mostra lo stato CREATE_COMPLETE.

Al termine della distribuzione, Lambda mostra lo stato Your application has been deployed (L'applicazione è stata distribuita).

I messaggi pubblicati nell'argomento Amazon SNS vengono aggiunti al buffer per la riproduzione nella coda Amazon SQS assegnata automaticamente dalla pipeline di riproduzione eventi.

> **Note**
>
> La riproduzione è disabilitata per impostazione predefinita. Per abilitarla, accedi alla pagina della funzione sulla console Lambda, espandi la sezione Designer, seleziona il riquadro SQS e quindi, nella sezione SQS, scegli Enabled (Abilitata).

Utilizzo del Pianificatore Amazon EventBridge con Amazon SNS

Il Pianificatore Amazon EventBridge è un pianficatore serverless che consente di creare, eseguire e gestire attività da un unico servizio gestito centralizzato. Con il Pianificatore EventBridge puoi creare pianificazioni utilizzando le espressioni cron e rate per schemi ricorrenti o configurare invocazioni una tantum. Puoi configurare finestre temporali flessibili per la consegna, definire limiti per nuovi tentativi e impostare il tempo massimo di conservazione per invocazioni API non riuscite.

Questa pagina descrive l'utilizzo del Pianificatore EventBridge per pubblicare un messaggio su un argomento su Amazon SNS relativo a una pianificazione.

Argomenti

Configurare il ruolo di esecuzione

Quando crei una nuova pianificazione, il Pianificatore EventBridge deve disporre dell'autorizzazione per richiamare automaticamente l'operazione dell'API di destinazione. Concedi queste autorizzazioni al Pianificatore EventBridge utilizzando un ruolo di esecuzione. La policy di autorizzazione collegata al ruolo di esecuzione della pianificazione definisce le autorizzazioni necessarie. Tali autorizzazioni dipendono dall'API di destinazione che deve essere richiamata dal Pianificatore EventBridge.

Quando utilizzi la console del Pianificatore EventBridge per creare una pianificazione, come nella procedura seguente, il Pianificatore EventBridge configura automaticamente un ruolo di esecuzione in base alla destinazione selezionata. Per creare una pianificazione utilizzando uno degli SDK del Pianificatore EventBridge, AWS CLI o AWS CloudFormation, devi disporre di un ruolo di esecuzione esistente che conceda le autorizzazioni richieste dal Pianificatore EventBridge per richiamare una destinazione. Per ulteriori informazioni sull'impostazione manuale di un ruolo di esecuzione per la pianificazione, consulta Configurazione di un ruolo di esecuzione nella Guida per l'utente del Pianificatore EventBridge.

Creare una pianificazione.

Per creare una pianificazione utilizzando la console

1. Apri la console del Pianificatore Amazon EventBridge all'indirizzo https://console.aws.amazon.com/scheduler/home.
2. Nella pagina Pianificazioni, scegli Crea pianificazione.
3. Nella pagina Specifica i dettagli della pianificazione, nella sezione Nome e descrizione della pianificazione, effettua le seguenti operazioni:

    a. Per Nome pianificazione, inserisci un nome per la pianificazione. Ad esempio, **`MyTestSchedule`**.

    b. (Facoltativo) Per Descrizione, inserisci una descrizione per la pianificazione. Ad esempio, **`My first schedule`**.

c. Per Gruppo di pianificazioni, scegli un gruppo di pianificazioni dall'elenco a discesa. Se non hai un gruppo, scegli predefinito. Per creare un gruppo di pianificazioni, scegli crea la tua pianificazione.

   I gruppi di pianificazione vengono utilizzati per aggiungere tag a gruppi di pianificazioni.

4. • Scegli le opzioni di pianificazione.

Ricorrenza	Esegui questa operazione…	
Pianificazione una tantum Una pianificazione unica richiama una destinazione solo una volta alla data e all'ora specificate.	Per Data e ora, effettua le seguenti operazioni: • Inserisci una data valida in formato `YYYY/MM/DD` . • Inserisci un timestamp in formato `hh:mm` 24 ore. • Per Fuso orario, scegli il fuso orario.	
Pianificazione ricorrente Una pianificazione ricorrente richiama una destinazione con una frequenza specificata utilizzando un'espressione cron o un'espressione rate.	a. Per Tipo di pianificazione, esegui una delle seguenti operazioni: • Per utilizzare un'espressione cron per definire la pianificazione, scegli Pianificazione basata su cron e inserisci l'espressione cron. • Per utilizzare un'espressione di frequenza per definire la pianificazione, scegli Pianificazione	

Ricorrenza	Esegui questa operazione…	
	basata su frequenza e inserisci l'espressione di frequenza. Per ulteriori informazioni sulle espressioni cron e rate, consulta Tipi di pianificazione nel Pianificatore EventBridge nella Guida per l'utente del Pianificatore Amazon EventBridge. b. Per Finestra temporale flessibile, scegli Disattivata per disattivare l'opzione o scegli una delle finestre temporali predefinite. Ad esempio, se scegli 15 minuti e imposti una pianificazione ricorrente per il richiamo della destinazione ogni ora, la pianificazione viene eseguita entro 15 minuti dall'inizio di ogni ora.	

5. (Facoltativo) Se hai scelto Pianificazione ricorrente nel passaggio precedente, nella sezione Intervallo di tempo effettua le seguenti operazioni:

   a. Per Fuso orario, scegli un fuso orario.

   b. Per Data e ora di inizio, inserisci una data valida in formato `YYYY/MM/DD`, quindi specifica un timestamp in formato `hh:mm` 24 ore.

c. Per Data e ora di fine, inserisci una data valida in formato `YYYY/MM/DD`, quindi specifica un timestamp in formato `hh:mm` 24 ore.

6. Seleziona Successivo.
7. Nella pagina Seleziona destinazione, scegli l'operazione dell'API AWS richiamata dal Pianificatore EventBridge:

   a. Scegli Pubblicazione di Amazon SNS.

   b. Nella sezione Pubblica, seleziona un argomento SNS o scegli Crea un nuovo argomento SNS.

   c. (Facoltativo) Inserisci un payload JSON. Se non inserisci un payload, il Pianificatore EventBridge utilizza un evento vuoto per richiamare la funzione.

8. Seleziona Successivo.
9. Nella pagina Settings (Impostazioni), eseguire le operazioni descritte di seguito.

   a. Per attivare la pianificazione, in Stato della pianificazione, attiva Abilita pianificazione.

   b. Per configurare una policy per nuovi tentativi per una pianificazione, in Policy per nuovi tentativi e DLQ (dead-letter queue) effettua le seguenti operazioni:

      - Attiva/disattiva Riprova.
      - Per Età massima dell'evento, inserisci il numero massimo di ore e minuti per cui il Pianificatore EventBridge deve mantenere un evento non elaborato.
      - La durata massima è 24 ore.
      - Per Numero massimo di tentativi, inserisci il numero massimo di volte che il Pianificatore EventBridge ritenta la pianificazione se la destinazione restituisce un errore.

        Il valore massimo è 185 tentativi.

      Con le policy per nuovi tentativi, se una pianificazione non riesce a richiamare la sua destinazione, il Pianificatore EventBridge esegue nuovamente la pianificazione. Se configurato, è necessario impostare il tempo di conservazione massimo e i nuovi tentativi per la pianificazione.

   c. Scegli la posizione in cui il Pianificatore EventBridge deve archiviare gli eventi non consegnati.

Opzione Dead-letter queue (DLQ)	Esegui questa operazione…
Non conservare	Scegliere None (Nessuno).
Memorizza l'evento nello stesso Account AWS in cui crei la pianificazione	a. Scegli Seleziona una coda Amazon SQS nel mio Account AWS come DLQ. b. Scegli il nome della risorsa Amazon (ARN) della coda di Amazon SQS.
Memorizza l'evento in un Account AWS diverso da quello in cui crei la pianificazione	a. Scegli Specifica una coda Amazon SQS in un altro Account AWS come DLQ. b. Inserisci il nome della risorsa Amazon (ARN) della coda di Amazon SQS.

d. Per utilizzare una chiave gestita dal cliente per crittografare l'input di destinazione, in Crittografia scegli Personalizza le impostazioni di crittografia (avanzate).

   Se scegli questa opzione, inserisci l'ARN di una chiave KMS esistente scegli Crea una AWS KMS key per accedere alla console AWS KMS. Per ulteriori informazioni sulla modalità con cui il Pianificatore EventBridge esegue la crittografia dei dati inattivi, consulta Crittografia a riposo nella Guida per l'utente del Pianificatore Amazon EventBridge.

e. Affinché il Pianificatore EventBridge crei automaticamente un nuovo ruolo di esecuzione, scegli Crea nuovo ruolo per questa pianificazione. Inserisci, quindi, un nome per Nome ruolo. Se scegli questa opzione, il Pianificatore EventBridge collega al ruolo le autorizzazioni necessarie per la destinazione basata sul modello.

10. Seleziona Successivo.

11. Nella pagina Rivedi e crea pianificazione, rivedi i dettagli della pianificazione. In ogni sezione, scegli Modifica per tornare a tale passaggio e modificarne i dettagli.
12. Scegli Crea pianificazione.

    Puoi visualizzare un elenco delle pianificazioni nuove ed esistenti nella pagina Pianificazioni. Nella colonna Stato, accertati che la nuova pianificazione sia Abilitata.

Risorse correlate

Per ulteriori informazioni sul Pianificatore EventBridge, consulta:

- Guida per l'utente del Pianificatore EventBridge
- Riferimento all'API del Pianificatore EventBridge
- Prezzi del Pianificatore EventBridge

Using Amazon SNS per la messaggistica da applicazione a persona (A2P)

Questa sezione fornisce informazioni sull'utilizzo di Amazon SNS per notifiche all'utente con sottoscrittori quali ad esempio applicazioni per dispositivi mobili, numeri di cellulare e indirizzi e-mail.

Argomenti

Messaggi di testo per dispositivi mobili (SMS)

Puoi utilizzare Amazon SNS; per inviare messaggi SMS a dispositivi abilitati. Puoi inviare un messaggio direttamente a un numero di telefono oppure inviarlo a più numeri contemporaneamente sottoscrivendo quei numeri a un argomento e inviando il messaggio all'argomento.

Puoi impostare preferenze SMS per il tuo account AWS, in modo da personalizzare le consegne SMS in base ai casi d'uso e al budget. Ad esempio, puoi scegliere se ottimizzare i costi o l'affidabilità della consegna dei messaggi, Nonché specificare i limiti di spesa per le consegne di messaggi individuali e i limiti di spesa mensili per il tuo Account AWS.

Dove richiesto da leggi e regolamenti locali (ad esempio negli Stati Uniti e in Canada), i destinatari SMS possono chiedere di essere esclusi dalla ricezione dei messaggi SMS provenienti dal tuo Account AWS. Dopo che un destinatario ha scelto l'esclusione, puoi, entro certo limiti, inserire nuovamente il numero di telefono nell'elenco di invio per riprendere la distribuzione dei messaggi al numero specificato.

Amazon SNS supporta la messaggistica SMS in diverse regioni e puoi inviare messaggi a oltre 200 paesi e regioni. Per ulteriori informazioni, consulta Regioni e paesi supportati.

Argomenti

- Impostazione delle preferenze di messaggistica SMS
- Invio di messaggi SMS
- Monitoraggio dell'attività SMS
- Gestione dei numeri di telefono e delle sottoscrizioni SMS
- Regioni e paesi supportati
- Best practice per il canale SMS

Sandbox SMS

Quando inizi a utilizzare Amazon SNS per inviare messaggi SMS, l'account AWS si trova nella sandbox SMS. La sandbox SMS offre un ambiente sicuro per provare le funzionalità Amazon SNS senza rischiare la reputazione del mittente SMS. Mentre il tuo account si trova nella sandbox SMS, puoi utilizzare tutte le funzionalità di Amazon SNS, con le seguenti restrizioni:

- È possibile inviare messaggi SMS solo a numeri di telefono di destinazione verificati.
- Puoi avere un massimo di 10 numeri di telefono di destinazione verificati.
- Puoi eliminare i numeri di telefono di destinazione solo dopo almeno 24 ore dalla verifica o dall'ultimo tentativo di verifica.

Quando il tuo account viene rimosso dalla sandbox, queste restrizioni vengono rimosse ed è possibile inviare messaggi SMS a qualsiasi destinatario.

Argomenti

- Aggiunta e verifica dei numeri di telefono nella sandbox SMS
- Eliminazione dei numeri di telefono dalla sandbox SMS
- Uscita dalla sandbox SMS

Aggiunta e verifica dei numeri di telefono nella sandbox SMS

Per iniziare a inviare messaggi SMS mentre il tuo AWS account è nella sandbox SMS, crea un'identità di origine, aggiungi i numeri di telefono di destinazione e quindi verificali.

> **Note**
>
> Come per gli account che non si trovano nella sandbox SMS, un'identità di origine è necessaria prima di poter inviare messaggi SMS ai destinatari in alcuni paesi o regioni. Per ulteriori informazioni, consulta Regioni e paesi supportati.
> Gli ID di origine includono ID mittente e diversi tipi di numeri di origine. Per visualizzare i numeri di origine esistenti, nel riquadro di navigazione della console Amazon SNS, scegliere numeri di origine. Al momento, gli ID mittente non vengono visualizzati in questo elenco.

Per aggiungere e verificare i numeri di telefono di destinazione

1. Accedi alla console Amazon SNS.
2. Crea un'identità di origine per il numero di telefono.
3. Nel menu della console, scegli una regione AWS che supporti la messaggistica SMS.
4. Nel riquadro di navigazione seleziona Text messaging (SMS) (SMS).
5. Alla pagina Messaggi di testo per dispositivi mobili (SMS), sotto Numeri di telefono di destinazione sandbox, scegliere Aggiungi numero di telefono.
6. Su Dettagli di destinazione, inserisci il codice paese e il numero di telefono, specifica la lingua da utilizzare per il messaggio di verifica, quindi seleziona Aggiungi numero di telefono.

   Amazon SNS invia una password una tantum (OTP) al numero di telefono di destinazione. Se il numero di telefono di destinazione non riceve l'OTP entro 15 minuti, scegliere Invia nuovamente il codice di verifica. Puoi inviare l'OTP allo stesso numero di telefono di destinazione fino a cinque volte ogni 24 ore.
7. Nella casella Codice di verifica immettere l'OTP inviato al numero di telefono di destinazione, quindi scegliere Verifica del numero di telefono.

   Il numero di telefono di destinazione e il relativo stato di verifica vengono visualizzati nella sezione Numeri di telefono di destinazione sandbox. Se lo stato di verifica è Pending (In sospeso), la verifica non è riuscita. Ciò può accadere, ad esempio, se non hai inserito il prefisso del paese con il numero di telefono. Puoi eliminare i numeri di telefono di destinazione in sospeso o verificati solo dopo almeno 24 ore dalla verifica o dall'ultimo tentativo di verifica.
8. Ripeti questi passaggi in ogni regione in cui desideri utilizzare questo numero di telefono di destinazione.

Risoluzione dei problemi relativi alla mancata ricezione di un testo OTP

Risolvi i problemi più comuni che possono impedire a un numero di telefono di ricevere messaggi OTP.

- Limite di spesa SMS di Amazon SNS: se hai Account AWS superato il limite di spesa per l'invio di messaggi SMS, è possibile che altri messaggi, inclusi gli SMS OTP, non vengano recapitati finché il limite non viene aumentato o il problema di fatturazione non viene risolto.
- Numero di telefono non utilizzato per le notifiche SMS: in alcuni paesi o regioni, i destinatari devono attivare la ricezione di messaggi SMS da codici brevi, che vengono comunemente utilizzati per i messaggi OTP. Se il numero di telefono del destinatario non è abilitato, non riceverà il testo OTP.
- Restrizioni o filtri dell'operatore: alcuni operatori di telefonia mobile potrebbero disporre di restrizioni o meccanismi di filtraggio che impediscono la consegna di determinati tipi di messaggi SMS, inclusi i messaggi OTP. Ciò potrebbe essere dovuto alle politiche di sicurezza o alle misure antispam implementate dal gestore.
- Numero di telefono non valido o errato: se il numero di telefono fornito dal destinatario è errato o non valido, il testo OTP non verrà recapitato.
- Problemi di rete: problemi o interruzioni temporanee della rete potrebbero impedire la consegna di messaggi SMS, inclusi SMS OTP, al telefono del destinatario.
- Consegna ritardata: in alcuni casi, i messaggi SMS possono subire ritardi nella consegna a causa della congestione della rete o di altri fattori. Il testo OTP potrebbe alla fine essere recapitato, ma potrebbe subire ritardi oltre il periodo di tempo previsto.
- Sospensione o chiusura dell'account: in caso di problemi con il tuo account Account AWS, come il mancato pagamento o la violazione dei AWS termini di servizio, le funzionalità di messaggistica di Amazon SNS, inclusi gli SMS OTP, potrebbero essere sospese o interrotte.

Eliminazione dei numeri di telefono dalla sandbox SMS

È possibile eliminare i numeri di telefono di destinazione in sospeso o verificati dalla sandbox SMS.

Per eliminare i numeri di telefono di destinazione dalla sandbox SMS

1. Attendere 24 ore dopo la verifica del numero di telefono o 24 ore dopo l'ultimo tentativo di verifica.
2. Accedi alla console Amazon SNS.

3. Nel menu della console, scegli un'area AWS che supporti la messaggistica SMSdove è stato aggiunto un numero di telefono di destinazione.
4. Nel riquadro di navigazione, seleziona Text messaging (SMS) (SMS).
5. Alla pagina Messaggi di testo per dispositivi mobili (SMS), su Numeri di telefono di destinazione sandbox, scegliere il numero di telefono da eliminare, quindi scegliere Elimina il numero di telefono.
6. Per confermare che si desidera eliminare il numero di telefono, immettere **`delete me`** e quindi scegliere Delete (Elimina).

   Se sono trascorse almeno 24 ore da quando hai verificato o hai tentato di verificare il numero di telefono di destinazione, questo viene eliminato e Amazon SNS aggiorna l'elenco dei tuoi numeri di telefono di destinazione.
7. Ripetere questi passaggi in ogni regione in cui è stato aggiunto il numero di telefono di destinazione e non si prevede più di utilizzarlo.

Uscita dalla sandbox SMS

Account AWS Per uscire dalla sandbox SMS è necessario innanzitutto aggiungere, verificare e testare i numeri di telefono di destinazione. Quindi, devi creare un caso con AWS Support.

Per richiedere che il tuo AWS account venga rimosso dalla sandbox SMS

1. Verifica i numeri di telefono
   a. Mentre ti Account AWS trovi nella sandbox SMS, apri la console Amazon SNS.
   b. Nel pannello di navigazione, sotto Mobile, scegli Messaggi di testo (SMS).
   c. Nella sezione Sandbox dedicata ai numeri di telefono di destinazione, aggiungi e verifica uno o più numeri di telefono di destinazione. Questa verifica garantisce che tu possa inviare e ricevere messaggi con successo.
2. Prova la pubblicazione di SMS
   - Conferma di essere in grado di inviare e ricevere messaggi ad almeno un numero di telefono verificato. Per istruzioni più dettagliate su come pubblicare messaggi SMS, consultaPubblicazione su un telefono cellulare.
3. Avvia la modifica della sandbox

- Nella pagina Messaggi di testo per dispositivi mobili (SMS) della console Amazon SNS, sotto Informazioni sull'account, scegliere Exit SMS sandbox (Uscire dalla sandbox SMS). Questa azione ti reindirizza all'Amazon Support Center e crea automaticamente un caso di supporto con l'opzione di aumento della quota di servizio selezionata.

4. Compila il modulo

   - Nel modulo di assistenza sotto Aumento della quota di servizio, procedi come segue:

     i. Scegli «SNS Text Messaging» come servizio.

     ii. Fornisci l'URL del sito Web o il nome dell'app da cui intendi inviare messaggi SMS.

     iii. Specificate il tipo di messaggi che invierete: password monouso, promozionale o transazionale.

     iv. Scegli il tipo Regione AWSda cui inviare i messaggi SMS.

     v. Elenca i paesi o le regioni in cui intendi inviare messaggi SMS.

     vi. Descrivi in che modo i tuoi clienti scelgono di ricevere messaggi.

     vii. Includi tutti i modelli di messaggio che intendi utilizzare.

5. Specificare quota e regione

   - In Requests (Richieste), procedere come segue:

     i. Scegli Regione AWSdove vuoi spostare il tuo Account AWS.

     ii. Scegli i limiti generali per il tipo di risorsa.

     iii. Scegli Exit SMS Sandbox per Quota.

     iv. (Facoltativo) Per richiedere ulteriori aumenti o altri aggiustamenti, scegli Aggiungi un'altra richiesta e specifica i dettagli necessari.

     v. In Nuovo valore di quota, inserisci il limite in USD che stai richiedendo.

6. Dettagli aggiuntivi

   a. Nella descrizione del caso, fornisci eventuali dettagli aggiuntivi pertinenti alla tua richiesta.

   b. In Opzioni di contatto, scegli la lingua di contatto preferita.

7. Invia la richiesta

   - Scegli Invia a cui inviare la richiesta AWS Support.

Il AWS Support team fornisce una risposta iniziale alla tua richiesta entro 24 ore.

Per evitare che i nostri sistemi vengano utilizzati per l'invio di contenuti indesiderati o dannosi, ogni richiesta verrà analizzata attentamente da parte nostra. In seguito a questa valutazione, saremo in grado di gestire la tua richiesta durante le prime 24 ore. Tuttavia, se la risoluzione richiede ulteriori informazioni da parte tua, i tempi di gestione della richiesta potranno essere più lunghi.

Se il caso d'uso specifico non è conforme con le nostre policy, potremmo non essere in grado di gestire la tua richiesta s

Identità di origine per i messaggi SMS

Quando invii messaggi SMS tramite Amazon SNS, puoi identificarti ai destinatari utilizzando i seguenti tipi di identità di origine:

- ID mittente
- Numeri di origine

> **Note**
>
> I messaggi SMS di Amazon SNS sono disponibili nelle regioni in cui Amazon Pinpoint non è attualmente supportato. Se operi in Europa (Stoccolma), Medio Oriente (Bahrein), Europa (Parigi), Sud America (San Paolo) o US West (California), apri la console Amazon Pinpoint nella regione Stati Uniti orientali (Virginia) per registrare la tua azienda e la tua campagna 10DLC, ma non richiedere un numero 10DLC. Utilizzare invece la Console Service Quotas di AWS per creare un caso di aumento del limite di servizio durante la richiesta del numero 10DLC per quella regione specifica. Per ulteriori informazioni su come richiedere identità di origine, consulta Richiesta di supporto per la messaggistica SMS con Amazon SNS.

ID mittente

Un ID mittente è un nome in caratteri alfabetici che identifica il mittente di un messaggio SMS. Quando si invia un messaggio SMS utilizzando un ID mittente e il destinatario si trova in un'area in cui è supportata l'autenticazione dell'ID mittente, l'ID mittente viene visualizzato sul dispositivo del destinatario anziché su un numero di telefono. Un ID mittente offre ai destinatari di SMS maggiori informazioni sul mittente rispetto a un numero di telefono o un codice lungo o breve.

Gli ID mittente sono supportati in diversi paesi e regioni di tutto il mondo. In alcuni, un'azienda che invia messaggi SMS a singoli clienti deve utilizzare un ID mittente preregistrato presso un ente normativo o un gruppo di settore. Per un elenco completo dei paesi e delle regioni che supportano o richiedono ID mittente, consulta Regioni e paesi supportati.

Non sono previsti costi aggiuntivi per l'utilizzo di un ID mittente. Tuttavia, il supporto e i requisiti per l'autenticazione dell'ID mittente variano. Alcuni mercati importanti (tra cui Canada, Cina e Stati Uniti) non supportano l'ID mittente. In alcuni, le aziende che inviano messaggi SMS a singoli clienti devono utilizzare un ID mittente preregistrato presso un ente normativo o un gruppo di settore.

> **Important**
>
> AWS proibisce lo spoofing degli SMS, in cui l'ID mittente viene utilizzato per rappresentare un'altra persona, azienda o prodotto. Utilizzare solo un ID mittente che rappresenta un marchio o un marchio di proprietà dell'utente.

Vantaggi

Gli ID mittente offrono al destinatario maggiori informazioni sul mittente del messaggio. Un ID mittente consente di stabilire l'identità del marchio con maggiore facilità rispetto a un codice breve o lungo. Non sono previsti costi aggiuntivi per l'utilizzo di un ID mittente.

Svantaggi

Il supporto e i requisiti per l'autenticazione di ID mittente non sono uniformi in tutti i paesi o le regioni. Alcuni mercati importanti (tra cui Canada, Cina e Stati Uniti) non supportano l'ID mittente. In alcune aree, per poter essere utilizzati gli ID mittente devono prima essere pre-approvati da un ente normativo.

Registrazione dell'ID mittente per paese

È necessario aprire una richiesta di supporto AWS per registrare gli ID dei mittenti per la messaggistica SMS. Dopo l'invio della richiesta di supporto, AWS condividerà i documenti aggiuntivi richiesti. È inoltre necessario fornire le seguenti informazioni per il Paese appropriato in cui si sta registrando l'ID del mittente.

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Australia (AU)	Transazionale e promozionale	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale • L'ID mittente deve essere il nome del marchio dell'azienda che invia l'SMS	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Numero di licenza commerciale ufficiale della società o numero di partita IVA

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione

			• Modelli di messaggi che intendi inviare
			• Una licenza di registrazione aziendale. Gli esempi includono (a titolo esemplifi cativo):
			  • Australian Business Number (ABN)
			  • Australian Company Number (ACN)
			  • Australian Registered Body Number (ARBN)
			  • Indigenous Corporation Number (ICN)
			• Letter of Authoriza tion (LOA)

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Bielorussia (BY)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale • L'ID mittente deve essere il nome del marchio dell'azienda che invia l'SMS	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Numero di licenza commerciale

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			ufficiale della società o numero di partita IVA • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Cina (CN) Invece della registrazione dell'ID mittente, la Cina richiede la registrazione del contenuto/modello del messaggio con la firma anteposta al corpo del messaggio (ad esempio, [Amazon]).	Le seguenti parole chiave non sono consentite: • Falung Gong • SB • Piazza Tienanmen Messaggi delle seguenti categorie: • Carte di credito • Pagamenti digitali (incluse le criptovalute) • URL di traffico fraudolento (phishing o spam) • Gioco d'azzardo • Contenuti inappropriati (per adulti, violenza, droga, alcol) • Prestiti • Chirurgia plastica • Politica • Religione • Trading azionario • Valute virtuali	• Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• Company name (Nome dell'azienda) • Indirizzo dell'azienda • Stato o provincia • Paese • Numero di telefono dell'azienda • Sito Web dell'azienda • Volume mensile stimato • Tipo di messaggio: promozionale/transazionale • Spiegazione del caso d'uso • Modelli che desideri registrare (i messaggi SMS inviati non devono discostarsi dai modelli che fornisci per garantire la conformità e il corretto recapito dei messaggi). Ad esempio, [CompanyName] La tua password monouso è {OTP}. Questo codice

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
	Le firme tra parentesi quadre devono essere anteposte al corpo del messaggio SMS. Affinché i messaggi vengano recapitati correttamente in Cina, è necessario registrare una firma con il modello del contenuto del messaggio. Tale firma deve essere aggiunta al contenuto di ogni messaggio SMS inviato. Se la firma registrata non viene anteposta al corpo dei messaggi SMS, questi ultimi potrebbero venire bloccati o filtrati. Le firme devono essere anteposte al corpo dell'SMS e incluse tra parentesi quadre.		scadrà tra 10 minuti. • Conferma che non verranno inviati contenuti promozionali come messaggi transazionali • Firma del messaggio. Per informazioni, consultare Restrizioni e requisiti relativi al formato della firma.

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Egitto (EG)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • La tua azienda ha un'entità/un ufficio

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			commerciale in Egitto? Oppure si tratta di un'azienda che non ha sede in Egitto, ma che invia messaggi in Egitto? • Conferma scritta che il caso d'uso riguarda tutti i messaggi che devono essere inviati con questo ID mittente • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Modelli di messaggi che intendi inviare
India (IN)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 6 caratteri • Nessuno spazio • Nessun carattere speciale	Per informazioni, consultare Requisiti per la registrazione dell'ID mittente per l'India.

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Giordania (JO)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			• Certificato di registrazione aziendale • Il tipo di messaggio che desideri inviare (ad esempio, OTP, avvisi) • Conferma scritta che il caso d'uso descrive tutti i messaggi che devono essere inviati con questo ID mittente • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Kuwait (KW)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Il tipo di messaggio che desideri inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			(ad esempio, OTP, avvisi) • Conferma scritta che il caso d'uso descrive tutti i messaggi che devono essere inviati con questo ID mittente • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Filippine (PH)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Il tipo di messaggio che desideri inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			(ad esempio, OTP, avvisi) • Conferma scritta che il caso d'uso descrive tutti i messaggi che devono essere inviati con questo ID mittente • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Qatar (QA)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Tipo di SMS da inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			• (transazionale/promozionale/OTP) Tieni presente che, per essere conformi, solo i messaggi transazionali/OTP possono essere utilizzati con gli ID mittente destinati al Qatar. • Conferma scritta che il caso d'uso descrive tutti i messaggi che devono essere inviati con questo ID mittente • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Russia (RU)	Solo messaggi transazionali	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Codice fiscale o numero di licenza • Certificato di registrazione aziendale • Indirizzo e-mail del contatto • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			• Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Conferma che questo caso d'uso si applicherà a tutti i messaggi inviati in Russia da questo account • Accettazione del fatto che i messaggi non transazionali devono essere inviati utilizzando un ID mittente separato • Canone di 272 USD/mese; conferma dell'approvazione di questo canone mensile ricorrente: Sì/No • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Arabia Saudita (SA)	Ogni ID mittente deve essere transazionale o promozionale. Un ID mittente non può essere utilizzato per entrambi i tipi di traffico. Se si invia traffico relativo a OTP o 2FA, l'ID mittente deve essere utilizzato solo a tale scopo. Gli ID mittenti promozionali saranno soggetti all'elenco Non disturbare (DND) dell'Arabia Saudita.	• Lunghezza dell'ID mittente promozionale: 2-8 caratteri, ID mittente preceduto da "-AD" • Lunghezza dell'ID mittente transazionale: 2-11 caratteri • L'ID mittente deve rappresentare l'identità del marchio del mittente • Includi almeno una lettera • Non utilizzare caratteri speciali ASCII (ad esempio, #, @) • Puoi includere lettere maiuscole e minuscole e numeri da 0 a 9	La registrazione dell'ID mittente in Arabia Saudita è supportata solo per le società internazionali, mentre le società nazionali dell'Arabia Saudita non sono al momento supportate • L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			• Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Modelli di messaggi che desideri inviare. • Questo ID mittente verrà utilizzato per contenuti transazionali o promozionali? • Accettazione del fatto che i messaggi non transazionali devono essere inviati utilizzando un ID mittente separato • In caso di invio di traffico relativo a 2FA o OTP, conferma che questo ID mittente verrà utilizzato SOLO per tale scopo

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Singapore (SG)	Solo messaggi transazionali	• Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	Per informazioni, consultare Requisiti per la registrazione dell'ID mittente per Singapore.

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Sri Lanka (LK)	Nessuna restrizione o nessun requisito speciale	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Tipo di azienda (nazionale/internazionale) • URL dell'azienda (link all'app o al sito Web dell'azienda) • Spiegazione del caso d'uso e scopo dei messaggi • Modelli di messaggi che intendi inviare • Questo ID mittente verrà utilizzato per contenuti transazionali o promozionali? • Accettazione del fatto che i messaggi non transazionali devono essere inviati utilizzando un ID mittente separato • In caso di invio di traffico relativo

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			a 2FA o OTP, conferma che questo ID mittente verrà utilizzato solo per tale scopo

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Thailandia (TH)	Nessuna restrizione o nessun requisito speciale	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Tipo di SMS da inviare (transazi

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			onale/promozionale /OTP) • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Turchia (TR)	Nessuna restrizione o nessun requisito speciale	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • URL dell'azienda (link all'app o al sito Web dell'azienda) • Se la tua azienda è nazionale (sede in Turchia) o internazionale (sede fuori dalla Turchia) • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Tipo di SMS da inviare (transazi

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			onale/promozionale /OTP) • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Ucraina (UA)	Nessuna restrizione o nessun requisito speciale	• Carattere alfanumerico • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • URL dell'azienda (link all'app o al sito Web dell'azienda) • Numero di partita IVA • Azienda nazionale o internazionale • Volume mensile stimato • Spiegazione del caso d'uso e scopo dei messaggi • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Tipo di SMS da inviare (transazi

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			onale/promozionale /OTP) • Modelli di messaggi che intendi inviare

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
Emirati Arabi Uniti (AE)	Solo messaggi transazionali	• Carattere alfanumerico • L'ID mittente deve identificare il marchio; gli ID mittenti generici non sono consentiti • Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Company name (Nome dell'azienda) • Indirizzo dell'azienda (inclusi città, stato/provincia, codice postale) • Paese dell'azienda • Il numero di telefono di contatto dell'azienda • URL dell'azienda (link all'app o al sito Web dell'azienda) • Volume mensile stimato • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Spiegazione del caso d'uso e scopo dei messaggi • Numero di licenza commerciale

Nome paese	Tipo di messaggio	Restrizioni e requisiti relativi al formato	Requisiti della registrazione
			ufficiale della società o numero di partita IVA • Conferma scritta che tutto il traffico inviato da questo ID mittente rientra nel caso d'uso fornito • Modelli di messaggi che intendi inviare
Vietnam (VN)	Solo messaggi transazionali. I messaggi promozionali e di marketing non sono consentiti. I contenuti proibiti includono: • Contenuti per adulti • Servizi di beneficenza • Criptovalute • Lotteria • Gioco d'azzardo su dispositivi mobili/casinò • Scommesse sportive • Votazioni	• Massimo 11 caratteri • Nessuno spazio • Nessun carattere speciale	• L'ID mittente che desideri registrare • Da quale regione AWS l'utente chiamerà l'API o il servizio • Volume mensile stimato • Se non è ovvia, spiega la connessione tra il nome dell'azienda e l'ID mittente • Spiegazione del caso d'uso e scopo dei messaggi • Conferma scritta che tutto il traffico inviato da questo ID mittente rientra nel caso d'uso fornito

Restrizioni e requisiti relativi al formato della firma

Affinché i messaggi vengano recapitati correttamente in Cina, è necessario registrare una firma con il modello del contenuto del messaggio. Tale firma deve essere aggiunta al contenuto di ogni messaggio SMS inviato. Se la firma registrata non viene anteposta al corpo dei messaggi SMS, questi ultimi potrebbero venire bloccati o filtrati.

- La firma deve essere anteposta al corpo del messaggio SMS e inclusa tra parentesi quadre
- Il testo standard deve essere incluso tra parentesi quadre
- Per il testo Unicode, la firma deve essere inclusa tra parentesi lenticolari: U+3010 PARENTESI LENTICOLARE SINISTRA e U+3011 PARENTESI LENTICOLARE DESTRA – Esempio: [Avviso]
- Deve essere di lunghezza compresa tra 3 e 11 caratteri
- Sono supportati caratteri cinesi e inglesi

Requisiti relativi all'ID mittente per la Francia

Questa guida fornisce le fasi e le linee guida necessarie per creare un ID mittente dedicato richiesto dagli operatori di telefonia mobile francesi per inviare SMS in Francia.

Argomenti

- Configurazione di un ID mittente dedicato per la Francia
- Linee guida per la denominazione dell'ID mittente

Configurazione di un ID mittente dedicato per la Francia

Per specificare un ID mittente dedicato, è possibile usare uno dei seguenti metodi. Amazon SNS utilizzerà l'ID mittente per conto dell'utente per i messaggi SMS pubblicati tramite l'API `Publish`.

- È possibile utilizzare la console Amazon SNS per configurare l'ID mittente predefinito da utilizzare per tutti gli SMS pubblicati. Per ulteriori informazioni, visitare Impostazione delle preferenze per i messaggi SMS utilizzando AWS Management Console.
- È possibile utilizzare l'API `Publish` per impostare l'ID mittente utilizzando l'attributo dei messaggio `AWS.SNS.SMS.SenderID` quando si richiede ad Amazon SNS di pubblicare un messaggio SMS. Per ulteriori informazioni, visitare Invio di un messaggio (Console).

Linee guida per la denominazione dell'ID mittente

- Il nome dell'ID mittente deve essere alfanumerico con un massimo di 11 caratteri.
- Il nome dell'ID mittente non deve contenere caratteri speciali o spazi.
- Consigliamo di utilizzare lo stesso nome per l'ID mittente e il marchio dell'azienda che invia il messaggio di testo SMS.

Requisiti per la registrazione dell'ID mittente per l'India

Per impostazione predefinita, quando si inviano messaggi a destinatari in India, Amazon SNS utilizza connessioni ILDO (International Long Distance Operator) per trasmettere tali messaggi. Quando i destinatari vedono un messaggio inviato tramite una connessione ILDO, sembra essere inviato da un ID numerico casuale (a meno che non sia stato acquistato un codice breve dedicato).

> **Note**
>
> Il prezzo per l'invio di messaggi tramite route locali è indicato nella pagina Prezzi globali SMS di Amazon SNS. Il prezzo per l'invio di messaggi tramite connessioni ILDO è superiore al prezzo per l'invio di messaggi tramite route locali.

Se si preferisce utilizzare un ID mittente alfabetico per i messaggi SMS, è necessario inviare tali messaggi tramite route locali piuttosto che su route ILDO. Per inviare messaggi utilizzando route locali, devi prima registrare il tuo caso d'uso e i modelli di messaggio presso la Telecom Regulatory Authority of India (TRAI) tramite i portali Distributed Ledger Technology (DLT). Questi requisiti di registrazione sono concepiti per ridurre il numero di messaggi non richiesti ricevuti dai consumatori indiani e per proteggere i consumatori da messaggi potenzialmente dannosi. Questo processo di registrazione è gestito da Vodafone India attraverso il suo servizio Vilpower.

Argomenti

- Fase 1: registrazione con TRAI
- Passaggio 2: Richiesta di un ID mittente
- Fase 3: invio di messaggi SMS
- Risoluzione dei problemi dei messaggi SMS inviati ai destinatari in India

Fase 1: registrazione con TRAI

Prima di poter inviare messaggi SMS a destinatari in India, è necessario registrare la propria organizzazione presso il TRAI (Telecom Regulatory Authority of India). Sii pronto a fornire le informazioni riportate di seguito durante il processo di registrazione:

- PAN (Permanent Account Number) dell'organizzazione.
- TAN (Tax Deduction Account Number) dell'organizzazione.
- GSTIN (Goods and Services Tax Identification Number) dell'organizzazione.
- CIN (Corporate Identity Number) dell'organizzazione.
- Una lettera di autorizzazione che ti dà l'autorità per registrare la tua organizzazione.

Di seguito è riportato un elenco di esempio di alcuni siti di registrazione Distributed Ledger Technology (DLT) che puoi utilizzare per registrare la tua organizzazione con il TRAI (potrebbero essere applicate commissioni). Il processo di registrazione varia in base al sito. Contatta i rispettivi team di supporto per assistenza.

- BSNL DLT: registrazione gratuita.
- Jio Trueconnect: addebita una commissione per il completamento del processo di registrazione.
- Smart Enterprise Solutions: addebita una commissione per il completamento del processo di registrazione.
- Vilpower: include un modello che puoi scaricare e modificare in base alle tue esigenze. Vilpower addebita una tassa per completare il processo di registrazione.

Per registrare l'organizzazione con TRAI

Di seguito viene illustrato come registrare l'organizzazione con TRAI utilizzando Vilpower.

1. In un browser Web, andare sul sito web di Vilpower all'indirizzo https://www.vilpower.in.
2. Scegliere Signup (Iscriviti) per creare un altro account. Durante il processo di registrazione, effettuare le seguenti operazioni:
   - Per il tipo di entità con cui registrarsi, scegliere Come azienda.
   - Per Nome del telemarketer, utilizzare Infobip Private Limited - TUTTI. Quando richiesto, inizia a digitare **`Infobip`** e quindi scegliere Infobip Private Limited — TUTTI dall'elenco a discesa.
   - In Inserire ID Telemarketer, inserire **`1102000011152`**.

- Quando viene richiesto di specificare gli ID intestazione, immettere gli ID mittente che si desidera registrare.

> **Note**
>
> L'India richiede che gli ID mittente siano lunghi esattamente sei caratteri.

- Quando viene richiesto di fornire i modelli di contenuto, immettere il contenuto del messaggio che si intende inviare ai destinatari. Includere un modello per ogni messaggio che si intende inviare.

> **Note**
>
> I siti Web del provider di registrazione DLT non sono gestiti da Amazon Web Services. I passaggi sui loro siti Web sono soggetti a modifica.

Passaggio 2: Richiesta di un ID mittente

Per richiedere un ID mittente in India, devi presentare un AWS Support. Completa le fasi descritte in Richiesta di ID mittente. Nella tua richiesta, fornisci le informazioni che seguono:

- L'area geografica AWS da cui il mittente intende inviare messaggi SMS.
- Il nome della società utilizzato durante il processo di registrazione DLT.
- L'ID entità principale (PEID) ricevuto dopo la registrazione dell'entità DLT completata.
- Volumi mensili stimati.
- Una spiegazione del caso d'uso.
- Una descrizione del flusso immesso dall'utente.
- Conferma che gli opt-ins dell'utente finale siano raccolti e registrati.

Fase 3: invio di messaggi SMS

Dopo la registrazione dell'organizzazione con TRAI è possibile inviare messaggi SMS ai destinatari in India.

1. Accedi alla console Amazon SNS.

2. Nel menu della console, imposta lo strumento di selezione della regione su una regione che supporti la messaggistica SMS.
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Nella pagina Text messaging (SMS) (Messaggi di testo (SMS)), scegli Send a text message (SMS) (Invia un messaggio di testo (SMS)). La finestra Pubblicazione di messaggi SMS si aprirà.
5. Per Message type (Tipo di messaggio), scegli una delle seguenti opzioni:

   - Promotional (Promozionali) - Messaggi non critici, come i messaggi di marketing.

     Quando utilizzi ID mittente numerici, scegli questa opzione.
   - Transactional (Transazionali) - Messaggi critici che supportano le transazioni dei clienti, come le password monouso per l'autenticazione a più fattori.

     Quando utilizzi ID mittente alfabetici o alfanumerici, scegli questa opzione.

   Questa impostazione a livello di messaggio sostituisce il tipo di messaggio predefinito, che è stato impostato sulla pagina Text messaging preferences (Preferenze messaggi di testo).

   Per informazioni sulle tariffe relative a messaggi promozionali e transazionali, consulta la pagina relativa alle tariffe SMS globali.
6. In Numero, digita il numero di telefono al quale vuoi inviare il messaggio.
7. In Message (Messaggio), inserisci il testo da inviare.

   Quando aggiungi contenuto ai messaggi SMS, assicurati che corrisponda esattamente al contenuto nel modello registrato DLT. I corrieri bloccano i messaggi SMS se il contenuto del messaggio include ritorni di caratteri aggiuntivi, spazi, punteggiatura o maiuscole e minuscole di frase non corrispondenti. Le variabili in un modello possono contenere fino a 30 caratteri.
8. Nella sezione Identità di origine, per ID mittente immetti un ID personalizzato che contiene da 3 a 11 caratteri.

   Gli ID mittente possono essere numerici per i messaggi promozionali oppure alfabetici o alfanumerici per i messaggi transazionali. L'ID mittente viene visualizzato come mittente del messaggio sul dispositivo ricevente.

   Per gli ID mittente promozionali numerici registrati per l'India, specifica l'ID mittente come parametro del numero di origine nella richiesta di invio SMS.

9. Espandere la sezione Attributi specifici del paese e specificare i seguenti attributi obbligatori per l'invio di messaggi SMS ai destinatari in India:

   - ID entità — L'ID entità o ID entità principale (PE) ricevuto dall'organismo di regolamentazione per l'invio di SMS a destinatari in India.

     Si tratta di una stringa personalizzata fornita da TRAI di 1-50 caratteri che identificherà in modo univoco l'entità registrata con TRAI.

   - ID modello — L'ID modello ricevuto dall'organismo di regolamentazione per l'invio di messaggi SMS ai destinatari in India.

     Si tratta di una stringa personalizzata fornita da TRAI ai di 1-50 caratteri che identificherà in modo univoco il modello registrato con TRAI. L'ID modello deve essere associato all'ID mittente specificato nel passaggio precedente e al contenuto del messaggio.

10. Seleziona Publish message (Pubblica messaggio).

Per informazioni sull'invio di messaggi SMS ai destinatari in altri paesi, consulta Pubblicazione su un telefono cellulare.

Risoluzione dei problemi dei messaggi SMS inviati ai destinatari in India

Di seguito sono riportati alcuni motivi per cui i vettori possono bloccare i messaggi SMS:

- Non è stato trovato alcun modello corrispondente al contenuto inviato.

  Contenuto inviato: **<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  Modello abbinato: Nessuno

  Problema: non ci sono modelli DLT che includono <#> o `{#var#}` all'inizio del modello registrato DLT.

- Il valore di una variabile supera i 30 caratteri.

  Contenuto inviato: **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  Modello abbinato: **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

Problema: il valore di "ABC Company - India Private Limited" nel contenuto inviato supera un singolo limite di caratteri `{#var#}` di 30.

- L'utilizzo delle maiuscole e minuscole nel messaggio non corrisponde all'utilizzo delle maiuscole e minuscole nel modello.

  Contenuto inviato: **`12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.`**

  Modello abbinato: **`{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.`**

  Problema: il nome della società aggiunto al modello DLT abbinato viene maiuscolo mentre il contenuto inviato ha cambiato parti del nome in minuscolo — "ABC Pvt. Ltd." contro "ABC PVT. LTD."

Requisiti per la registrazione dell'ID mittente per Singapore

I clienti Amazon SNS sono in grado di inviare traffico SMS a Singapore utilizzando un ID mittente che è stato registrato tramite il Singapore SMS Sender ID Registry (SSIR). SSIR è stato lanciato nel marzo 2022 tramite il Singapore Network Information Centre (SGNIC), di proprietà dell'Info-Communications Media Development Authority (IMDA) di Singapore, e consente alle organizzazioni di registrare il proprio ID mittente quando inviano SMS ai telefoni cellulari a Singapore.

Per utilizzare un ID mittente di Singapore registrato, è necessario ottenere un numero identificativo di entità (UEN) di Singapore, inviare una richiesta ad Amazon per consentire l'utilizzo dell'ID mittente registrato e infine completare il processo di registrazione tramite SSIR.

Se non registri il tuo ID entro il 30-01-2023, l'ID di qualsiasi messaggio inviato utilizzando un ID mittente verrà modificato in LIKELY-SCAM, in conformità con le regole dell'ente normativo. Dopo questa data, gli enti normativi continueranno a filtrare o bloccare il traffico non registrato a loro discrezione.

> **Important**
>
> Se richiedi l'ID mittente nelle regioni Amazon Pinpoint, utilizza la console Amazon Pinpoint per registrare l'ID mittente. Per completare manualmente il processo di registrazione per le regioni diverse da quelle Amazon Pinpoint, consulta la pagina relativa alla registrazione dell'ID mittente di Singapore.

Per poter continuare a inviare messaggi utilizzando l'ID utente di Singapore, devi completare la registrazione entro il 30-01-2023.
È molto importante completare le fasi della procedura di registrazione nell'ordine indicato di seguito. L'esecuzione di queste fasi in un ordine diverso potrebbe comportare il blocco del tuo ID mittente dal servizio o impedire che il tuo ID mittente venga mantenuto sul dispositivo mobile.
Fase 1. Registrazione per ottenere un numero di entità univoco (UEN) di Singapore
Fase 2. Se desideri registrare l'ID mittente nelle regioni Amazon Pinpoint, segui le istruzioni per la registrazione dell'ID mittente di Amazon Pinpoint.

- Per registrare un ID mittente se l'account non si trova in una regione Amazon Pinpoint, utilizza le istruzioni per la registrazione dell'ID mittente di Singapore per registrare manualmente l'ID mittente.
- Quando si inviano messaggi di testo SMS per conto di un'altra azienda, è necessaria una LOA (Letter of Authorization) dell'azienda.
- Non attendere l'approvazione o la modifica dello stato dopo aver inviato la richiesta di registrazione dell'ID mittente AWS. Vai immediatamente al passaggio 3.

Fase 3. Registrazione di un ID mittente presso il Singapore Network Information Centre (SGNIC)

Argomenti

Registrazione per ottenere un numero di entità univoco (UEN) di Singapore

Per effettuare la registrazione presso il SSIR è necessario innanzitutto ottenere un numero di entità univoco (UEN) di Singapore. L'UEN è un numero di entità univoco che si ottiene al momento della registrazione della propria attività presso l'ACRA (Account and Corporate Registry Authority). Per ulteriori informazioni, consulta la pagina relativa a chi deve registrarsi presso l'ACRA. Il tempo di elaborazione della richiesta dipende dalla facilità con cui l'ACRA è in grado di verificare le informazioni fornite.

Come registrare l'ID mittente di Singapore con Amazon Pinpoint

Dopo aver registrato il numero di entità univoco di Singapore (UEN), è possibile completare la procedura di registrazione dell'ID mittente nella console Amazon Pinpoint (solo per le regioni Amazon Pinpoint). Quando registri il tuo ID mittente, assicurati che le informazioni siano complete e accurate o la richiesta di registrazione potrebbe avere esito negativo.

> **Important**
>
> Per completare la registrazione, le informazioni inviate tramite la console Amazon Pinpoint verranno trasmesse ai nostri operatori partner.

Come registrare un ID mittente di Singapore

Utilizza questa procedura per la registrazione di un ID mittente per un account situato in una regione Amazon Pinpoint. Se il tuo account non si trova in una regione Amazon Pinpoint, consulta Procedura di registrazione manuale per completare la registrazione dell'ID mittente di Singapore.

1. Accedi alla console di gestione AWS e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in SMS and voice (Messaggi SMS e vocali), scegli Phone numbers (Numeri di telefono).
3. Nella scheda Sender ID registrations (Registrazioni ID mittente), scegli Create registration (Crea registrazione).
4. Seleziona Singapore come Paese di destinazione.
5. Nella sezione Company Information (Informazioni aziendali), procedi come segue:
   - In Company Name (Nome azienda), inserisci il nome della tua azienda esattamente come indicato nella registrazione dell'UEN.

- In Tax ID (ID fiscale), inserisci il numero UEN che hai ricevuto dall'ACRA.
- In Company Website (Sito web aziendale), inserisci l'URL del sito web della tua azienda.
- In Address 1 (Indirizzo 1), inserisci l'indirizzo della tua sede aziendale principale.
- Se pertinente, in Address 2 (Indirizzo 2), inserisci il numero di suite della tua sede centrale.
- In City (Città), inserisci la città della tua sede aziendale principale.
- In State (Stato), inserisci lo stato della tua sede aziendale principale.
- In Zip Code (Codice postale), inserisci il codice di avviamento postale della tua sede aziendale principale.
- In Country (Paese), inserisci il codice del Paese ISO a due cifre.

6. Nella sezione Contact Information (Informazioni di contatto), inserisci le seguenti informazioni:
   - In First Name (Nome), inserisci il nome del referente della tua azienda.
   - In Last Name (Cognome), inserisci il cognome del referente della tua azienda.
   - In Email Support (Indirizzo e-mail per l'assistenza), inserisci l'indirizzo e-mail della persona che sarà il punto di contatto per il supporto tecnico.
   - In Support Phone Number (Numero di telefono per l'assistenaza), inserisci il numero di telefono della persona che sarà il punto di contatto per per il supporto tecnico.
7. In Sender ID Information (Informazioni ID mittente), inserisci le seguenti informazioni:
   - In Sender ID (ID mittente), inserisci l'ID mittente che desideri mostrare nei tuoi messaggi.
   - In Registering on behalf of another brand/entity? (Registrazione per conto di un altro marchio o di un'altra entità?), seleziona True per rispondere di sì. Se non sei l'utente finale che invia i messaggi, allora stai agendo "in rappresentanza" di un altro marchio o di un'altra entità.
   - In Letter of authorization image – optional (Immagine della Letter of Authorization – facoltativo), se hai selezionato la casella Registering on behalf of another brand/entity? (Registrazione per conto di un altro marchio o di un'altra entità?), carica un'immagine della Letter of Authorization (LOA) completa. Il tipo di file supportato è PNG e la dimensione massima del file è di 400 KB. È possibile scaricare un modello di LOA qui.
   - In Sender ID connection – optional (Connessione con ID mittente – facoltativo), puoi aggiungere ulteriori dettagli sulla connessione tra l'ID mittente richiesto e il nome dell'azienda.
8. In Messaging Use Case (Caso d'uso per l'invio di messaggi), procedi in questo modo:
   - In Monthly SMS Volume (Volume SMS mensile) seleziona il numero di SMS che saranno inviati ogni mese.

- In Use Case Category (Categoria caso d'uso), seleziona uno dei seguenti tipi di casi d'uso in cui verrà utilizzato il numero:
  - Two-factor authentication (Autenticazione a due fattori): utilizza questa opzione per l'invio di codici di autenticazione a due fattori.
  - One-time passwords (Password monouso): utilizza questa opzione per l'invio di password monouso.
  - Notifications (Notifiche): per inviare ai tuoi utenti solo notifiche importanti.
  - Polling and surveys (Valutazioni e sondaggi): per interrogare gli utenti sulle loro preferenze.
  - Info on demand (Informazioni su richiesta): Questo serve per inviare agli utenti messaggi dopo che hanno inviato una richiesta.
  - Promotions and Marketing (Promozioni e marketing): per inviare ai tuoi utenti solo messaggi di marketing.
  - Other (Altro): utilizza questa opzione se il tuo caso d'uso non rientra in nessun'altra categoria. Assicurati di compilare il campo Use Case Details (Utilizza dettagli del caso d'uso) per questa opzione.
- Completa il campo Use Case Details – optional (Dettagli del caso d'uso – facoltativo) per aggiungere contesto alla categoria dei casi d'uso selezionata.

9. Nella sezione Messaging Samples (Invio di messaggi di esempio), esegui queste operazioni:
   - In Message Sample 1 (Messaggio di esempio 1), inserisci un esempio del corpo del messaggio SMS che verrà inviato agli utenti finali.
   - Se necessario, in Message Sample 2 – optional (Messaggio di esempio 2 – facoltativo) e Message Sample 3 – optional (Messaggio di esempio 3 – facoltativo), puoi inserire altri esempi del corpo del messaggio SMS da inviare.
   - Per ogni casella di testo Message Sample (Messaggio di esempio) si applica un limite massimo di 306 caratteri.
10. Una volta terminato, scegli Submit registration (Invia registrazione).

> **Important**
>
> Per verificare lo stato della registrazione, segui le indicazioni in Stato della registrazione dell'ID mittente di Singapore.

Non attendere l'approvazione o la modifica dello stato dopo aver inviato la richiesta di registrazione dell'ID mittente. Vai subito a Registrazione di un ID mittente presso il Singapore Network Information Centre (SGNIC).

Procedura di registrazione manuale per completare la registrazione dell'ID mittente di Singapore

Utilizza questa procedura per la registrazione di un ID mittente per un account non situato in una regione Amazon Pinpoint. Se il tuo account si trova in una regione Amazon Pinpoint, consulta Come registrare l'ID mittente di Singapore con Amazon Pinpoint.

1. Scarica il file Singapore_Sender_ID_Registration_LOA_Template.zip e inserisci le informazioni richieste.
2. Creare un caso con AWS Support.
3. Nella scheda Open support cases (Apri pratiche di supporto), scegli Create case (Crea pratica).
4. Scegli Looking for service limit increases? (Stai cercando un aumento dei limiti di servizio?) e, per il tipo di limite, seleziona SNS Text Messaging (Messaggi di testo SNS).
5. In Resource Type (Tipo di risorsa), scegli Sender ID Registration (Registrazione ID mittente).
6. Allega il documento LOA e invia la richiesta.

Registrazione di un ID mittente presso il Singapore Network Information Centre (SGNIC)

⚠ Warning

L'esecuzione di queste fasi in un ordine diverso potrebbe comportare il blocco del tuo ID mittente dal servizio o impedire che il tuo ID mittente venga mantenuto sul dispositivo mobile.

1. Innanzitutto, devi registrare l'ID mittente di Singapore (SG) per il tuo account con AWS tramite la console Amazon Pinpoint o eseguendo la registrazione manuale per le regioni non Amazon Pinpoint. Una volta completato questo passaggio, puoi passare alla fase successiva.
2. Collabora con l'SGNIC per la registrazione dell'ID mittente tramite la procedura indicata dal Singapore SMS Sender ID Registry (SSIR).
   - Durante il completamento della procedura, assicurati di elencare tutti i seguenti aggregatori partecipanti:

- AMCS SG Private Limited (Amazon Media Communications Services)
- Nexmo PTE LTD
- Sinch Singapore PTE LTD
- Telesign Singapore PTE LTD
- Twilio Singapore LTD LTD

> **Note**
>
> È necessario inviare una registrazione dell'ID mittente da ciascun account AWS necessario per utilizzare l'ID mittente.

Stato della registrazione dell'ID mittente di Singapore

Quando registri il tuo ID mittente di Singapore con Amazon SNS, la registrazione si troverà in uno dei seguenti cinque stati:

- Created (Creata): la registrazione è stata creata ma non inviata.
- Submitted (Inviata): la registrazione è stata inviata ed è in fase di revisione.
- Reviewing (In revisione): la registrazione è stata accettata ed è in fase di revisione. Per la revisione potrebbero essere necessarie da 1 a 3 settimane o, in alcuni casi, più tempo.
- Complete (Completata): la registrazione è stata approvata ed è possibile iniziare a utilizzare l'ID mittente.
- Requires Updates (Aggiornamenti necessari): è necessario apportare correzioni alla registrazione e inviarla di nuovo. Per ulteriori informazioni, consulta Modifica della registrazione di un ID mittente di Singapore. I campi che richiedono aggiornamenti saranno contrassegnati da un'icona di avviso e includeranno una breve descrizione del problema.

Per tutte le regioni diverse da quelle di Amazon Pinpoint, il supporto AWS invierà un'e-mail di conferma al momento della registrazione o aprirà una pratica con il supporto AWS.

- Nella scheda Open support cases (Apri pratiche di supporto), scegli Create case (Crea pratica).
- Selezionare Service limit increase (Aumento limiti del servizio).
- In Resource Type (Tipo di risorsa), scegli Sender ID Registration (Registrazione ID mittente) e per richiedere la modifica del limite seleziona General Inquiry (Richiesta generale).

Check your registration status (Controllare lo stato della registrazione)

1. Accedi alla console di gestione AWS e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in SMS and voice (Messaggi SMS e vocali), scegli Phone numbers (Numeri di telefono).
3. Nella scheda Sender ID registrations (Registrazioni ID mittente), scegli Sender ID (ID mittente).
4. Verrà visualizzato lo stato della registrazione di ciascun ID mittente.

Modifica della registrazione di un ID mittente di Singapore

Dopo l'invio, la richiesta di registrazione con Amazon Pinpoint si troverà nello stato Requires Updates (Aggiornamenti necessari) se si sono verificati problemi con la registrazione. In questo stato, è possibile modificare il modulo di registrazione. I campi che richiedono aggiornamenti saranno contrassegnati da un'icona di avviso e includeranno una breve descrizione del problema.

Per modificare un ID mittente

1. Apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in SMS and voice (Messaggi SMS e vocali), scegli Phone numbers (Numeri di telefono).
3. Nella scheda Sender ID Registration (Registrazione ID mittente), scegli il numero che desideri modificare e seleziona l'ID della registrazione.
4. Scegli Update registration (Aggiorna registrazione) per modificare il modulo e correggere i campi contrassegnati da un'icona di avviso.
5. Se stai effettuando la registrazione per conto di un altro marchio o un'altra entità, dovrai caricare nuovamente i file precedentemente inviati in Letter of authorization image – optional (Immagine della Letter of Authorization – facoltativo).
6.
> **Important**
>
> Ricontrolla tutti i campi per assicurarti che siano stati compilati correttamente.

7. Una volta terminato, seleziona Submit registration (Invia registrazione) per inviare nuovamente la registrazione.

Eliminazione della registrazione di un ID mittente di Singapore

Se non desideri più procedere con la registrazione di un ID mittente di Singapore, puoi eliminare la registrazione. È possibile eliminare una registrazione solo se si trova nello stato Created (Creata) o Requires Updates (Aggiornamenti necessari).

Per eliminare una registrazione

1. Apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in SMS and voice (Messaggi SMS e vocali), scegli Phone numbers (Numeri di telefono).
3. Nella scheda Sender ID (ID mittente), scegli l'ID della registrazione che desideri eliminare, quindi seleziona Delete Registration (Elimina registrazione).

Problemi con la registrazione di un ID mittente di Singapore

Se il tuo ID mittente di Singapore non viene accettato da Amazon Pinpoint, riceverai un messaggio che spiega le ragioni del rifiuto. Se hai domande in merito a un rifiuto e non trovi le risposte che cerchi nelle nostre best practice, puoi inviare una richiesta di informazioni ai nostri team di supporto.

Come inviare una richiesta di informazioni su un ID mittente di Singapore rifiutato

1. Apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel menu Supporto scegliere Centro di supporto.
3. Nella pagina di Supporto, scegli Create case (Crea caso).
4. In Tipo di caso, scegli Aumento dei limiti di servizio.
5. In Tipo di limite scegli SMS Pinpoint.
6. Nella sezione Richieste, procedere come segue:
   - In Resource Type (Tipo di risorsa), scegli Sender ID Registration (Registrazione ID mittente).
   - In Limit (Limite), scegli Registration Rejection Query (Domanda sul rifiuto di una richiesta di registrazione).
7. In Use case description (Descrizione del caso d'uso), inserisci l'ID mittente di Singapore non accettato e il motivo del rifiuto indicato.
8. In Opzioni contatto, per Lingua di contatto preferita, scegli la lingua che preferisci utilizzare quando comunichi con il team di supporto AWS.
9. In Metodo di contatto, scegli il metodo preferito per le comunicazioni con il team di supporto AWS.

10. Seleziona Submit (Invia).

Il team di supporto AWS fornirà informazioni sui motivi per cui la registrazione del tuo ID mittente è stata rifiutata nel tuo caso di supporto AWS.

Domande frequenti sulla registrazione dell'ID mittente di Singapore

Domande frequenti relative al processo di registrazione di un ID mittente di Singapore con Amazon Pinpoint

Ho già un ID mittente di Singapore?

Per verificare se possiedi un ID mittente di Singapore:

1. Apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in SMS and voice (Messaggi SMS e vocali), scegli Phone numbers (Numeri di telefono).
3. Nella scheda Sender ID Registration (Registrazione ID mittente), scegli l'ID mittente che desideri visualizzare e seleziona l'ID della registrazione.

Quanto tempo occorre per la registrazione?

In genere, il processo di revisione richiede da 1 a 3 settimane, ma in alcuni casi potrebbero essere necessarie fino a 5 settimane o più per verificare le informazioni con gli enti governativi.

Che cos'è un numero di entità univoco (UEN) e come posso ottenerne uno?

Un UEN è un ID aziendale di Singapore emesso dall'Accounting and Corporate Regulatory Agency (ACRA). Le aziende e le imprese di Singapore possono ottenere un UEN richiedendolo all'ACRA. L'UEN verrà emesso una volta superata la procedura di registrazione e incorporazione standard. Puoi richiedere un UEN all'ACRA tramite Bizfile.

Devo registrarmi per ottenere un ID mittente di Singapore?

Sì. Se non registri l'ID mittente di Singapore entro il 30-01-2023, qualsiasi messaggio che invii utilizzando un ID mittente avrà l'ID modificato in LIKELY-SCAM.

Come faccio a registrare un ID mittente di Singapore con Amazon Pinpoint?

Per registrare un ID mittente, segui le istruzioni riportate nella pagina sulla registrazione di un ID mittente di Singapore con Amazon Pinpoint.

Qual è lo stato della registrazione del mio ID mittente di Singapore e cosa significa?

Per verificare la registrazione e il relativo stato, segui le indicazioni riportate nello stato della registrazione dell'ID mittente di Singapore.

Quali informazioni devo fornire?

Dovrai fornire l'indirizzo dell'azienda, il contatto di un referente aziendale e un caso d'uso. Le informazioni richieste sono disponibili nella pagina relativa alla registrazione dell'ID mittente di Singapore con Amazon Pinpoint.

Cosa succede se la registrazione del mio ID mittente di Singapore non viene accettata?

Se la tua registrazione viene rifiutata, il suo stato verrà modificato in Requires Updates (Aggiornamenti necessari) e dovrai effettuare gli aggiornamenti richiesti seguendo le istruzioni riportate nella pagina relativa alla modifica della registrazione di un ID mittente di Singapore.

Di quali autorizzazioni ho bisogno?

Il ruolo/l'utente IAM che utilizzi per visitare la console Amazon Pinpoint deve disporre dell'autorizzazione `"sms-voice:*"`.

Numeri di origine

Un numero di origine è una stringa numerica che identifica il numero di telefono del mittente di un messaggio SMS. Quando si invia un messaggio SMS utilizzando un numero di origine, il dispositivo del destinatario mostra il numero di origine come numero di telefono del mittente. È possibile specificare diversi numeri di origine in base al caso d'uso.

> **Tip**
>
> Per visualizzare un elenco di tutti i numeri di origine esistenti nell'account AWS nel riquadro di navigazione della console Amazon SNS, scegliere Numeri di origine.

Il Support per i numeri di origine non è disponibile nei paesi in cui le leggi locali richiedono l'uso di ID mittente invece dei numeri di origine.

Argomenti

- 10DLC
- Numeri verdi
- Codici brevi
- Codici lunghi da persona a persona (P2P)
- Confronto numero prodotto negli Stati Uniti

10DLC

I corrieri statunitensi non supportano più l'utilizzo di messaggi SMS da applicazione a persona (A2P) su codici lunghi locali non registrati. Per i messaggi SMS A2P ad alto volume, gli operatori statunitensi offrono invece un nuovo tipo di codice lungo chiamato codici lunghi a 10 cifre (10DLC).

Important

A partire dal 26 gennaio 2023, i fornitori di SMS di Amazon SNS hanno introdotto nuovi processi di revisione manuale sulle campagne 10DLC per risolvere i problemi relativi allo spam SMS sollevati dai corrieri statunitensi. Puoi utilizzare codici brevi e numeri verdi come alternativa a 10DLC per inviare SMS negli Stati Uniti.
Al momento i nostri fornitori di SMS non hanno fornito target ai livello di servizio sulla durata delle revisioni della campagna 10DLC. Le revisioni vengono attivate quando un numero viene associato a una campagna 10DLC. Le revisioni richiedono più tempo rispetto a quello stimato di 14 giorni comunicato in precedenza da Amazon SNS.
Amazon SNS collabora quotidianamente con i fornitori di SMS per garantire che:

- I fornitori completino il prima possibile eventuali revisioni in sospeso della campagna 10DLC
- I fornitori assegnino priorità più elevate alle richieste AWS nei loro backlog

Puoi controllare lo stato delle campagne 10DLC eseguendo le istruzioni riportate in Campagne 10DLC. Se sono richieste informazioni aggiuntive per approvare una campagna 10DLC, il team del Supporto AWS ti invierà una notifica.
Potresti essere in grado di registrare un numero verde statunitense più velocemente piuttosto che ottenere 10 numeri DLC. Per ulteriori informazioni sui numeri verdi statunitensi e sul processo di registrazione, consultare Requisiti e processo di registrazione per i numeri verdi.

Che cos'è 10DLC?

10DLC è un tipo di codice lungo registrato con gli operatori per supportare messaggi SMS A2P ad alto volume utilizzando il formato del numero di telefono a 10 cifre. Amazon SNS non offre più codici lunghi locali come prodotto SMS, offre invece 10DLC. 10DLC non ha conseguenze se si utilizzano solo codici brevi e numeri verdi.

10DLC è un numero di telefono di 10 cifre utilizzato solo negli Stati Uniti. I messaggi inviati da un 10DLC ai destinatari mostrano un numero di 10 cifre come mittente. A differenza dei numeri verdi, 10DLC supporta sia messaggistica transazionale e promozionale e può includere qualsiasi prefisso statunitense.

Se si dispone di codici lunghi locali esistenti, è possibile richiedere che essi siano abilitati per 10DLC. A tale scopo, completare il processo di registrazione 10DLC e quindi inviare un ticket di supporto. Nel caso in cui si verifichi un problema con l'attivazione del codice lungo per 10DLC, riceverai una notifica e dovrai richiedere un nuovo 10DLC tramite la console Amazon Pinpoint (non Amazon SNS). Per informazioni su come archiviare un ticket di supporto per convertire un codice lungo, vedere Associazione di un codice lungo a una campagna 10DLC.

Per utilizzare un numero 10DLC, registra prima la tua azienda e crea una campagna 10DLC utilizzando la console Amazon Pinpoint (non Amazon SNS). AWS condivide queste informazioni con il Registro delle Campagne, una terza parte che approva o rifiuta la registrazione in base alle informazioni. In alcuni casi, la registrazione avviene immediatamente. Ad esempio, se ti sei registrato in precedenza nel Registro delle Campagne, potrebbero già avere le tue informazioni. Tuttavia, alcune campagne potrebbero richiedere almeno settimana per essere approvate. Dopo l'approvazione della tua azienda e campagna 10DLC, puoi acquistare un numero 10DLC e associarlo alla tua campagna. La richiesta di un 10DLC potrebbe richiedere fino a una settimana per essere approvata. Sebbene sia possibile associare più 10DLC a una singola campagna, non è possibile utilizzare lo stesso 10DLC in più campagne. Per ogni campagna che crei, devi avere un 10DLC unico.

Funzionalità 10DLC

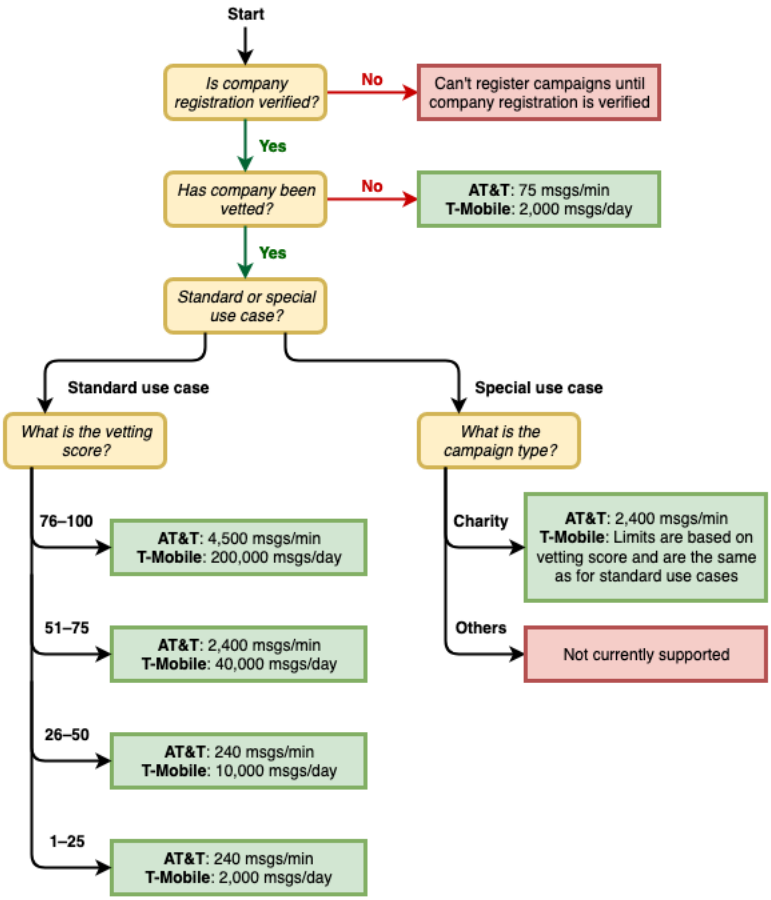
Le funzionalità dei numeri di telefono 10DLC dipendono dai gestori di telefonia mobile usati dai destinatari. AT&T fornisce un limite al numero di parti del messaggio che possono essere inviate ogni minuto per ciascuna campagna. T-Mobile fornisce un limite giornaliero di messaggi che può essere inviato per ogni azienda, senza limitazioni sul numero di parti del messaggio che possono essere inviate al minuto. Verizon non ha pubblicato limiti di velocità effettiva, ma utilizza un sistema di

filtraggio per 10DLC progettato per rimuovere spam, messaggi non richiesti e contenuti abusivi, con meno enfasi sulla velocità effettiva dei messaggi.

Le nuove campagne 10DLC associate a società non controllate possono inviare 75 parti di messaggio al minuto ai destinatari che utilizzano AT&T e 2.000 messaggi al giorno ai destinatari che utilizzano T-Mobile. Il limite aziendale è condiviso tra tutte le campagne 10DLC. Ad esempio, se hai registrato una società e due campagne, l'assegnazione giornaliera di 2.000 messaggi ai clienti di T-Mobile viene condivisa tra queste campagne. Analogamente, se registri la stessa azienda in più di un account AWS, l'assegnazione giornaliera è condivisa tra tali account.

Se le tue esigenze di velocità effettiva superano questi limiti, puoi richiedere la verifica della registrazione della tua azienda. Quando controlli la registrazione della tua azienda, un provider di verifica di terze parti analizza i dettagli della tua azienda. Il provider di verifica fornisce quindi un punteggio di controllo, che determina le funzionalità delle tue campagne 10DLC. È previsto un singolo addebito per il servizio di controllo. Per ulteriori informazioni, consulta Verifica della registrazione Amazon SNS 10DLC.

Il tasso di velocità effettiva varia a seconda di vari fattori, come ad esempio se la tua azienda è stata controllata o meno, i tipi di campagna e il punteggio di controllo. Il seguente diagramma di flusso mostra i tassi delle velocità effettive per varie situazioni.



I tassi di velocità effettiva per 10DLC sono determinati dai vettori mobili statunitensi in collaborazione con Campaign Registry. Né Amazon SNS né altri servizi di invio SMS possono aumentare la velocità effettiva 10DLC oltre questi tassi. Se hai bisogno di una velocità di trasmissione effettiva elevata e percentuali elevate di recapito su tutti i gestori statunitensi, ti consigliamo di utilizzare un codice breve. Per ulteriori informazioni su come ottenere un codice breve, consulta Richiesta di codici brevi dedicati per la messaggistica SMS con Amazon SNS.

Nozioni di base su 10DLC

Utilizza la console Amazon Pinpoint (non Amazon SNS) per richiedere il 10DLC. Effettua la procedura riportata di seguito per configurare 10DLC da utilizzare con le tue campagne 10DLC.

1. Registra la tua azienda.

   Prima di poter richiedere un 10DLC, la tua azienda deve essere registrata nel Registro delle Campagne; per informazioni, vedi Registrazione di un'azienda. La registrazione è solitamente istantanea, a meno che il Registro delle Campagne non richieda ulteriori informazioni. C'è una quota di registrazione una tantum per registrare la tua azienda, visualizzata nella pagina di

registrazione. Questa quota una tantum viene pagata separatamente dalle spese mensili per la campagna e dai 10DLC.

> **Note**
>
> I messaggi SMS di Amazon SNS sono disponibili nelle regioni in cui Amazon Pinpoint non è attualmente supportato. I casi sono due:
>
> a. Se utilizzi un account cloud commerciale, devi aprire la console Amazon Pinpoint nella regione Stati Uniti orientali (Virginia settentrionale) per registrare la tua azienda e la tua campagna 10DLC. Non richiedere un numero 10DLC.
>
> b. Utilizza invece la Console AWS Service Quotas per creare un caso di aumento del limite di servizio durante la richiesta del numero 10DLC per quella regione specifica. Per informazioni sulle regioni in cui Amazon Pinpoint è disponibile, consultare Amazon Pinpoint endpoints and quotas (Endpoint e quote di Amazon Pinpoint) nei Riferimenti generali di AWS.
>
> c. Se stai usando un account AWS GovCloud (US), apri la console Amazon Pinpoint nella regione degli Stati Uniti occidentali per registrare la tua azienda e la tua campagna 10DLC. Non richiedere un numero 10DLC. Utilizzare invece la Console Service Quotas di AWS per creare un caso di aumento del limite di servizio durante la richiesta del numero 10DLC per quella regione specifica. Per informazioni sulle regioni in cui Amazon Pinpoint è disponibile, consultare Amazon Pinpoint endpoints and quotas (Endpoint e quote di Amazon Pinpoint) nei Riferimenti generali di AWS.

2. (Facoltativo, ma consigliato) Richiedi il controllo

   Se la registrazione della tua azienda ha esito positivo, puoi iniziare a creare campagne 10DLC a basso volume e a uso misto. Queste campagne possono inviare 75 messaggi al minuto ai destinatari che utilizzano AT&T e la tua azienda registrata può inviare 2.000 messaggi al giorno ai destinatari che utilizzano T-Mobile. Se il tuo caso d'uso richiede una velocità effettiva di trasmissione superiore a questi valori, puoi richiedere il controllo della registrazione della tua azienda. Il controllo della registrazione della tua azienda può aumentare i tassi di velocità effettiva per le aziende e le campagne, ma non è garantito. Per ulteriori informazioni sul controllo, consulta Verifica della registrazione Amazon SNS 10DLC.

3. Registra la tua campagna.

   Dopo la registrazione della tua azienda, crea una campagna 10DLC e associala a una delle tue società registrate. Questa campagna viene inviata al Registro delle Campagne per l'approvazione.

Nella maggior parte dei casi, l'approvazione della campagna 10DLC è immediata, a meno che il Registro delle Campagne non richieda ulteriori informazioni. Per ulteriori informazioni, consulta Registrazione di una campagna 10DLC.

4. Richiedi il tuo numero 10DLC.

   Dopo l'approvazione della campagna 10DLC, puoi richiedere un 10DLC e associarlo alla campagna approvata. La tua campagna 10DLC può utilizzare solo un numero approvato. Per informazioni, consultare Richiesta di numeri 10DLC, numeri verdi e codici lunghi P2P per la messaggistica SMS con Amazon SNS.

Registrazione 10DLC e canoni mensili

Ci sono quote di registrazione e mensili associate all'utilizzo di 10DLC, come la registrazione della tua azienda e la campagna 10DLC. Questi sono separati da qualsiasi altro canone mensile addebitato da AWS. Per ulteriori informazioni, consulta la pagina Prezzi globali degli SMS Amazon SNS.

Registrazione di un'azienda

Prima di poter richiedere un 10DLC, devi registrare la tua azienda nel Registro delle Campagne.

> **Note**
>
> I messaggi SMS di Amazon SNS sono disponibili nelle regioni in cui Amazon Pinpoint non è attualmente supportato. In questi casi, apri la console Amazon Pinpoint nella regione Stati Uniti orientali (Virginia settentrionale) per registrare la tua azienda e la tua campagna 10DLC, ma non richiedere un numero 10DLC. Utilizzare invece la Console Service Quotas di AWS per creare un caso di aumento del limite di servizio durante la richiesta del numero 10DLC per quella regione specifica. Per informazioni sulle regioni in cui Amazon Pinpoint è disponibile, consultare Amazon Pinpoint endpoints and quotas (Endpoint e quote di Amazon Pinpoint) nei Riferimenti generali di AWS.

Stati di registrazione dell’azienda 10DLC

Quando registri la tua azienda o il tuo brand, viene restituito uno dei due stati: Non verificato o Verificato. Se lo stato per la registrazione della tua azienda è Non verificato, significa che c'è stato un problema con la registrazione. Ad esempio, il nome dell’azienda registrata fornito potrebbe non corrispondere esattamente al nome registrato dell’azienda associata al codice fiscale fornito. Se

riscontri un problema con i dettagli di registrazione della tua azienda, puoi correggerli. Per ulteriori informazioni sulla modifica dei dettagli di registrazione dell'azienda, consulta Modifica o eliminazione di un'azienda registrata.

Se lo stato della registrazione della tua azienda è Verificato, i dettagli di registrazione forniti sono accurati e puoi iniziare a creare campagne 10DLC.

Registrazione della tua azienda o del tuo brand

È sufficiente registrare la propria azienda una sola volta. Dopo la registrazione, puoi modificare la tua azienda e le informazioni di contatto. Per eliminare una società registrata, crea un caso con AWS Support. Per ulteriori informazioni sulla modifica o l'eliminazione dei dettagli aziendali, consulta Modifica o eliminazione di un'azienda registrata.

Per registrare un'azienda

1. Accedi a AWS Management Console e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in Impostazioni, scegli Numeri di telefono.
3. Nella scheda Campagne 10DLC, scegli Registrare l'azienda.

> **Note**
>
> La pagina Registrare l'aziendav mostra la Quota di iscrizione. Si tratta di una quota una tantum associata alla registrazione della tua azienda. Questo costo è separato da qualsiasi altro costo mensile. Ti viene addebitato quando registri la tua azienda o quando modifichi i dettagli di una registrazione aziendale esistente.

4. Nella sezione Informazioni generali, esegui queste operazioni:

   - In Nome legale dell'azienda, inserire il nome con cui la società è registrata. Il nome immesso deve corrispondere esattamente al nome della società associato al codice fiscale fornito.

> **Important**
>
> Utilizza l'esatto nome legale della tua azienda. Una volta inviato non puoi modificare queste informazioni. Informazioni errate o incomplete potrebbero comportare un ritardo o il rifiuto della registrazione.

- In Che tipo di forma legale è questa organizzazione, scegli l'opzione che meglio descrive la tua azienda.

> **Note**
>
> Le opzioni Governo statunitense e Senza scopo di lucro possono essere utilizzate solo per registrare organizzazioni con sede negli Stati Uniti. Se la tua organizzazione ha sede in un paese diverso dagli Stati Uniti, devi registrarti come Privato a scopo di lucro, indipendentemente dalla forma giuridica effettiva della tua organizzazione.

- Se hai scelto Pubblico a scopo di lucro nella fase precedente, inserisci il simbolo azionario della società e la borsa in cui è quotata.
- Scegliere il paese in cui l'azienda è registrata dall'elenco Paese di registrazione.
- Per Doing Business As (DBA) o nome del brand, inserisci qualsiasi altro nome con cui la tua azienda conduce trattative.
- In Codice fiscale, inserisci il codice fiscale della tua azienda. L'ID immesso dipende dal paese in cui la tua azienda è registrata.
  - Se stai registrando un'entità statunitense o non statunitense con un IRS Employer Identification Number (EIN), inserisci il tuo EIN a nove cifre. Il nome della società legale, l'EIN e l'indirizzo fisico immesso devono tutti corrispondere alle informazioni aziendali registrate con l'IRS.
  - Se stai registrando un'entità canadese, inserisci il tuo numero Corporation federale o provinciale. Non inserire il Business Number (BN) fornito dal CRA. Il nome della società legale, il numero Corporation e l'indirizzo fisico immesso devono tutti corrispondere alle informazioni aziendali registrate presso Corporations Canada.
  - Se stai registrando un'entità con sede in un altro paese, inserisci il codice fiscale principale per il tuo paese. In molti paesi, questa è la parte numerica del numero di partita IVA.
- In Verticale, scegliere la categoria che meglio descriva l'azienda che stai registrando.

5. Nella sezione Informazioni generali, eseguire queste operazioni:

- In Indirizzo/strada, inserisci l'indirizzo stradale fisico associato alla tua azienda.
- In Città, inserisci la città in cui si trova l'indirizzo fisico.

- In Stato o regione, inserisci lo stato o la regione in cui si trova l'indirizzo.
- In CAP/codice postale, inserisci il CAP o il codice postale associato per l'indirizzo.
- In Sito web aziendale, inserisci l'URL completo del sito web della tua azienda. Includi "http://" o "https://" all'inizio dell'indirizzo.
- In Indirizzo e-mail digita l'indirizzo e-mail.
- In Numero di telefono del supporto, immettere il numero di telefono con il codice del paese.

> **Note**
>
> Il Registro delle campagne richiede un indirizzo e-mail di contatto e un numero di telefono nel caso in cui debbano verificare le informazioni di registrazione con un rappresentante della tua azienda.

6. Al termine, scegli Save (Salva). In questo modo la registrazione dell'azienda viene inviata al Registro delle Campagne. Nella maggior parte dei casi, la registrazione viene accettata immediatamente e viene fornito uno stato.

Se lo stato per la registrazione della tua azienda è Verificato, puoi iniziare a creare campagne 10DLC a basso volume e a uso misto. Puoi utilizzare questo tipo di campagna per inviare fino a 75 messaggi al minuto ai destinatari che utilizzano AT&T e la tua azienda registrata può inviare 2.000 messaggi al giorno ai destinatari che utilizzano T-Mobile. Puoi anche inviare messaggi ai destinatari che utilizzano altri gestori statunitensi, come Verizon e US Cellular. Questi gestori non applicano rigorosamente i limiti di velocità effettiva, ma monitorano pesantemente i messaggi 10DLC per rilevare segni di spam e uso illecito.

Se il tuo caso d'uso richiede una velocità effettiva superiore a questi valori, puoi richiedere un controllo aggiuntivo della registrazione della tua azienda. Per ulteriori informazioni sul controllo della registrazione del brand, consulta Verifica della registrazione Amazon SNS 10DLC.

Se lo stato per la registrazione della tua azienda è Non verificato, si sono verificati problemi con le informazioni che hai fornito. Controlla le informazioni fornite e conferma che tutti i campi contengano le informazioni corrette. Puoi apportare modifiche ad alcune parti della registrazione aziendale nella console di Amazon Pinpoint. Per ulteriori informazioni sulla modifica dei dettagli di registrazione dell'azienda, consulta Modifica della registrazione di un'azienda 10DLC.

Verifica della registrazione Amazon SNS 10DLC

Se la registrazione della tua azienda ha esito positivo e desideri registrare una campagna con capacità di velocità effettiva più elevate, devi controllare la registrazione della tua azienda.

Quando controlli la tua registrazione, un'organizzazione di terze parti analizza i dettagli dell'azienda che hai fornito e restituisce un punteggio di controllo. Un punteggio di controllo elevato può portare a tassi di velocità effettiva più elevati per la tua azienda 10DLC e le campagne a essa associate. Tuttavia, non è garantito che il controllo aumenti la velocità effettiva.

I punteggi di controllo non vengono applicati in maniera retroattiva. In altre parole, se hai già creato una campagna 10DLC e successivamente controlli la registrazione della tua azienda, il punteggio di controllo non viene applicato automaticamente alla campagna esistente. Per questo motivo, dovresti controllare la tua azienda o il tuo brand prima di creare qualsiasi campagna 10DLC.

> **Note**
>
> È prevista una commissione non rimborsabile di $40 per il controllo della tua azienda o del tuo brand.

Per controllare la registrazione della tua azienda

1. Accedi a AWS Management Console e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in Messaggi SMS e vocali, scegliere Numeri di telefono.
3. Nella scheda Campagne 10DLC, scegliere l’azienda 10DLC che vuoi controllare.
4. Nella pagina dei dettagli dell'azienda, verso il fondo della pagina, scegliere Apply for vetting (Richiedi un controllo).
5. Nella finestra Apply for additional vetting (Richiedi un controllo aggiuntivo), scegli Submit (Invia).

   Per le aziende statunitensi, il processo di controllo richiede in genere circa un minuto per essere completato. Per le aziende con sede al di fuori degli Stati Uniti, il processo di controllo potrebbe richiedere molto più tempo, a seconda della disponibilità dei dati per quel paese.

Dopo aver inviato una richiesta di controllo, torni alla pagina dei dettagli dell'azienda. La sezione Risultati dei controlli aziendalil mostra lo stato e i risultati della tua richiesta di controllo. Al termine

del processo di controllo, questa tabella mostra un punteggio di controllo nella colonna Punteggio. Il punteggio di controllo determina le capacità della velocità effettiva 10DLC. La velocità effettiva varia in base al tipo di campagna creata. Se crei campagne 10DLC ad uso misto o relative al marketing, devi avere un punteggio di controllo più alto di quello necessario per altri tipi di campagne per ottenere tassi di velocità effettiva elevati. Per ulteriori informazioni sulle funzionalità dei numeri di telefono 10DLC, consulta Funzionalità 10DLC.

Se modifichi i dettagli della registrazione della tua azienda dopo aver completato il processo di controllo, puoi richiedere di controllare nuovamente la registrazione. Se modifichi solo Verticale per la registrazione della tua azienda, il punteggio di controllo non cambierà. Se modifichi dettagli diversi da Verticale, il risultato del controllo potrebbe cambiare. In entrambi i casi, ti viene addebitata nuovamente la commissione una tantum per il controllo.

Modifica o eliminazione di un'azienda registrata

Puoi modificare alcune delle informazioni di registrazione 10DLC per la tua azienda direttamente nella console Amazon Pinpoint. È inoltre possibile eliminare una registrazione di un'azienda 10DLC creando un caso nel Centro di supporto AWS .

Modifica della registrazione di un'azienda 10DLC

Dopo aver completato il processo di registrazione 10DLC per un'azienda, è possibile modificare i dettagli della registrazione.

Se viene visualizzato un messaggio di errore dopo aver modificato i dettagli di registrazione dell'azienda, potrebbero esserci altri problemi con la registrazione. Puoi aprire un ticket con AWS Support per richiedere maggiori informazioni.

Per modificare la registrazione dell'azienda

1. Apri la AWS SMS console all'indirizzo https://console.aws.amazon.com/sms-voice/.
2. Segui le istruzioni per modificare la registrazione nella Guida per l'utente di Amazon Pinpoint SMS.

Eliminazione della registrazione di un'azienda 10DLC

Per eliminare la registrazione di una società

1. Apri la AWS SMS console all'indirizzo https://console.aws.amazon.com/sms-voice/.

2. Segui le istruzioni per eliminare la registrazione nella Guida per l'utente di Amazon Pinpoint SMS.

Registrazione di una campagna 10DLC

Quando registri una campagna 10DLC, fornisci una descrizione del tuo caso d'uso e i modelli di messaggio che intendi utilizzare. Prima di poter creare e registrare una campagna 10DLC, la tua azienda deve essere registrata. Per informazioni sulla registrazione di una società, vedi Registrazione di un'azienda.

> **Note**
>
> Dopo aver registrato la tua azienda, Amazon Pinpoint mostra uno dei due stati per la registrazione: Verificato o Non verificato. Puoi completare il processo di registrazione della campagna 10DLC solo se lo stato della registrazione della tua azienda è Verificato. Sarai in grado di creare campagne a uso misto a basso volume.
> Se lo stato è Non verificato, di solito significa che alcuni dei dati che hai fornito al momento della registrazione della tua azienda non erano corretti. Non sarai in grado di creare campagne 10DLC se la tua azienda ha questo stato. Puoi modificare la registrazione della tua azienda per tentare di risolvere i problemi con la registrazione della tua azienda. Per ulteriori informazioni sulla modifica delle registrazioni aziendali 10DLC, consultare Modifica o eliminazione di un'azienda registrata.

In questa pagina, fornisci prima i dettagli sull'azienda per la quale stai creando la campagna 10DLC e poi quelli del caso d'uso della campagna stessa. Le informazioni contenute in questa pagina vengono quindi fornite al Registro delle Campagne per l'approvazione.

In questa sezione, sceglierai l'azienda per la quale stai creando la campagna 10DLC e fornirai ulteriori dettagli.

Registrazione di una campagna 10DLC

1. Accedi a AWS Management Console e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. In SMS e voce, scegli Numeri di telefono.
3. Nella scheda Campagne 10DLC, scegliere Creare una campagna 10DLC.

4. Nella pagina Creare una campagna 10DLC, nella sezione Informazioni sulla campagna, eseguire le seguenti operazioni:

   a. Per Nome dell'azienda, scegli l'azienda per la quale stai creando questa campagna. Se non hai ancora registrato l'azienda, devi farlo prima di procedere. Per informazioni sulla registrazione di una società, vedere Registrazione di un'azienda.

   b. In Nome della campagna 10DLC, immetti un nome per la campagna.

   c. In Verticale, scegli l'opzione che meglio rappresenta la tua azienda.

   d. In Messaggio di aiuto inserisci il messaggio che i tuoi clienti ricevono se inviano la parola chiave "HELP" al tuo numero di telefono 10DLC.

   e. Per Messaggio di arresto inserisci il messaggio che i tuoi clienti ricevono se inviano la parola chiave "STOP" al tuo numero di telefono 10DLC.

   > **Tip**
   >
   > I tuoi clienti possono rispondere ai tuoi messaggi con la parola "HELP" per saperne di più sui messaggi che ricevono da te. Possono anche rispondere a "STOP" per non ricevere messaggi da te. I gestori mobili statunitensi richiedono di fornire risposte a entrambe queste parole chiave.
   > Di seguito è riportato un esempio di risposta HELP conforme ai requisiti dei gestori mobili statunitensi:
   > **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
   > Nel seguente esempio viene mostrata la risposta STOP di una richiesta:
   > **`You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.`**
   > Le risposte a queste parole chiave devono contenere al massimo 160 caratteri.

5. Nella sezione Caso d'uso della campagna, eseguire le seguenti operazioni:

   a. Per Tipo di caso d'uso, se hai un caso d'uso relativo all'organizzazione benefica, scegli Speciale. In caso contrario, scegli Standard.

   b. Per Caso d'uso, scegli un caso d'uso più simile alla tua campagna dall'elenco dei casi d'uso preimpostati. La quota mensile per ogni caso d'uso viene visualizzata accanto al nome del caso d'uso.

> **Note**
>
> L'addebito mensile per la registrazione della campagna 10DLC viene mostrato accanto a ciascun tipo di caso d'uso. La maggior parte dei tipi di campagne 10DLC ha lo stesso costo mensile. Il costo per la registrazione di casi d'uso Misti a basso volume è inferiore rispetto ad altri tipi di casi d'uso. Tuttavia, le campagne miste a basso volume supportano tassi di velocità effettiva più bassi rispetto ad altri tipi di campagne.

c. Inserisci almeno un Esempio di messaggio SMS. Questo è il messaggio di esempio che intendi inviare ai tuoi clienti. Se prevedi di utilizzare più modelli di messaggio per questa campagna 10DLC, includili.

> **Important**
>
> Non utilizzare il testo segnaposto per i messaggi di esempio. I messaggi di esempio forniti devono riflettere i messaggi effettivi che intendi inviare nel modo più accurato possibile.

6. La sezione Attributi della campagna e dei contenuti contiene una serie di domande Sì o No relative alle caratteristiche particolari della campagna. Alcuni attributi sono obbligatori, quindi non è possibile modificare il valore di default.

   Assicurati che gli attributi che scegli siano applicabili alla tua campagna.

   Indica se ciascuna delle seguenti opzioni si applica alla campagna che stai registrando:

   - Accettazione dell'abbonato — Gli abbonati possono scegliere di ricevere messaggi relativi a questa campagna.
   - Opt-out dell'abbonato — Gli abbonati possono scegliere di non ricevere più messaggi relativi a questa campagna.
   - Aiuto sottoscrittore — Gli abbonati possono contattare il mittente del messaggio dopo aver inviato la parola chiave HELP.
   - Pooling numerico — Questa campagna 10DLC utilizza più di 50 numeri di telefono.
   - Prestiti diretti o accordi di prestito — La campagna include informazioni sui prestiti diretti o altri accordi di prestito.

- Link incorporato — La campagna 10DLC include un link incorporato. I link agli abbreviatori di URL comuni, ad esempio TinyUrl o Bit.ly, non sono consentiti. Tuttavia, è possibile utilizzare gli abbreviatori di URL che offrono domini personalizzati.
- Numero di telefono incorporato — La campagna include un numero di telefono incorporato che non è un numero di assistenza clienti.
- Marketing di affiliazione — La campagna 10DLC include informazioni dal marketing di affiliazione.
- Contenuti soggetti all'età — La campagna 10DLC include contenuti con età gated come definito dal vettore e dalle linee guida CTIA (Cellular Telecommunications and Internet Association).

7. Scegliere Create (Crea).

   Dopo aver inviato i dettagli di registrazione per la campagna, si apre la pagina SMS e voce. Viene visualizzato un messaggio che indica che la campagna è stata inviata ed è in fase di revisione. Puoi consultare lo stato della tua richiesta nella scheda Campagne 10DLC. Puoi controllare lo stato della tua registrazione nella scheda 10DLC, che sarà una delle seguenti opzioni:

   - Attiva — La tua campagna 10DLC è stata approvata. Puoi richiedere un numero di telefono 10DLC e assegnare quel numero alla tua campagna. Per ulteriori informazioni, consulta Richiesta di numeri 10DLC, numeri verdi e codici lunghi P2P per la messaggistica SMS con Amazon SNS.
   - In attesa— La tua campagna 10DLC non è ancora stata approvata. In alcuni casi, l'approvazione potrebbe richiedere una o più settimane. Se lo stato cambia, la console Amazon Pinpoint riflette tale modifica. Non ti informiamo di cambiamenti di stato.
   - Rifiutata— La tua campagna 10DLC è stata rifiutata. Per ottenere ulteriori informazioni, inviare una richiesta di supporto che includa l'ID campagna della campagna rifiutata.
   - Sospesa — Uno o più gestori hanno sospeso la tua campagna 10DLC. Per ottenere ulteriori informazioni, invia una richiesta di supporto che includa l'ID campagna della campagna sospesa. Amazon Pinpoint non include motivi di sospensione sulla console e non ti inviamo alcuna notifica se la tua campagna è sospesa.

8. Se il tuo 10DLC è approvato, puoi richiedere un numero 10DLC da associare a quella campagna. Per informazioni su come richiedere un numero 10DLC, consulta Richiesta di numeri 10DLC, numeri verdi e codici lunghi P2P per la messaggistica SMS con Amazon SNS.

Utilizzo di campagne 10DLC in più regioni AWS

Quando registri una società, tale società è disponibile nel tuo account AWS in tutte le regioni AWS. Tuttavia, lo stesso non vale per le campagne 10DLC. Una campagna 10DLC può essere utilizzata solo nella regione AWS in cui è stata registrata.

Se si prevede di utilizzare 10DLC in più di una regione AWS, è necessario registrare campagne 10DLC separate in ciascuna di queste regioni. Questo passaggio è necessario per soddisfare i requisiti del gestore. Ti viene addebitato un costo per ogni campagna che registri, anche se il caso d'uso è esattamente lo stesso.

La registrazione di più campagne ha il vantaggio aggiuntivo di aumentare la velocità effettiva dei messaggi inviati ai destinatari che utilizzano AT&T come gestore mobile, poiché AT&T fornisce tassi di velocità effettiva 10DLC per ogni campagna. Confrontalo con il modo in cui T-Mobile gestisce la velocità effettiva 10DLC, basato su un'allocazione giornaliera dei messaggi per ogni azienda (indipendentemente dal numero di campagne).

Modifica o eliminazione di una campagna 10DLC

Puoi modificare la risposta HELP, la risposta STOP e i messaggi di esempio per una campagna 10DLC utilizzando la console Amazon Pinpoint. Puoi anche eliminare le campagne 10DLC utilizzando la console.

Modifica di una campagna 10DLC

Dopo l'approvazione della campagna, puoi modificare i messaggi HELP, STOP e di esempio. È inoltre possibile aggiungere ulteriori messaggi di esempio. Le modifiche a questi campi non richiedono la nuova approvazione dal Registro delle campagne o dai carrier. Non è possibile modificare altri campi dopo l'approvazione della campagna 10DLC.

È possibile avere al massimo cinque messaggi di esempio. Non è possibile ridurre il numero di messaggi di esempio registrati originariamente. Ad esempio, se hai registrato la tua campagna con tre SMS di esempio, non puoi ridurre il numero di messaggi SMS di esempio a meno di tre.

> **Note**
>
> Se si desidera modificare campi diversi da HELP, STOP e messaggi di esempio, è necessario prima eliminare la campagna 10DLC e quindi ricreare la campagna per includere le informazioni aggiornate.

Per modificare una campagna 10DLC

1. Accedere a AWS Management Console e aprire la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in Impostazioni, scegliere Messaggi SMS e vocali.
3. Nella scheda Campagne 10DLC, scegliere la campagna 10DLC da modificare.
4. Nella sezione Messaggi della campagna della pagina dei dettagli della campagna, scegliere Edit (Modifica).
5. Aggiornare uno dei seguenti campi:

   - Messaggio di aiuto
   - Messaggio di arresto
   - Messaggio SMS di esempio

   Non è possibile eliminare un messaggio di esempio precedentemente aggiunto o eliminare il contenuto di un messaggio di esempio in modo che il campo sia vuoto. Se si elimina il contenuto di un messaggio senza sostituire tale contenuto, il messaggio originale verrà utilizzato durante l'aggiornamento.
6. Scegliere Update (Aggiorna). Viene visualizzato un banner di conferma che ti informa che i messaggi della campagna sono stati aggiornati.

Eliminazione di una campagna 10DLC

Puoi eliminare una campagna 10DLC utilizzando la console Amazon Pinpoint. Prima di eliminare una campagna 10DLC, devi rimuovere prima tutti i numeri di telefono associati alla campagna.

> **Important**
>
> Quando rimuovi un numero 10DLC da una campagna, non hai più accesso a quel numero. Inoltre, le campagne 10DLC eliminate non possono essere ripristinate.

Per modificare o eliminare una campagna 10DLC

1. Accedere a AWS Management Console e aprire la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.

2. Nel pannello di navigazione, in Messaggi SMS e vocali, scegliere Numeri di telefono.
3. Nella scheda Campagne 10DLC scegliere la campagna da modificare.
4. Nella sezione Numeri di telefono, annotare i numeri di telefono associati alla campagna.
5. Nella scheda Numeri di telefono, scegliere il numero 10DLC da rimuovere, quindi scegliere Remove phone number (Rimuovi numero di telefono).

> **Note**
>
> Questo passaggio è richiesto solo se hai più numeri di telefono 10DLC associati alla campagna. Se hai un solo numero di telefono associato alla campagna 10DLC, questo numero apparirà nella scheda Campagne 10DLC. Annotare il numero visualizzato nella scheda.

6. Nella casella di conferma immettere **`delete`**, quindi scegliere Confirm (Conferma). Nella parte superiore della pagina SMS e voce viene visualizzato un messaggio di operazione riuscita.
7. Ripetere i due passaggi precedenti per ogni numero 10DLC associato alla campagna.
8. Dopo aver rimosso i numeri associati alla campagna 10DLC, scegliere la scheda Campagne 10DLC.
9. Scegliere la campagna 10DLC da eliminare.
10. Nell'angolo in alto a destra della pagina Dettagli della campagna 10DLC, scegliere Delete (Elimina).
11. Nella casella di conferma immettere **`delete`**, quindi scegliere Confirm (Conferma). Nella parte superiore della pagina SMS e voce viene visualizzato un messaggio di operazione riuscita.

Associazione di un codice lungo a una campagna 10DLC

Se disponi di un codice lungo esistente, puoi associare quel codice lungo a una delle tue campagne 10DLC correnti presentando una richiesta di supporto. Il codice lungo che associ alla campagna 10DLC può essere utilizzato solo con quella campagna e non può essere utilizzato per nessun'altra campagna 10DLC. Mentre il tuo codice lungo viene migrato a 10DLC, sarai comunque in grado di usarlo. Fino a quando non sarà approvato, non potrai più utilizzarlo per nessuna campagna 10DLC.

Al momento della presentazione della richiesta, avrai bisogno di:

- I codici lunghi da associare a una campagna 10DLC
- L'ID campagna 10DLC da associare al codice lungo

> **Note**
>
> Prima di poter associare codici lunghi a una campagna, è necessario che la campagna 10DLC sia registrata. Se non hai ancora creato e registrato una campagna 10DLC, consulta Registrazione di una campagna 10DLC.

Per assegnare un codice lungo a 10DLC

1. Accedi alla AWS Management Console e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Su Impostazioni, e quindi in SMS e messaggi vocali, scegliere la scheda Numeri di telefono.
3. Scegli il codice lungo che desideri convertire in un 10DLC.
4. Per aprire Support Center, scegliere Assegna alla campagna 10DLC.
5. Per il tipo di caso, scegliere Service limit increase (Aumento dei limiti di servizio).
6. In Tipo di limite scegliere Pinpoint.
7. Nella sezione Richieste, scegliere la Regione e poi per il Limitescegliere 10 DLC - Associare il codice lungo statunitense esistente alla campagna 10DLC.
8. In Descrizione del caso, per Descrizione del caso d'uso, assicurati di includere l'ID della campagna 10DLC e i numeri di codice lungo che vuoi associare alla campagna. Puoi includere più codici lunghi nella richiesta, ma devi includere solo un ID campagna.
9. In Opzioni contatto, per Lingua di contatto preferita, scegli la lingua che si preferisci utilizzare quando comunichi con il team di supporto AWS.
10. In Metodo di contatto, scegli il metodo preferito per le comunicazioni con il team di supporto AWS.
11. Seleziona Submit (Invia).

Accesso tra account 10DLC

Ogni numero di telefono 10DLC è associato a un singolo account in una singola regione AWS. Se desideri usare lo stesso numero di telefono 10DLC per inviare messaggi in più di un account o regione, sono disponibili due opzioni:

1. Puoi registrare la stessa azienda e la stessa campagna in ciascuno dei tuoi account AWS. Queste registrazioni sono gestite e addebitate separatamente. Se registri la stessa azienda in più

account AWS, il numero di messaggi che è possibile inviare ai clienti di T-Mobile ogni giorno è condiviso tra ciascuno di questi account.

2. È possibile completare il processo di registrazione 10DLC in un account AWS e utilizzare AWS Identity and Access Management (IAM) per concedere ad altri account l'autorizzazione di inviare il tuo numero 10DLC.

> **Note**
>
> Questa opzione consente un effettivo accesso multi-account ai numeri di telefono 10DLC. Tuttavia, tieni presente che i messaggi inviati dai tuoi account secondari vengono trattati come se fossero stati inviati dal tuo account principale. Le quote e la fatturazione vengono conteggiate in base a questo account e non a tutti gli account secondari.

Configurazione dell'accesso multi-account utilizzando le policy IAM

Puoi utilizzare i ruoli IAM per associare altri account al tuo account principale. Quindi, puoi delegare le autorizzazioni di accesso dal tuo account principale agli account secondari consentendo loro l'accesso ai numeri 10DLC nell'account principale.

Per concedere l'accesso a un numero 10DLC nel tuo account principale

1. Se non è già stato fatto, completa la procedura di registrazione 10DLC nell'account principale. Questo processo prevede tre fasi:
   - Registra l'azienda. Per ulteriori informazioni, consulta Registrazione della tua azienda o del tuo brand per l'utilizzo con 10DLC.
   - Registra la tua campagna 10DLC (caso d'uso). Per ulteriori informazioni, consulta Registrazione di una campagna 10DLC.
   - Associa un numero di telefono alla tua campagna 10DLC. Per ulteriori informazioni, consulta Associazione di un codice lungo a una campagna 10DLC.
2. Creazione di un ruolo IAM nell'account principale che consente a un altro account di chiamare l'operazione API `Publish` per il tuo numero di telefono 10DLC. Per ulteriori informazioni sulla creazione di ruoli, consulta Creazione di ruoli IAM nella Guida per l'utente di IAM.
3. Delega e verifica le autorizzazioni di accesso dal tuo account principale utilizzando i ruoli IAM con qualsiasi altro account che deve utilizzare i tuoi numeri 10DLC. Ad esempio, potresti

delegare l'autorizzazione di accesso dall'account di produzione per l'account di sviluppo. Consulta Delega dell'accesso su account AWS usando ruoli IAM nella Guida per l'utente IAM per ulteriori informazioni sulla delega e il test delle autorizzazioni.

4. Utilizzando il nuovo ruolo, inviare un messaggio utilizzando un numero 10DLC dall'account principale. Per ulteriori informazioni sull'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

Ottenere informazioni sui problemi di registrazione 10DLC

In alcune situazioni, potresti ricevere un messaggio di errore quando tenti di registrare la tua azienda o la campagna 10DLC.

Problemi di registrazione dell'azienda

Quando registri la tua azienda, vedi uno dei due stati di registrazione: Verificato o Non verificato. Se lo stato di registrazione dell'azienda è Verificato, la registrazione della tua azienda ha avuto esito positivo. Puoi iniziare a creare campagne 10DLC.

Se lo stato per la registrazione della tua azienda è Non verificato, ci sono stati problemi con le informazioni che hai fornito. La console Amazon Pinpoint fornisce informazioni sui motivi per cui la registrazione della tua azienda ha ricevuto questo stato.

Per visualizzare i problemi di registrazione per la tua azienda 10DLC

1. Accedi a AWS Management Console e apri la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/.
2. Nel pannello di navigazione, in SMS, scegli Numeri di telefono.
3. Nella scheda Campagne 10DLC, nell'elenco delle campagne, scegli il nome della società su cui desideri trovare ulteriori informazioni.
4. La pagina di dettaglio dell'azienda contiene informazioni sui problemi identificati nella registrazione. Se un campo nella sezione Informazioni aziendali contiene un simbolo di avviso, il problema della registrazione è correlato alle informazioni in quel campo.

   Controlla le informazioni fornite e conferma che tutti i campi contengono le informazioni corrette. Puoi modificare la registrazione aziendale nella console Amazon Pinpoint. Per ulteriori informazioni sulla modifica dei dettagli di registrazione dell’azienda, consulta Modifica o eliminazione di un’azienda registrata.

Problemi di registrazione della campagna

Quando registri la campagna 10DLC, è possibile che venga visualizzato un messaggio di errore in determinate situazioni.

Se non riesci a identificare il problema della registrazione, puoi creare un caso nel Centro AWS Support per richiedere ulteriori informazioni. Usa le procedure riportate di seguito per creare un caso di supporto AWS. Il team di supporto AWS fornirà informazioni sui motivi per cui la registrazione della campagna 10DLC è stata rifiutata.

Per inviare una richiesta di informazioni su una campagna 10DLC rifiutata

1. Accedi all'AWS Management Console all'indirizzo https://console.aws.amazon.com/.
2. Nel menu Supporto scegliere Centro di supporto.
3. Nella scheda I tuoi casi di supporto, scegli Create caso.
4. Scegli il collegamento Cerchi aumenti del limite di servizio?, quindi completa quanto segue:
   - Per Tipo di limite, scegli Pinpoint SMS.
5. In Requests (Richieste), completa le seguenti sezioni:
   - Per la Regione, scegli la Regione AWS in cui hai tentato di registrare la campagna.

     > **Note**
     >
     > La regione è obbligatoria nella sezione Richieste. Anche se hai fornito queste informazioni nella sezione Dettagli del caso, devi includerle anche qui.

   - In Tipo di risorse, scegli Registrazione 10DLC.
   - Per Limite, scegli Rifiuto della registrazione della campagna aziendale o 10DLC.
6. Per Nuovo valore limite, scegli l'aumento del limite per il tipo di limite. Questo valore è in genere **1**.
7. In Descrizione del caso, inserisci l'ID della campagna 10DLC rifiutata.
8. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Se si includono più richieste, specificare le informazioni necessarie per ciascuna. Per le informazioni richieste, consulta le altre sezioni all'interno di Richiesta di supporto per la messaggistica SMS con Amazon SNS.

9. In Opzioni di contatto, per Lingua di contatto preferita, scegli la lingua in cui desideri ricevere le comunicazioni per questo caso.
10. Al termine, scegli Submit (Invia).

Numeri verdi

Un numero verde (toll-free number (TFN)) è un numero di 10 cifre che inizia con uno dei seguenti prefissi: 800, 888, 877, 866, 855, 844 o 833. È possibile utilizzare i numeri TFN solo per inviare messaggi transazionali.

Important

Gli operatori di telefonia mobile statunitensi hanno recentemente modificato le normative e richiederanno che tutte le aziende che dispongono di un numero verde (TFN) completino un processo di registrazione presso un organo di regolamentazione entro il 30 settembre 2022. Verificare lo stato del TFN accedendo a the section called “Stato della registrazione del numero verde”. Per informazioni sulla registrazione della tua azienda, consulta the section called “Registrazione del numero verde”.
Possono essere necessari fino a 15 giorni lavorativi affinché la registrazione venga elaborata.
Aggiornamento del 3 marzo 2023: a partire dal 1° aprile 2023, i gestori di telefonia mobile applicheranno le seguenti soglie a livello di settore per i messaggi inviati mediante qualsiasi numero verde non registrato:

- Limite giornaliero: 500 messaggi al giorno, viene ripristinato alle 00:00 PT
- Limite settimanale: 1.000 messaggi a settimana, viene ripristinato la domenica alle 00:00 PT
- Limite mensile: 2.000 messaggi, ripristinato alla fine del mese solare alle 00:00 PT

Aggiornamento del 19 settembre 2022: a partire dal 1° ottobre 2022, i gestori di telefonia mobile applicheranno le seguenti soglie a livello di settore per i messaggi inviati mediante qualsiasi numero verde non registrato:

- Limite giornaliero: 2.000 messaggi
- Limite settimanale: 12.000 messaggi
- Limite mensile: 25.000 messaggi

È consigliabile completare la registrazione il prima possibile. I messaggi inviati tramite TFN non registrati saranno in base al miglior tentativo. I messaggi saranno soggetti a maggiori filtri e blocchi nel tempo man mano che i gestori continueranno a limitare il traffico non registrato.

Argomenti

Linee guida per l'utilizzo di numeri verdi

I numeri TFN sono in genere utilizzati per la messaggistica transazionale, come per l'invio di una conferma di registrazione o di una password monouso, e solo negli Stati Uniti. Possono essere utilizzati sia per l'invio di messaggi vocali sia per SMS. La velocità di trasmissione effettiva media è di tre parti di messaggio al secondo (MPS). Tuttavia, questa velocità di trasmissione effettiva è condizionata dalla codifica dei caratteri. Per ulteriori informazioni su come la codifica dei caratteri influisce sulle parti del messaggio, consulta Limiti relativi ai caratteri per gli SMS in Amazon SNS. Per informazioni sulla registrazione di un numero TFN, consulta Requisiti e processo di registrazione per i numeri verdi.

Ogni account può avere fino a cinque TFN. Se invii più di 15 SMS al secondo ma meno di 100, ti consigliamo di registrare uno o più ID di origine 10DLC. Se i tuoi casi d'uso richiedono l'invio di più di 100 messaggi di testo al secondo, ti consigliamo di acquistare e registrare uno o più Codici brevi.

Quando si utilizza un numero TFN come numero di origine, attenersi alle seguenti linee guida:

- Non utilizzare URL abbreviati creati da abbreviatori di URL di terze parti, poiché è probabile che questi messaggi vengano filtrati come spam.

Se è necessario utilizzare un URL abbreviato, considerare l'utilizzo di un Numero 10DLC o Codice breve. L'utilizzo di codici brevi e 10DLC richiede la registrazione del modello di messaggio, in cui è possibile specificare un URL abbreviato.

- Tenere presente che le risposte parola-chiave per rifiutare (STOP (Ferma)) e acconsentire (UNSTOP (Riprendi)) sono impostate a livello di operatore. Non è possibile modificare queste parole chiave o altre parole chiave. Inoltre, non è possibile modificare i messaggi inviati quando gli utenti rispondono con STOP (Ferma) e UNSTOP (Riprendi).
- Non inviare messaggi uguali o simili utilizzando più numeri verdi. I corrieri chiamano questa pratica snowshoe spam o pool di numeri e indirizzare questi messaggi per il filtraggio.
- Tutti i messaggi relativi ai seguenti settori possono essere considerati soggetti a restrizioni e sono soggetti a filtri pesanti o a blocchi definitivi. Ciò può includere codice OTP (One-Time Password) e autenticazione a più fattori (MFA) per servizi relativi a categorie limitate.

Se ti è stata negata la registrazione perché si tratta di un caso d'uso non conforme e ritieni che questa designazione non sia corretta, puoi inviare una richiesta tramite l'assistenza. Per informazioni dettagliate su come eseguire questa operazione, consulta Problemi di registrazione.

La tabella seguente illustra i tipi di contenuti con restrizioni:

Categoria	Esempi
Gioco d'azzardo	• App/siti Web • Casinò • Lotterie
Servizi finanziari ad alto rischio	• Auto prestiti • Criptovalute • Recupero crediti • Prestiti Payday • Prestiti a breve termine • Mutui ipotecari • Prestiti per studenti • Avvisi di azioni
Remissione debito	• Consolidamento debito

Categoria	Esempi
	• Riduzione debito • Programmi di riparazione crediti
et-rich-quick Schemi G	• ork-from-home Programmi W • Opportunità di investimento in rischio • Sistemi di marketing piramidali o multilivello
Sostanze vietate/controllate	• Cannabis/CBD
Phishing	• Tentativi di convincere gli utenti a rivelare informazioni personali o informazioni di accesso al sito web
S.H.A.F.T.	• Sex • Odio • Alcol • Armi da fuoco • Tabacco/Vape

Acquisto di un numero verde

Per acquistare TFN, usa la console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/sms-voice/. Per ulteriori informazioni, consulta Requisiti e processo di registrazione per i numeri verdi.

Attualmente, Amazon Pinpoint SMS supporta numeri verdi per messaggi vocali e SMS. Amazon SNS supporta solo la messaggistica SMS.

Requisiti e processo di registrazione per i numeri verdi

> **Important**
>
> Se un numero TFN viene utilizzato per qualcosa di diverso dal caso d'uso specificato, può essere revocato.

Casi d'uso per i numeri verdi

Amazon SNS ha una capacità limitata di inviare messaggi nei casi in cui i messaggi vengono bloccati (ad esempio, casi d'uso relativi a sostanze controllate o phishing) o quando sono previsti livelli elevati di filtraggio (ad esempio, messaggi finanziari ad alto rischio). Potrebbe non essere possibile registrare i numeri TFN associati ai casi d'uso dei contenuti limitati definiti in Linee guida per l'utilizzo di numeri verdi.

Registrazione del numero verde

Dopo aver acquistato un TFN, devi registrare il numero. Per istruzioni su come eseguire questa operazione, consulta la procedura di registrazione del numero verde nella Guida per l'utente di Amazon Pinpoint SMS.

Registrazione self-service per numeri verdi nelle regioni SMS di Amazon Pinpoint

Se hai richiesto il TFN nelle regioni Amazon Pinpoint SMS, completa la procedura di registrazione dell'azienda direttamente nella console Amazon Pinpoint SMS utilizzando le istruzioni disponibili nel modulo di registrazione dei numeri verdi statunitensi nella Guida per l'utente di Amazon Pinpoint SMS.

Quando si registra il numero TFN, assicurarsi che le informazioni siano complete e accurate o la registrazione potrebbe essere respinta. Le informazioni inserite devono corrispondere esattamente alla sede principale della società.

Procedura di registrazione manuale basata su moduli per numeri verdi in regioni diverse dalle regioni SMS di Amazon Pinpoint

1. Scarica questo US_TFN_Registration.zip e utilizza il modulo di registrazione di esempio (AWS US Toll-Free Registration Form-Business - Final.docx) per completare le informazioni richieste nel file CSV di registrazione TFN (BulkustFN - Final.csv).

   Ogni richiesta di registrazione o caso d'uso può avere un massimo di cinque numeri TFN. Se si ritiene di avere diritto a un'esenzione da questa regola, fornire una spiegazione dettagliata del motivo. Elencare tutti i numeri di telefono associati alla registrazione o al caso d'uso.
2. Creare un caso con AWS Support. Allegare il file CSV completo al caso e inviare la richiesta di registrazione TFN.
3. Scegli Crea caso, quindi scegli Cerchi aumenti del limite di servizio?
4. Per il tipo di limite, scegli Messaggi di testo SNS.

5. In Resource Type (Tipo di risorsa), scegli 10DLC or Toll-free number registration (Registrazione con numero 10DLC o verde).
6. Allega il documento US_TFN_Registration e seleziona Submit (Invia) per inviare la richiesta.

Punto chiave da tenere presente

1. L'elaborazione delle registrazioni può richiedere fino a due settimane dopo l'invio di tutte le informazioni richieste. Se le informazioni sono mancanti o incomplete, il processo di registrazione verrà ritardato. Se la registrazione viene rifiutata, ti aiuteremo a scoprire il motivo per cui è stata rifiutata e ti suggeriremo metodi per migliorare la tua campagna in modo che possa essere registrata.
2. I numeri TFN funzionano bene per casi d'uso transazionali come l'autenticazione a più fattori (MFA) in cui è richiesta una velocità di trasmissione effettiva limitata. Ogni numero TFN può inviare fino a tre parti di messaggi di testo al secondo e ogni account cliente può avere fino a cinque numeri TFN. Se invii più di 15 SMS al secondo ma meno di 100, ti consigliamo di registrare uno o più ID di origine 10DLC. Se i tuoi casi d'uso richiedono l'invio di più di 100 messaggi di testo al secondo, ti consigliamo di acquistare e registrare uno o più codici brevi. Per ulteriori dettagli, consulta Linee guida per l'utilizzo di numeri verdi.

Stato della registrazione del numero verde

Per verificare lo stato della registrazione, consulta Verifica lo stato della registrazione nella Guida per l'utente di Amazon Pinpoint SMS.

Modifica, eliminazione ed eliminazione della registrazione

Utilizza la Guida per l'utente di Amazon Pinpoint SMS per eseguire le seguenti attività:

- Modifica la tua registrazione
- Annulla la tua registrazione
- Eliminare la registrazione
- Visualizza le risorse per la registrazione

Problemi di registrazione

Se la registrazione del numero verde non viene accettata, verrà visualizzato un messaggio che spiega perché è stata rifiutata.

Per inviare una richiesta di informazioni su un numero verde rifiutato

1. Accedi all' AWS Management Console indirizzo https://console.aws.amazon.com/.
2. Nel menu Supporto scegliere Centro di supporto.
3. Nella scheda I tuoi casi di supporto, scegli Create caso.
4. Scegli il collegamento Cerchi aumenti del limite di servizio?, quindi completa quanto segue:
   - In Tipo di limite scegli SMS Pinpoint.
5. In Requests (Richieste), completa le seguenti sezioni:
   - Per la Regione, in cui hai tentato di registrare la campagna.

     > **Note**
     >
     > La regione è obbligatoria nella sezione Richieste. Anche se hai fornito queste informazioni nella sezione Dettagli del caso, devi includerle anche qui.

   - In Tipo di risorse, scegli Registrazione 10DLC o TFN.
   - Per Limite, scegli Rifiuto della registrazione della campagna o dell'azienda.
6. Per Nuovo valore limite, scegli l'aumento del limite per il tipo di limite. Questo valore è in genere **1**.
7. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Per le informazioni richieste, consulta le altre sezioni all'interno di Richiesta di supporto per la messaggistica SMS con Amazon SNS.
8. In Descrizione del caso, inserisci il numero verde rifiutato.
9. In Opzioni di contatto, per Lingua di contatto preferita, scegli la lingua in cui desideri ricevere le comunicazioni per questo caso.
10. Al termine, scegli Submit (Invia).

Domande frequenti sui numeri verdi

Domande frequenti sulla procedura di registrazione del numero TFN.

Al momento possiedo un numero verde?

Per verificare se possiedi un numero verde

- Apri la console Amazon Pinpoint SMS all'indirizzo https://console.aws.amazon.com/sms-voice/.
- Nel pannello di navigazione, in SMS, scegli Numeri di telefono.
- Il tipo di numero TFN è elencato come numero verde.

Devo registrare il mio numero verde?

Sì. Per continuare a utilizzare un numero TFN che possiedi attualmente, devi registrarlo prima del 30 settembre 2022. Se acquisti un nuovo numero TFN dopo il 30 settembre 2022, devi eseguire la registrazione prima di poter inviare messaggi.

Come faccio ad acquistare un numero verde?

Segui le istruzioni riportate in Richiedere un numero di telefono utilizzando la console SMS di Amazon Pinpoint per acquistare un TFN.

Come faccio a registrare il mio numero verde?

Per registrare un numero TFN, segui le indicazioni riportate in the section called “Registrazione del numero verde”.

Qual è lo stato di registrazione del mio numero verde e cosa significa?

Per verificare la registrazione e lo stato, segui le indicazioni in the section called “Stato della registrazione del numero verde”.

Quali informazioni devo fornire?

Dovrai fornire l'indirizzo dell'azienda, il contatto di un referente aziendale e un caso d'uso per il numero TFN. Puoi trovare le informazioni richieste all'indirizzo the section called “Registrazione del numero verde”.

Cosa succede se la mia registrazione viene rifiutata?

Se la registrazione viene rifiutata, lo stato viene modificato in Requires Updates (Richiede aggiornamenti). Per effettuare gli aggiornamenti, consulta the section called “Modifica, eliminazione ed eliminazione della registrazione”.

Di quali autorizzazioni ho bisogno?

L'utente/ruolo IAM che usi per visitare la console Amazon Pinpoint SMS deve avere *l'autorizzazione «sms-voice: *»*, altrimenti riceverai un errore di accesso negato.

Vantaggi e svantaggi dei numeri verdi

Vantaggi

Gli originatori che utilizzano numeri verdi hanno un MPS più elevato rispetto ai codici lunghi e una buona efficienza di recapito.

Svantaggi

Non c'è alcun controllo sulle negazioni o le conferme dei consensi, in quanto queste sono gestite a livello di operatore.

Non includere URL abbreviati nel messaggio, né utilizzare il numero per inviare un messaggio promozionale. A questo scopo, utilizza un numero 10DLC o un codice breve. Quando si utilizza un codice breve o un numero 10DLC, registrare i modelli di messaggio, che possono contenere URL abbreviati ed essere messaggi promozionali. Per ulteriori informazioni sui codici brevi, consulta Codici brevi. Per ulteriori informazioni su 10DLC, consulta 10DLC.

Codici brevi

I codici brevi sono sequenze numeriche più brevi rispetto a un normale numero di telefono. Negli Stati Uniti e in Canada, ad esempio, i numeri di telefono standard (codici lunghi) contengono 11 cifre, mentre i codici brevi ne contengono cinque o sei. Amazon SNS supporta codici brevi dedicati.

Codici brevi dedicati

Se invii un volume elevato di messaggi SMS a destinatari negli Stati Uniti o in Canada, puoi acquistare un codice breve dedicato. A differenza dei codici brevi del pool condiviso, i codici brevi dedicati sono riservati per il tuo uso esclusivo.

Vantaggi

L'utilizzo di un codice breve facile da ricordare può instaurare maggiore fiducia. Se devi inviare informazioni riservate, ad esempio password una tantum, è consigliabile inviarle utilizzando un codice breve in modo che il cliente possa determinare rapidamente se un messaggio proviene effettivamente da te.

Se stai eseguendo una nuova campagna di acquisizione dei clienti, puoi invitare i potenziali clienti a inviare una parola chiave al tuo codice breve (ad esempio, "Invia un SMS con scritto FOOTBALL al 10987 per ricevere notizie e informazioni sul calcio"). I codici brevi sono più facili da ricordare rispetto ai codici lunghi, nonché più semplici da immettere nei dispositivi per i clienti. Riducendo il

livello di difficoltà affrontato dai clienti durante la registrazione per i tuoi programmi di marketing, puoi aumentare l'efficacia delle tue campagne.

Poiché gli di operatori telefonia mobile devono approvare i nuovi codici brevi prima di renderli attivi, è meno probabile che contrassegnino come non sollecitati i messaggi inviati da codici brevi.

Quando utilizzi codici brevi per inviare messaggi SMS, per ogni intervallo di 24 ore puoi inviare un volume di messaggi più elevato rispetto a quello consentito con altri tipi di identità di origine. In altri termini, la quota di invio è molto più elevata. È inoltre possibile inviare un volume molto più elevato di messaggi al secondo. Disponi pertanto di una frequenza di invio molto più elevata.

Svantaggi

L'acquisizione di codici brevi comporta costi aggiuntivi. I tempi di implementazione, inoltre, possono essere lunghi. Negli Stati Uniti, ad esempio, ogni codice breve comporta una commissione di configurazione una tantum di 650 USD e un'ulteriore spesa ricorrente di 995 USD al mese. Possono essere necessarie 8-12 settimane prima che i codici brevi diventino attivi su tutte le reti corriere. Per trovare il costo e il tempo di provisioning per un paese o una regione diversa, completare la procedura descritta in Richiesta di codici brevi dedicati per la messaggistica SMS con Amazon SNS.

Codici lunghi da persona a persona (P2P)

> **Important**
>
> A partire dal 31 agosto 2023, un numero dedicato come un numero 10DLC o un numero verde è necessario per inviare messaggi SMS negli Stati Uniti e nei suoi territori (Porto Rico, Guam, Isole Samoa americane e Isole Vergini US). La tua richiesta di codice lungo verrà rifiutata se utilizzi gli Stati Uniti come sede per queste regioni.

> **Important**
>
> A partire dal 1 giugno 2021, i provider di telecomunicazioni statunitensi non supportano più l'utilizzo di codici lunghi da persona a persona (P2P) per comunicazioni da applicazione-persona (A2P) verso destinazioni statunitensi. Al contrario, è necessario utilizzare un altro tipo di ID di origine per questi messaggi. Per ulteriori informazioni, consulta 10DLC.

I codici lunghi P2P sono numeri di telefono che utilizzano il formato di numero del paese o della regione in cui si trovano i destinatari. I codici lunghi P2P sono detti anche numeri lunghi o numeri di

cellulare virtuali. Negli Stati Uniti e in Canada, ad esempio, i codici lunghi P2P contengono 11 cifre: il numero 1 (prefisso internazionale), un prefisso locale di tre cifre e un numero di telefono di sette cifre.

Per ulteriori informazioni sulla richiesta di codici lunghi P2P, consulta Richiesta di numeri 10DLC, numeri verdi e codici lunghi P2P per la messaggistica SMS con Amazon SNS.

Vantaggi

I codici lunghi P2P dedicati sono riservati solo al tuo account Amazon SNS e non sono condivisi con altri utenti. Quando utilizzi codici lunghi P2P dedicati, all'invio di ogni messaggio puoi specificare quale codice lungo desideri utilizzare. Se invii più messaggi allo stesso cliente, puoi assicurarti che ciascun messaggio risulti inviato dallo stesso numero di telefono. Per questo motivo, i codici lunghi P2P dedicati possono rivelarsi utili per stabilire il marchio o l'identità.

Svantaggi

I codici lunghi P2P non sono supportati per le comunicazioni A2P verso destinazioni US.

Se invii diverse centinaia di messaggi al giorno da un codice lungo P2P dedicato, gli operatori di telefonia mobile potrebbe identificare il tuo numero come un numero che invia messaggi non sollecitati. Se il codice lungo P2P viene contrassegnato, i tuoi messaggi potrebbero non essere recapitati ai destinatari.

I codici lunghi P2P, inoltre, hanno anche un throughput limitato. Le tariffe massime di invio variano in base al paese. Per ulteriori informazioni, contatta AWS Support. Se intendi inviare messaggi SMS in volumi elevati o con una frequenza superiore a un messaggio al secondo, dovresti acquistare un codice breve dedicato.

Alcuni corrieri non consentono di utilizzare codici lunghi P2P per inviare messaggi SMS A2P, anche negli Stati Uniti. Un messaggio SMS A2P è un messaggio che viene inviato al dispositivo mobile di un cliente quando il cliente invia il proprio numero di cellulare a un'applicazione. I messaggi A2P sono conversazioni unidirezionali, ad esempio messaggi di marketing, password una tantum e promemoria di appuntamenti. Se intendi inviare messaggi A2P, dovresti acquistare un codice breve dedicato (se i tuoi clienti si trovano negli Stati Uniti o in Canada) oppure utilizzare un ID mittente (se i destinatari si trovano in un paese o una regione in cui gli ID mittente sono supportati).

Un numero 10DLC viene utilizzato solo per l'invio di messaggi all'interno degli Stati Uniti. Per utilizzare un numero 10DLC è richiesta la registrazione del marchio della tua azienda e della campagna a cui desideri associare il numero. Una volta approvata la registrazione, puoi richiedere

un numero di telefono 10DLC nella pagina SMS and voice (SMS e messaggi vocali) della console Amazon Pinpoint all'indirizzo https://console.aws.amazon.com/pinpoint/. Una volta effettuata la richiesta, le tempistiche per l'approvazione sono di 7-10 giorni. Il numero non può essere utilizzato con altre campagne.

Confronto numero prodotto negli Stati Uniti

Questa tabella mostra il confronto del supporto per i tipi di numeri di telefono negli Stati Uniti.

Funzionalità del prodotto	Codice breve	Numero verde	10DLC	
Formato numerico	5-6 cifre	Numero a 10 cifre	Numero a 10 cifre	
Canale di supporto	SMS	SMS	SMS	
Tipo di traffico SMS	Promozionale e transazionale	Transazionale	Promozionale e transazionale	
Richiede un controllo	Sì	No	Sì	
Tempo stimato di provisioning	12 settimane[1]	15 giorni lavorativi	1 settimana	
Throughput SMS (numero di messaggi SMS al secondo)[2]	100 parti per messaggi al secondo; velocità effettiva più elevata disponibile a un costo aggiuntivo.	3 parti di messaggio al secondo	Varia in base alla tua registrazione 10DLC. Supporta fino a 100 parti di messaggi al secondo.	
Parole chiave richieste	Opt-in, opt-out e HELP	STOP, UNSTOP. Gestiti in rete. Non è possibile	Opt-in, opt-out e HELP	

Funzionalità del prodotto	Codice breve	Numero verde	10DLC	
		modificare l'opt-out e opt-back nei messaggi.		

[1] La stima del provisioning non include il tempo di approvazione.

[2] Per ulteriori informazioni sulle dimensioni massime degli SMS, consulta Pubblicazione su un telefono cellulare.

Richiesta di supporto per la messaggistica SMS con Amazon SNS

Alcune opzioni SMS con Amazon SNS non sono disponibili per il tuo account AWS fino a quando non si contatta AWS Support. Apri un caso nel Center AWS Support per fare una delle richieste seguenti:

- Un aumento della soglia di spesa mensile per gli SMS

  Per impostazione predefinita, la soglia di spesa mensile è di 1 USD. La soglia di spesa determina il volume di messaggi che è possibile inviare con Amazon SNS. Puoi richiedere una soglia che soddisfi il volume mensile previsto di messaggi SMS per il tuo caso d'uso.
- Una mossa dalla sandbox SMS in modo da poter inviare messaggi SMS senza restrizioni. Per ulteriori informazioni, consulta Uscita dalla sandbox SMS.
- Un dedicato Numero di origine
- Un ID mittente dedicato

  Un ID mittente è un ID personalizzato che viene mostrato come mittente nel dispositivo del destinatario. Ad esempio, puoi utilizzare il tuo marchio commerciale per rendere più facilmente riconoscibile l'origine del messaggio. Il supporto per gli ID mittente varia in base al paese o alla Regione. Per ulteriori informazioni, consulta Regioni e paesi supportati.

Quando crei il tuo caso nel Centro AWS Support, includi tutte le informazioni necessarie per il tipo di richiesta che stai inviando. In caso contrario, AWS Support ti contatterà per ottenere queste informazioni prima di continuare. Inviando un caso dettagliato, aumenti le probabilità che venga soddisfatto senza ritardi. Per i dettagli che sono necessari per tipi specifici di richieste SMS, consulta i seguenti argomenti.

Argomenti

Richiesta di codici brevi dedicati per la messaggistica SMS con Amazon SNS

Un codice breve è un numero che puoi utilizzare per l'invio di volumi elevati di messaggi SMS. I codici brevi vengono spesso utilizzati nella messaggistica application-to-person (A2P), nell'autenticazione a due fattori e nel marketing. Un codice breve contiene in genere da tre a sette cifre, a seconda del paese in cui è basato.

Puoi utilizzare codici brevi solo per inviare messaggi a destinatari nello stesso paese in cui il codice breve è basato. Se il tuo caso d'uso richiede l'utilizzo di codici brevi in più paesi, devi richiedere un codice breve separato per ogni paese in cui si trovano i tuoi destinatari.

Per informazioni sui prezzi dei codici brevi, consulta Prezzi di Amazon SNS.

> **Important**
>
> Se non hai mai utilizzato la messaggistica SMS con Amazon SNS, richiedi una soglia di spesa mensile per SMS che soddisfi le esigenze previste del tuo caso d'uso di SMS. Per impostazione predefinita, la soglia di spesa mensile è di 1 USD. Puoi richiedere di aumentare la soglia di spesa nello stesso caso di supporto che include la richiesta di codice breve oppure usare un caso separato. Per ulteriori informazioni, consulta Richiesta di aumento della quota di spesa mensile per l'invio di SMS per Amazon SNS.
> Inoltre, se richiedi un codice breve dedicato per inviare messaggi che contengono informazioni sanitarie protette (PHI), devi identificare questo scopo nella descrizione del caso quando apri un caso di supporto, come descritto di seguito.

Apertura di un caso di supporto per un codice breve Amazon SNS

Apri un caso in AWS Support completando la procedura seguente.

> **Note**
>
> Alcuni dei campi nel modulo di richiesta sono contrassegnate come "facoltativi". Tuttavia, AWS Support richiede tutte le informazioni menzionate nelle seguenti fasi per elaborare la tua richiesta. Se non fornisci tutte le informazioni richieste, possono verificarsi ritardi nell'elaborazione della richiesta.

Per richiedere un codice breve dedicato

1. Vai ad AWS Support Center.
2. Accedi all'AWS Management Console all'indirizzo https://console.aws.amazon.com/.
3. Nel menu Supporto scegliere Centro di supporto.
4. Nella scheda I tuoi casi di supporto, scegli Create caso.
5. Scegli il collegamento Cerchi aumenti del limite di servizio?, quindi completa quanto segue:
   - Per Tipo di limite, scegli Messaggi di testo SNS, quindi completa quanto segue:
   - (Facoltativo) In Fornire un link a un sito o app che invierà messaggi SMS, fornisci un link al sito o al nome dell'applicazione in cui i destinatari acconsentiranno esplicitamente a ricevere i messaggi SMS.
   - In Tipo di messaggi di cui è previsto l'invio, scegli il tipo di messaggio che intendi inviare con codici lunghi:
     - One-Time Password (Password una tantum) - Messaggi che forniscono password che i clienti utilizzano per l'autenticazione a un sito o un'applicazione.
     - Promotional (Promozionale) - Messaggi non critici che promuovono l'azienda o un servizio, ad esempio offerte speciali o annunci.
     - Transactional (Transazionale) - Messaggi informativi importanti che supportano le transazioni con i clienti, come conferme d'ordine o avvisi dell'account. I messaggi transazionali non devono contenere contenuti promozionali o di marketing.
   - (Facoltativo) In Regione AWS da cui si prevede di inviare i messaggi, scegli la regione dalla quale invierai i messaggi SMS.
   - (Facoltativo) In Paesi a cui si prevede di inviare messaggi, inserisci i paesi o le regioni a cui intendi inviare messaggi.
   - (Facoltativo) Nella sezione In che modo i clienti decidono di ricevere i messaggi, fornire una descrizione di come i clienti decidono di ricevere i messaggi.

- (Facoltativo) Nel campo Fornire il modello di messaggio che si intende utilizzare per inviare messaggi ai clienti, includi il modello che utilizzerai.

6. In Requests (Richieste), completa le seguenti sezioni:

   - In Regione, scegli la Regione AWS per la richiesta di codice breve.

     > **Note**
     >
     > La regione è obbligatoria nella sezione Richieste. Anche se hai fornito queste informazioni nella sezione Dettagli del caso, devi includerle anche qui.

   - In Resource Type (Tipo di risorsa) scegliere Dedicated SMS Short Codes (Codici brevi per SMS dedicati).
   - Per Limit (Limite) seleziona l'opzione più simile al tuo caso d'uso.

7. In Nuovo valore limite, scegli il numero di ID mittente che viene richiesto. Questo valore è in genere **1**.
8. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Per le informazioni richieste, consulta le altre sezioni all'interno di Richiesta di supporto per la messaggistica SMS con Amazon SNS.
9. In Descrizione caso, inserisci un riepilogo del caso d'uso e includi il modo in cui i destinatari si registreranno per ricevere i messaggi inviati con il codice breve e fornisci le seguenti informazioni:

   - Informazioni aziendali:
     - Company name (Nome dell'azienda)
     - Indirizzo e-mail dell'azienda
     - Nome e numero di telefono del contatto principale per la richiesta
     - Indirizzo e-mail e numero gratuito per il supporto presso la propria azienda
     - Codice fiscale dell'azienda
     - Nome del prodotto o del servizio

   - Procedura di registrazione utente:
     - Sito Web dell'azienda oppure sito Web in cui si registreranno i clienti per ricevere messaggi dal codice breve.

- Come possono registrarsi gli utenti per ricevere messaggi dal codice breve. Specificare una o più delle seguenti opzioni:
  - **`Text messages`**
  - **`Website`**
  - **`Mobile app`**
  - **`Other`** in questo caso, fornire una spiegazione.
- Il testo associato all'opzione di registrarsi per i messaggi su sito Web, applicazione o altrove.
- La sequenza di messaggi che si prevede di utilizzare per il doppio consenso. Specificare tutte le informazioni seguenti:
  1. Il messaggio SMS che si prevede di inviare quando un utente si registra. Questo messaggio richiede l'autorizzazione dell'utente per i messaggi ricorrenti. Ad esempio: ExampleCorp: Reply YES to receive account transaction alerts. È possibile che vengano applicate tariffe specifiche per messaggi e dati.
  2. La risposta di consenso che si dovrà ricevere dall'utente. In genere è una parola chiave, ad esempio YES.
  3. Il messaggio di conferma che si desidera inviare quando i clienti inviano questa parola chiave al codice breve. Ad esempio: You are now registered for account alerts from ExampleCorp. È possibile che vengano applicate tariffe specifiche per messaggi e dati. Digitare STOP per annullare o HELP per informazioni.

- Lo scopo dei messaggi:
  - Lo scopo dei messaggi che si prevede di inviare con il codice breve. Specificare una delle seguenti opzioni:
    - **`Promotions and marketing`**
    - **`Location-based services`**
    - **`Notifications`**
    - **`Information on demand`**
    - **`Group chat`**
    - **`Two-factor authentication (2FA)`**
    - **`Polling and surveys`**
    - **`Sweepstakes or contests`**
    - **`Other`** in questo caso, fornire una spiegazione.

  - Indicare se si prevede di utilizzare il codice breve per inviare messaggi promozionali o di marketing per un business diverso dal proprio.
  - Indica se prevedi di utilizzare il codice breve per inviare messaggi contenenti informazioni sanitarie protette (PHI), come definito dalla legge HIPAA (Health Insurance Portability and Accountability Act) e dalla legislazione e dai regolamenti associati.

- Contenuto del messaggio:
  - Il messaggio che si prevede di inviare quando i clienti danno il consenso a ricevere i messaggi specificando una parola chiave stabilita. Prestare attenzione quando si specifica questa parola chiave e il messaggio: potrebbero essere necessarie diverse settimane per modificare questo messaggio. Quando viene creato il codice abbreviato, la parola chiave e il messaggio vengono registrati con gli operatori di telefonia mobile nel paese in cui si utilizza il codice breve. Il messaggio potrebbe essere simile all'esempio seguente: Welcome to *ProductName* alerts! Msg&data rates apply.*2* MSGS per month. Rispondere HELP per informazioni, STOP per annullare.
  - La risposta che si desidera inviare quando i clienti rispondono ai messaggi con la parola chiave HELP. Questo messaggio deve includere le informazioni di contatto del supporto clienti. Ad esempio: *ProductName* Alerts: Help at *example.com/help* or *(800) 555-0199.*. Msg&data rates apply. *2* msgs per month. Rispondere STOP per annullare.
  - La risposta che si desidera inviare quando i clienti rispondono ai messaggi con la parola chiave STOP. Questo messaggio deve confermare che l'utente non ricevere più messaggi da te. Ad esempio: You are unsubscribed from *ProductName* Alerts. No more messages will be sent. Reply HELP for help or *(800) 555-0199*. (Hai annullato la sottoscrizione ad Avvisi NomeProdotto. Non riceverai più messaggi da questo account. Rispondi HELP per info o chiama il numero (800) 555-0199.)
  - Il testo che si prevede di inviare come promemoria per ricordare periodicamente all'utente che ha effettuato la sottoscrizione ai messaggi. Ad esempio: Reminder: You're subscribed to account alerts from ExampleCorp. È possibile che vengano applicate tariffe specifiche per messaggi e dati. Digitare STOP per annullare o HELP per informazioni.
  - Un esempio per ciascun tipo di messaggio che si prevede di inviare utilizzando il codice breve. Fornire almeno tre esempi. Se si prevede di inviare più di tre tipi di messaggi, fornire esempi per tutti.

> **Important**
>
> Gli operatori di telefonia mobile ci richiedono di fornire tutte le informazioni elencate in precedenza per effettuare il provisioning di codici brevi. Non siamo in grado di elaborare la richiesta finché non vengono fornite tutte queste informazioni.

10. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Se si includono più richieste, specificare le informazioni necessarie per ciascuna. Per le informazioni richieste, consulta le altre sezioni all'interno di Richiesta di supporto per la messaggistica SMS con Amazon SNS.
11. In Opzioni di contatto, per Lingua di contatto preferita, scegli la lingua in cui desideri ricevere le comunicazioni per questo caso.
12. Al termine, scegli Submit (Invia).

Una volta ricevuta la tua richiesta, ti forniamo una prima risposta iniziale entro 24 ore. Potremmo contattarti per richiedere ulteriori informazioni. Se siamo in grado di fornirti un codice breve, ti invieremo informazioni sui costi associati alla concessione di un codice breve nel paese o nella regione che hai specificato nella richiesta. Ti forniremo inoltre una stima del tempo richiesto per effettuare il provisioning di un codice breve nel tuo paese o regione. In genere sono necessarie alcune settimane per effettuare il provisioning di un codice breve, anche se questo ritardo può essere molto più breve o molto più lungo a seconda del paese o della regione in cui il codice breve è basato.

> **Note**
>
> I costi associati all'utilizzo dei codici brevi vengono applicati immediatamente dopo l'inizializzazione della richiesta agli operatori. L'utente è responsabile del pagamento di queste tariffe anche nel caso in cui l'assegnazione del codice breve non sia stata ancora completata.

Per evitare che i nostri sistemi vengano utilizzati per l'invio di contenuti indesiderati o dannosi, ogni richiesta dovrà essere analizzata attentamente da parte nostra. Potremmo non essere in grado di gestire la tua richiesta se il caso d'uso specifico non è conforme con le nostre policy.

Fasi successive

È stato registrato un codice breve con gli operatori wireless e sono state esaminate le impostazioni nella console Amazon SNS. Ora puoi utilizzare Amazon SNS per inviare messaggi SMS con il codice breve come numero di origine.

Richiesta di numeri 10DLC, numeri verdi e codici lunghi P2P per la messaggistica SMS con Amazon SNS

> **Important**
>
> A partire dal 1° giugno 2021, i provider di telecomunicazioni statunitensi non supportano più l'uso di codici lunghi person-to-person (P2P) per le comunicazioni application-to-person (A2P) verso destinazioni statunitensi. Al contrario, è necessario utilizzare un altro tipo di ID di origine per questi messaggi. Per ulteriori informazioni, consulta 10DLC.

Per richiedere Numeri 10DLC, Numeri verdi, e Codici lunghi P2P utilizza la console Amazon Pinpoint. Per istruzioni dettagliate, consulta Richiesta di un numero nella Guida per l'utente di Amazon Pinpoint.

> **Important**
>
> Gli operatori di telefonia mobile statunitensi hanno recentemente modificato le loro normative e richiederanno che tutte le aziende interessate ad attivare un numero verde (TFN) completino un processo di registrazione presso un organo di regolamentazione entro il 30 settembre 2022. Per informazioni sulla registrazione di un numero verde, consulta Registrazione del numero verde.
>
> Se hai acquistato un numero verde entro il 30 settembre 2022, lo stato sarà Active (Attivo) fino al 1 ottobre 2022, a meno che tu non abbia completato la registrazione correttamente, dopo la quale lo stato diventa Completed (Completato). In caso contrario, la richiesta verrà inserita nello stato Pending (In attesa) e non sarai in grado di inviare messaggi con quel numero fino a quando non avrai effettuato la registrazione e ricevuto conferma della stessa, oppure finché la registrazione non è impostata su Active (Attiva).
>
> Per la registrazione possono essere necessari fino a 15 giorni lavorativi.

Per registrare la tua azienda e la tua campagna con 10 DLC, apri la console Amazon Pinpoint nella regione Stati Uniti orientali (Virginia settentrionale). Invece di richiedere un numero di 10 DLC, utilizza la console AWS Service Quotas per creare un caso di aumento del limite di servizio mentre richiedi il numero di 10 DLC per quella regione. Per informazioni sulle regioni in cui Amazon Pinpoint è disponibile, consultare Amazon Pinpoint endpoints and quotas (Endpoint e quote di Amazon Pinpoint) nei Riferimenti generali di AWS.

> **Note**
>
> Se non hai mai utilizzato la messaggistica SMS con Amazon SNS, devi anche richiedere una soglia di spesa mensile per SMS che soddisfi le esigenze previste del tuo caso d'uso di SMS. Per impostazione predefinita, la soglia di spesa mensile è di 1 USD. Per ulteriori informazioni, consulta Richiesta di aumento della quota di spesa mensile per l'invio di SMS per Amazon SNS.

Richiesta di ID mittente per la messaggistica SMS con Amazon SNS

> **Important**
>
> Se non hai mai utilizzato la messaggistica SMS con Amazon SNS, richiedi una soglia di spesa mensile per SMS che soddisfi le esigenze previste del tuo caso d'uso di SMS. Per impostazione predefinita, la soglia di spesa mensile è di 1 USD. Puoi richiedere di aumentare la soglia di spesa nello stesso caso di supporto contenente la richiesta di un ID mittente In alternativa, se preferisci, è possibile aprire un caso separato. Per ulteriori informazioni, consulta Richiesta di aumento della quota di spesa mensile per l'invio di SMS per Amazon SNS.

Nella messaggistica SMS, un ID mittente è un nome che appare come mittente del messaggio sui dispositivi dei destinatari. Gli ID mittente sono un modo utile per identificarti con i destinatari dei tuoi messaggi.

Il supporto per gli ID mittente varia in base al paese . Ad esempio, gli operatori negli Stati Uniti non supportano gli ID mittente, mentre gli operatori in India richiedono che i mittenti utilizzino gli ID mittente. Per un elenco completo dei paesi che supportano gli ID mittente, consulta Regioni e paesi supportati.

> **Important**
>
> Alcuni paesi richiedono la registrazione degli ID mittente prima di utilizzarli per l'invio di messaggi. A seconda del paese, questa procedura di registrazione potrebbe richiedere diverse settimane. I paesi che richiedono ID mittente pre-registrati sono indicati nella tabella della pagina Paesi supportati.
> È possibile utilizzare e registrare lo stesso ID mittente in più account AWS per i messaggi SMS. Se disponi di supporto aziendale e stai registrando più modelli su più paesi o regioni, segui le istruzioni riportate di seguito e collabora con il tuo Technical Account Manager per assicurarti che la tua esperienza formativa sia coordinata.

Se invii messaggi a destinatari in un paese in cui sono supportati gli ID mittente e quel paese non richiede la registrazione dell'ID mittente, non devi eseguire ulteriori operazioni. Puoi iniziare immediatamente a inviare messaggi che includono valori di ID mittente.

Devi completare le procedure in questa pagina solo se prevedi di inviare messaggi a un paese in cui è richiesta la registrazione degli ID mittente.

Fase 1: apertura di un caso per SMS Amazon SNS

Se prevedi di inviare messaggi a destinatari in un paese in cui sono necessari gli ID mittente, puoi richiedere un ID mittente creando un nuovo caso nel Centro di supporto AWS.

> **Note**
>
> Se prevedi di inviare messaggi a destinatari in un paese in cui gli ID mittente sono consentiti ma non necessari, non è necessario aprire un caso nel Centro di supporto. Puoi iniziare a inviare messaggi che utilizzano gli ID mittente immediatamente.

Per richiedere un ID mittente

1. Accedi all'AWS Management Console all'indirizzo https://console.aws.amazon.com/.
2. Nel menu Supporto scegliere Centro di supporto.
3. Nel riquadro I tuoi casi di supporto, scegli Crea caso.
4. Scegli il collegamento Cerchi aumenti del limite di servizio?, quindi completa quanto segue:

- In Tipo di limite scegli SMS Pinpoint.
- (Facoltativo) In Fornire un link a un sito o app che invierà messaggi SMS, identifica il sito Web o l'applicazione in cui i destinatari acconsentiranno esplicitamente a ricevere i messaggi SMS.
- In Tipo di messaggi di cui è previsto l'invio, scegli il tipo di messaggio che intendi inviare con codici lunghi:
  - One-Time Password (Password una tantum) - Messaggi che forniscono password che i clienti utilizzano per l'autenticazione a un sito o un'applicazione.
  - Promotional (Promozionale) - Messaggi non critici che promuovono l'azienda o un servizio, ad esempio offerte speciali o annunci.
  - Transactional (Transazionale) - Messaggi informativi importanti che supportano le transazioni con i clienti, come conferme d'ordine o avvisi dell'account. I messaggi transazionali non devono contenere contenuti promozionali o di marketing.
- (Facoltativo) In Regione AWS da cui si prevede di inviare i messaggi, scegli la Regione AWS dalla quale invierai i messaggi SMS.
- (Facoltativo) In Paesi a cui si prevede di inviare messaggi, specifica i paesi in cui si desidera registrare un ID mittente. Il supporto per gli ID mittente e i requisiti per la relativa registrazione variano a seconda del paese. Per ulteriori informazioni, consulta Regioni e paesi supportati.

  Se l'elenco dei paesi supera il numero di caratteri consentiti per questa casella di testo, è possibile elencare i paesi nella sezione Descrizione caso.
- (Facoltativo) Nella sezione In che modo i clienti decidono di ricevere i messaggi, fornire una descrizione di come i clienti decidono di ricevere i messaggi.
- (Facoltativo) Nel campo Fornire il modello di messaggio che si intende utilizzare per inviare messaggi ai clienti, includi tutti i modelli di messaggio che utilizzerai.

5. In Requests (Richieste), completa le seguenti sezioni:

   - Per Regione, scegli la Regione AWS dell'ID mittente.

     > **Note**
     >
     > La regione è obbligatoria nella sezione Richieste. Anche se hai fornito queste informazioni nella sezione Dettagli del caso, devi includerle anche qui.

   - In Resource Type (Tipo di risorsa) scegliere General Limits (Limiti generali).

- In Limite scegliere Accesso produzione SMS.

6. In Nuovo valore limite, scegli il numero di ID mittente che viene richiesto. Questo valore è in genere **1**.
7. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Per le informazioni richieste, consulta le altre sezioni all'interno di Richiesta di supporto per la messaggistica SMS con Amazon SNS.
8. In Case description (Descrizione caso), per Use case description (Descrizione del caso d'uso), specificare i dettagli seguenti:

   - L'ID mittente che desideri registrare.
   - Il modello che prevedi di utilizzare per i tuoi messaggi SMS.
   - Il numero di messaggi che intendi inviare a ciascun destinatario al mese.
   - Informazioni su come i clienti scelgono di ricevere messaggi da te.
   - Il nome della tua azienda o organizzazione.
   - L'indirizzo associato alla tua azienda o organizzazione.
   - Il paese in cui si trova la tua azienda o organizzazione.
   - Un numero di telefono per la tua azienda o organizzazione.
   - L'URL del sito Web per la tua azienda o organizzazione.
9. In Opzioni di contatto, per Lingua di contatto preferita, scegli se le comunicazioni ricevute devono essere in inglese o in giapponese.
10. Al termine, scegli Submit (Invia).

Una volta ricevuta la tua richiesta, ti forniamo una prima risposta iniziale entro 24 ore. Potremmo contattarti per richiedere ulteriori informazioni. Se saremo in grado di fornirti un ID mittente, ti invieremo una stima della quantità di tempo necessaria per effettuare il provisioning.

Per evitare che i nostri sistemi vengano utilizzati per l'invio di contenuti indesiderati o dannosi, ogni richiesta dovrà essere analizzata attentamente da parte nostra. Potremmo non essere in grado di gestire la tua richiesta se il caso d'uso specifico non è conforme con le nostre policy.

Fase 2: aggiornamento delle impostazioni relative agli SMS nella console Amazon SNS

Una volta completato il processo per ottenere l'ID mittente, rispondiamo al tuo caso. Quando si riceve questa notifica, completare la procedura descritta in questa sezione per configurare Amazon SNS per l'utilizzo dell'ID mittente come ID mittente predefinito per tutti i messaggi inviati tramite l'account. In

alternativa, è possibile scegliere di specificare l'ID mittente da utilizzare durante la pubblicazione del messaggio.

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, seleziona Dispositivi mobili, Messaggi di testo (SMS).
3. Nella sezione Preferenze per i messaggi di testo, scegli Modifica.
4. Nella sezione Dettagli, nel campo ID mittente predefinito, immetti l'ID mittente fornito da utilizzare come predefinito per tutti i messaggi provenienti dall'account.
5. Al termine, scegliere Save changes (Salva modifiche).

Fasi successive

Dopo aver registrato un ID mittente e aver aggiornato le impostazioni nella console Amazon SNS. Si può utilizzare Amazon SNS per inviare messaggi SMS con l'ID mittente. L'ID mittente verrà visualizzato sui dispositivi dei destinatari di SMS nei paesi supportati come mittente del messaggio. Se durante la pubblicazione dei messaggi viene utilizzato un ID mittente diverso, verrà ignorato l'ID predefinito configurato qui.

Richiesta di aumento della quota di spesa mensile per l'invio di SMS per Amazon SNS

Amazon SNS fornisce quote di spesa che ti aiutano a gestire il costo massimo mensile sostenuto per l'invio di SMS utilizzando il tuo account. La quota di spesa limita i rischi in caso di attacchi dannosi e impedisce all'applicazione a monte di inviare più messaggi del previsto. Puoi configurare Amazon SNS in modo tale da interrompere la pubblicazione di messaggi SMS qualora il sistema determini che l'invio di un messaggio SMS comporta un costo che supera la quota di spesa per il mese corrente.

Per garantire che la tua attività rimanga operativa, ti consigliamo di richiedere una quota di spesa sufficientemente elevata per supportare i carichi di lavoro di produzione. Per ulteriori informazioni, consulta Fase 1: apertura di un caso per SMS Amazon SNS. Una volta ricevuta la quota, puoi gestire il rischio applicando la quota completa o un valore inferiore, come descritto nella sezione Fase 2: aggiornamento delle impostazioni relative agli SMS. Applicando un valore più basso, puoi controllare la spesa mensile con la possibilità di aumentarla in base alle esigenze.

> **Important**
>
> Poiché è un sistema distribuito, Amazon SNS blocca l'invio di SMS alcuni minuti dopo il superamento della quota di spesa. Se continui a inviare SMS durante tale periodo, potresti incorrere in costi che superano la tua quota di spesa.

La quota di spesa è impostata su 1 USD al mese per tutti i nuovi account. Questa quota è pensata per consentirti di testare le funzionalità di invio di messaggi di Amazon SNS. Per richiedere un aumento della quota di spesa per gli SMS per l'account, apri un caso di aumento delle quote in AWS Support Center.

Fase 1: apertura di un caso per SMS Amazon SNS

È possibile richiedere l'aumento della quota di spesa mensile aprendo un caso di aumento delle quote in AWS Support Center.

> **Note**
>
> Alcuni dei campi nel modulo di richiesta sono contrassegnate come "facoltativi". Tuttavia, AWS Support richiede tutte le informazioni menzionate nelle seguenti fasi per elaborare la tua richiesta. Se non fornisci tutte le informazioni richieste, possono verificarsi ritardi nell'elaborazione della richiesta.

1. Accedi all'AWS Management Console all'indirizzo https://console.aws.amazon.com/.
2. Nel menu Supporto scegliere Centro di supporto.
3. Nella scheda I tuoi casi di supporto, scegli Create caso.
4. Scegli il collegamento Cerchi aumenti del limite di servizio?, quindi completa quanto segue:
   - Per Tipo di limite, scegli Messaggi di testo SNS.
   - (Facoltativo) In Fornisci un link al sito o all'app che invierà messaggi SMS, fornisci informazioni relative al sito Web, all'applicazione o al servizio che invierà messaggi SMS.
   - (Facoltativo) In Tipo di messaggi di cui è previsto l'invio, scegli il tipo di messaggio che intendi inviare con codici lunghi:
     - One-Time Password (Password una tantum) - Messaggi che forniscono password che i clienti utilizzano per l'autenticazione a un sito o un'applicazione.

- Promotional (Promozionale) - Messaggi non critici che promuovono l'azienda o un servizio, ad esempio offerte speciali o annunci.
- Transactional (Transazionale) - Messaggi informativi importanti che supportano le transazioni con i clienti, come conferme d'ordine o avvisi dell'account. I messaggi transazionali non devono contenere contenuti promozionali o di marketing.

- (Facoltativo) In Regione AWS da cui si prevede di inviare i messaggi, scegli la regione dalla quale invierai i messaggi SMS.
- (Facoltativo) In Paesi a cui si prevede di inviare messaggi, immetti il paese o la regione in cui desideri acquistare codici brevi.
- (Facoltativo) Nella sezione In che modo i clienti decidono di ricevere messaggi dall'utente, fornisci dettagli sul processo di consenso esplicito.
- (Facoltativo) Nel campo Fornire il modello di messaggio che si intende utilizzare per inviare messaggi ai clienti, includi il modello che utilizzerai.

5. In Requests (Richieste), completa le seguenti sezioni:

   - Per la Regione, scegli la regione da cui invierai i messaggi.

     > **Note**
     >
     > La regione è obbligatoria nella sezione Richieste. Anche se hai fornito queste informazioni nella sezione Dettagli del caso, devi includerle anche qui.

   - In Resource Type (Tipo di risorsa) scegliere General Limits (Limiti generali).
   - In Limit (Limite) scegliere Account Spend Threshold Increase (Aumento soglia di spesa account).

6. In Nuovo valore limite, immetti l'importo massimo (in USD) che puoi spendere in messaggi SMS per ogni mese di calendario.
7. In Case description (Descrizione caso), per Use case description (Descrizione del caso d'uso), specificare i dettagli seguenti:

   - Il sito Web o l'app dell'azienda o del servizio che invia messaggi SMS
   - Servizio fornito dal sito Web o dall'app e contributo dei messaggi SMS a tale servizio
   - In che modo gli utenti si registrano per ricevere volontariamente i messaggi SMS sul sito Web, sull'app o altra posizione.

Se la quota di spesa richiesta, ovvero il valore specificato per New quota value (Nuovo valore quota), è superiore a 10.000 USD, fornire i seguenti dettagli aggiuntivi per ciascun paese a cui si inviano messaggi:

- Se si utilizza un ID mittente o un codice breve. Se si utilizza un ID mittente, fornire:
  - L'ID del mittente.
  - Se l'ID del mittente è registrato con operatori wireless nel paese.
- Il numero massimo di transazioni al secondo (TPS) previste per l'invio di messaggi.
- La dimensione media dei messaggi.
- Il modello dei messaggi che vengono inviati al paese specifico.
- (Facoltativo) eventuali esigenze di codifica dei caratteri.

8. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Se si includono più richieste, specificare le informazioni necessarie per ciascuna. Per le informazioni richieste, consulta le altre sezioni all'interno di Richiesta di supporto per la messaggistica SMS con Amazon SNS.
9. In Opzioni di contatto, per Lingua di contatto preferita, scegli la lingua in cui desideri ricevere le comunicazioni per questo caso.
10. Al termine, scegli Submit (Invia).

Il team AWS Support fornisce una risposta iniziale alla richiesta entro 24 ore.

Per evitare che i nostri sistemi vengano utilizzati per l'invio di contenuti indesiderati o dannosi, ogni richiesta verrà analizzata attentamente da parte nostra. In seguito a questa valutazione, saremo in grado di gestire la tua richiesta durante le prime 24 ore. Tuttavia, se la risoluzione richiede ulteriori informazioni da parte tua, i tempi di gestione della richiesta potranno essere più lunghi.

Se il caso d'uso specifico non è conforme con le nostre policy, potremmo non essere in grado di gestire la tua richiesta s

Fase 2: aggiornamento delle impostazioni relative agli SMS nella console Amazon SNS

Dopo aver ricevuto la notifica relativa all'aumento della quota di spesa mensile, devi modificare la quota di spesa per l'account nella console Amazon SNS.

> **Important**
>
> È necessario completare i seguenti passaggi o il limite di spesa per gli SMS non verrà aumentato.

Per modificare la quota di spesa nella console

1. Accedi alla console Amazon SNS.
2. Apri il menu di navigazione a sinistra, espandi Dispositivi mobili, quindi scegli Messaggi di testo (SMS).
3. Nella pagina Mobile text messaging (SMS) (Messaggi di testo mobili (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).
4. Nella pagina Modifica le preferenze per i messaggi di testo, nella sezione Dettagli, inserisci il nuovo limite di spesa SMS nel campo Limite di spesa per account.

   > **Note**
   >
   > Potrebbe essere visualizzato un avviso che indica che il valore immesso è maggiore del limite di spesa predefinito. È possibile ignorare questo avviso.

5. Seleziona Salva modifiche.

   > **Note**
   >
   > Se ricevi un errore di "Parametro non valido", controlla il contatto da Support AWS e conferma di aver inserito il nuovo limite di spesa SMS corretto. Se si verifica ancora un problema, aprire un caso in Support Center AWS.

Impostazione delle preferenze di messaggistica SMS

Utilizza Amazon SNS per specificare le preferenze per i messaggi SMS. Ad esempio, puoi specificare se ottimizzare le consegne per costo o affidabilità, il limite di spesa mensile, la registrazione delle consegne e se eseguire la sottoscrizione a report di utilizzo di SMS giornalieri.

Queste preferenze diventano effettive per ogni SMS che invii dal tuo account, ma alcune possono essere sovrascritte all'invio di un singolo messaggio. Per ulteriori informazioni, consulta Pubblicazione su un telefono cellulare.

Argomenti

- Impostazione delle preferenze per i messaggi SMS utilizzando AWS Management Console
- Impostazione delle preferenze (AWS SDK)

Impostazione delle preferenze per i messaggi SMS utilizzando AWS Management Console

1. Accedi alla console Amazon SNS.
2. Scegliere una regione che supporti la messaggistica SMS.
3. Nel riquadro di navigazione, selezionare Mobile (Dispositivi mobili), Text messaging (SMS) (Messaggi di testo (SMS)).
4. Nella pagina Mobile text messaging (SMS) (Messaggi di testo mobili (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).
5. Nella pagina Modifica le preferenze per i messaggi di testo, nella sezione Dettagli, effettuare queste operazioni:

   a. Per Default message type (Tipo di messaggio predefinito), scegliere una di queste opzioni:

      - Promozionale: messaggi non critici (ad esempio marketing). Amazon SNS ottimizza la consegna dei messaggi per generare il costo più basso.
      - Transazionale: messaggi critici che supportano le transazioni dei clienti, come le password monouso per l'autenticazione a più fattori. Amazon SNS ottimizza la consegna dei messaggi per ottenere la migliore affidabilità.

      Per informazioni sulle tariffe relative a messaggi promozionali e transazionali, consulta la pagina relativa alle tariffe SMS globali.

   b. In Account spend limit (Limite di spesa per account), digitare l'importo massimo in USD da spendere in messaggi SMS ogni mese di calendario.

> **Important**
>
> - Per impostazione predefinita, la quota di spesa è impostata a 1,00 USD. Se si desidera aumentare la quota di servizio, inviare una richiesta.
> - Se l'importo impostato nella console supera la quota di servizio, Amazon SNS interrompe la pubblicazione di messaggi SMS.
> - Poiché Amazon SNS è un sistema distribuito, blocca l'invio di SMS alcuni minuti dopo il superamento della quota di spesa. Se continui a inviare SMS durante tale intervallo, potresti incorrere in costi che superano la tua quota di spesa.

6. (Facoltativo) Per Default sender ID (ID mittente predefinito), immettere un ID personalizzato, ad esempio il marchio aziendale, visualizzato come mittente sul dispositivo di ricezione.

> **Note**
>
> Il supporto per gli ID mittente varia in base al paese.

7. (Facoltativo) Immettere il Amazon S3 bucket name for usage reports (Nome bucket Amazon S3 per i report di utilizzo).

> **Note**
>
> La policy di bucket S3 deve concedere l'autorizzazione in scrittura ad Amazon SNS..

8. Seleziona Save changes (Salva modifiche).

Impostazione delle preferenze (AWS SDK)

Per impostare le tue preferenze SMS utilizzando uno degli AWS SDK, utilizza l'azione in quell'SDK che corrisponde alla `SetSMSAttributes` richiesta nell'API Amazon SNS. Con questa richiesta, assegni valori a differenti attributi SMS, come la quota di spesa mensile e il tipo di SMS predefinito (promozionale o transazionale). Per tutti gli attributi SMS, consulta SetSMSAttributes nella Amazon Simple Notification Service API Reference.

I seguenti esempi di codice mostrano come usare. `SetSMSAttributes`

C++

SDK per C++

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Come utilizzare Amazon SNS per impostare l'attributo DefaultSMType.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Impostazione degli attributi dei messaggi SMS

Nell'esempio `set-sms-attributes` seguente l'ID mittente predefinito per i messaggi SMS viene impostato su `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Questo comando non produce alcun output.

- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nel Riferimento ai comandi AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
        // Transactional – Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for JavaScript .

PHP

SDK per PHP

> **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.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for PHP .

Invio di messaggi SMS

Questa sezione descrive come inviare messaggi SMS.

Argomenti

- Pubblicazione in un argomento
- Pubblicazione su un telefono cellulare

Pubblicazione in un argomento

Puoi inviare un singolo messaggio SMS a più numeri di telefono contemporaneamente sottoscrivendo tali numeri in un argomento Amazon SNS. Un argomento SNS è un canale di comunicazione dove è possibile aggiungere gli abbonati ai quali si vogliono indirizzare determinati messaggi. Un utente riceve tutti i messaggi pubblicati in tale argomento fino a quando l'iscrizione non verrà annullata o l'utente non rinuncerà alla loro ricezione da parte del tuo account AWS.

Argomenti

- Invio di un messaggio SMS in un argomento (console)
- Invio di un messaggio a un argomento (SDK AWS)

Invio di un messaggio SMS in un argomento (console)

Per creare un argomento

Se ancora non disponi di un argomento al quale inviare messaggi SMS, completa i seguenti passaggi.

1. Accedi alla console Amazon SNS.

2. Nel menu della console, scegli un'area AWS che supporti la messaggistica SMS.
3. Nel pannello di navigazione, scegli Topics (Argomenti).
4. Nella pagina Topics (Argomenti), seleziona Create new topic (Crea nuovo argomento).
5. Nella pagina Create topic (Crea argomento), in Details (Dettagli), effettuare le seguenti operazioni:

   a. Per Tipo, scegliere Standard.
   b. In Name (Nome), immettere un nome.
   c. (Facoltativo) In Display name (Nome visualizzato), inserisci un prefisso personalizzato per il messaggio SMS. Quando invii un messaggio all'argomento, Amazon SNS antepone il nome visualizzato a una parentesi angolare destra (**>**) e uno spazio. I nomi visualizzati non sono sensibili alle maiuscole/minuscole, in quanto sarà poi Amazon SNS a convertirli in caratteri maiuscoli. Ad esempio, se il nome visualizzato di un argomento è `MyTopic` e il messaggio è `Hello World!`, il messaggio appare in questo formato:

   ```
   MYTOPIC> Hello World!
   ```

6. Scegliere Create topic (Crea argomento). Il nome dell'argomento e l'Amazon Resource Name (ARN) vengono visualizzati nella pagina Topics (Argomenti)

Per creare una sottoscrizione SMS

Con le sottoscrizioni puoi inviare un messaggio SMS a più destinatari pubblicandolo una volta sola nel tuo argomento.

> **Note**
>
> Quando inizi a utilizzare Amazon SNS per inviare messaggi SMS, l'account AWS si trova nella Sandbox SMS. La sandbox SMS offre un ambiente sicuro per provare le funzionalità Amazon SNS senza rischiare la reputazione del mittente SMS. Quando il tuo account è nella sandbox SMS, puoi utilizzare tutte le caratteristiche di Amazon SNS, ma puoi inviare SMS solo a numeri di telefono di destinazione verificati. Per ulteriori informazioni, consulta Sandbox SMS.

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione scegli Subscriptions (Sottoscrizioni).

3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Create subscription (Crea sottoscrizione), nella sezione Details (Dettagli), eseguire queste operazioni:

   a. Per Topic ARN (ARN argomento), inserisci o seleziona l'Amazon Resource Name (ARN) dell'argomento a cui desideri inviare messaggi SMS.
   b. In Protocol (Protocollo), scegli SMS.
   c. Per Endpoint, inserisci il numero di telefono a cui desideri iscriverti all'argomento.
5. Scegli Create Subscription (Crea sottoscrizione). Le informazioni sull'abbonamento vengono visualizzate nella pagina Subscriptions (Abbonamenti).

   Per aggiungere altri numeri di telefono, ripetere questi passaggi. Puoi anche aggiungere altri tipi di sottoscrizione, ad esempio le e-mail.

Per inviare un messaggio

Quando pubblichi un messaggio in un argomento, Amazon SNS prova a consegnare il messaggio a tutti i numeri di telefono che hanno effettuato la sottoscrizione a un dato argomento.

1. Nella Console Amazon SNS, nella pagina Topics (Argomenti), scegli il nome dell'argomento a cui desideri inviare messaggi SMS.
2. Nella pagina dettagli, seleziona Publish message (Pubblica messaggio).
3. Nella pagina Publish message to topic (Pubblica messaggio nell'argomento), alla sezione Message details (Dettagli messaggio) procedi come indicato di seguito:

   a. Per Subject (Oggetto), lascia il campo vuoto a meno che il tuo argomento non contenga anche sottoscrizioni e-mail e tu non voglia pubblicare il messaggio sulle sottoscrizioni e-mail ed SMS. Amazon SNS utilizza il Subject (Oggetto) che inserisci come oggetto dell'e-mail.
   b. (Facoltativo) Per Time to Live (TTL) (Durata), inserisci un numero di secondi che Amazon SNS deve inviare il tuo messaggio SMS a tutti gli abbonati agli endpoint delle applicazioni mobili.
4. Su Message body (Corpo del messaggio), procedere come segue:

   a. Per Message structure (Struttura dei messaggi), scegliere Identical payload for all delivery protocols (Payload identico per tutti i protocolli di consegna) per inviare lo stesso messaggio a tutti i tipi di protocollo sottoscritti all'argomento. In alternativa, scegliere Custom payload for each delivery protocol (Payload personalizzato per ogni protocollo di consegna) per

personalizzare il messaggio per i sottoscrittori di diversi tipi di protocollo. Ad esempio, è possibile immettere un messaggio predefinito per gli abbonati al numero di telefono e un messaggio personalizzato per i sottoscrittori di posta elettronica.

b. Per Message body to send to the endpoint (Corpo del messaggio da inviare all'endpoint), immettere il messaggio o i messaggi personalizzati per protocollo di recapito.

   Se il tuo argomento ha un nome visualizzato, Amazon SNS lo aggiunge al messaggio, aumentando così la lunghezza del messaggio. La lunghezza del nome visualizzato dipende dal numero di caratteri nel nome più due per la parentesi angolare destra (**>**) e lo spazio aggiunto da Amazon SNS.

   Per informazioni sulle dimensioni massime degli SMS, consulta Pubblicazione su un telefono cellulare.

5. (Facoltativo) Per Message attributes (Attributi di messaggio), aggiungere metadati dei messaggi quali timestamp, firme e ID.
6. Seleziona Publish message (Pubblica messaggio). Amazon SNS invia il messaggio SMS e ne visualizza quindi la conferma.

Invio di un messaggio a un argomento (SDK AWS)

Per utilizzare un SDK AWS, è necessario configurarlo con le proprie credenziali. Per ulteriori informazioni, consulta The shared config and credentials files (File di configurazione e credenziali condivisi) nella Guida per l'utente SDK e strumenti AWS.

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.

Java

SDK per Java 2.x

> **Note**
>
> Ulteriori informazioni 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 su un telefono cellulare

Puoi utilizzare Amazon SNS per inviare messaggi SMS direttamente a un telefono cellulare senza sottoscrivere il numero di telefono a un argomento Amazon SNS.

> **Note**
>
> La sottoscrizione di numeri di telefono a un argomento è utile se desideri inviare un messaggio a più numeri di telefono contemporaneamente. Per istruzioni sulla pubblicazione di un messaggio SMS in un argomento, consulta Pubblicazione in un argomento.

Quando invii un messaggio, puoi controllare se è ottimizzato relativamente al costo e all'affidabilità della consegna. È anche possibile specificare un sender ID or origination number (ID mittente o numero di origine). Se invii il messaggio in modo programmatico utilizzando l'API Amazon SNS o AWS gli SDK, puoi specificare un prezzo massimo per la consegna del messaggio.

Ogni messaggio SMS può contenere fino a 140 byte, mentre la quota di caratteri dipende dallo schema di codifica. Ad esempio, un messaggio SMS può contenere:

- 160 caratteri GSM
- 140 caratteri ASCII
- 70 caratteri UCS-2

Se il messaggio supera la quota della dimensione, Amazon SNS lo invia sotto forma di più messaggi, ciascuno nel rispetto della quota della dimensione indicata. I messaggi non vengono troncati nel mezzo di una parola ma tra una parola e l'altra. Le quota della dimensione totale massima di una singola pubblicazione SMS è di 1600 byte.

Quando invii un SMS, specifichi il numero di telefono utilizzando il formato E.164, una struttura di numerazione standard utilizzata per le telecomunicazioni internazionali. I numeri di telefono che seguono questo formato possono avere un massimo di 15 cifre oltre al prefisso con il segno più (+) e il prefisso del paese. Ad esempio, un numero di telefono negli Stati Uniti nel formato E.164 verrà visualizzato come +1XXX5550100.

Argomenti

- Invio di un messaggio (Console)
- Invio di un messaggio (SDK)AWS

Invio di un messaggio (Console)

1. Accedi alla console Amazon SNS.
2. Nel menu della console, scegli un'area AWS che supporti la messaggistica SMS.
3. Nel riquadro di navigazione seleziona Text messaging (SMS) (SMS).
4. Nella pagina Messaggi di testo per dispositivi mobili (SMS) scegli Pubblicazione di testo.
5. Nella pagina Pubblicazione di messaggi SMS, per Tipo di messaggio scegliere una delle opzioni seguenti:

   - Promotional (Promozionali) - Messaggi non critici, come i messaggi di marketing.
   - Transactional (Transazionali) - Messaggi critici che supportano le transazioni dei clienti, come le password monouso per l'autenticazione a più fattori.

   > **Note**
   >
   > Questa impostazione a livello di messaggio sostituisce il tipo di messaggio predefinito a livello di account. È possibile impostare un tipo di messaggio predefinito a livello di account dalla sezione Preferenze di messaggistica della pagina Messaggi di testo per dispositivi mobili (SMS).

   Per informazioni sulle tariffe relative a messaggi promozionali e transazionali, consulta la pagina relativa alle tariffe SMS globali.
6. In Number (Numero), inserisci il numero di telefono al quale vuoi inviare il messaggio.
7. In Message (Messaggio), inserisci il testo da inviare.
8. (Facoltativo) Su Identità di origine, specificare come identificarsi ai destinatari:

   - Per specificare l'ID mittente, digita un ID personalizzato contenente da 3 a 11 caratteri alfanumerici, tra cui almeno una lettera e nessuno spazio. L'ID mittente viene visualizzato come mittente del messaggio sul dispositivo ricevente. Ad esempio, puoi utilizzare il tuo marchio commerciale per rendere più facilmente riconoscibile l'origine del messaggio.

     Il supporto per gli ID mittente varia in base al paese e/o alla regione. Ad esempio, i messaggi consegnati a numeri di telefono statunitensi non visualizzeranno l'ID mittente. Per i paesi e le regioni che supportano gli ID mittente, consulta Regioni e paesi supportati.

Se non specifichi un ID mittente, una delle seguenti identità verrà visualizzata come identità di origine:

- Nei paesi che supportano codici lunghi, verrà visualizzato il codice lungo.
- Nei paesi in cui sono supportati solo gli ID mittente, verrà visualizzato NOTICE (PREAVVISO).

Tale ID mittente a livello di messaggio sovrascrive l'ID mittente predefinito, che è stato impostato sulla pagina Text messaging preferences (Preferenze messaggi di testo).

- Per specificare un Numero di origine, inserisci una stringa di 5-14 numeri da visualizzare come numero di telefono del mittente sul dispositivo del ricevitore. Questa stringa deve corrispondere a un numero di origine configurato nel Account AWS per il paese di destinazione. Il numero di origine può essere un numero da 10 DLC, un numero verde, un codice lungo o codici brevi. person-to-person Per ulteriori informazioni, consulta Identità di origine per i messaggi SMS.

  Se non specifichi un numero di origine, Amazon SNS seleziona un numero di origine da utilizzare per il messaggio di testo SMS, in base alla configurazione. Account AWS

9. Se invii messaggi SMS ai destinatari in India, espandi Attributi specifici del paese e specifica gli attributi seguenti:

   - ID entità — L'ID entità o l'ID entità principale (PE) per l'invio di messaggi SMS ai destinatari in India. Questo ID è una stringa univoca di 1-50 caratteri fornita dalla Telecom Regulatory Authority of India (TRAI) per identificare l'entità registrata presso il TRAI.
   - ID modello — L'ID modello per l'invio di messaggi SMS ai destinatari in India. Questo ID è una stringa univoca fornita da TRAI di 1-50 caratteri che identificherà il modello registrato con TRAI. L'ID modello deve essere associato all'ID mittente specificato per il messaggio.

   Per ulteriori informazioni sull'invio di messaggi SMS ai destinatari in India, consulta Requisiti per la registrazione dell'ID mittente per l'India.

10. Seleziona Publish message (Pubblica messaggio).

> **Tip**
>
> Per inviare messaggi SMS da un numero di origine, puoi anche scegliere Numeri di origine nel pannello di navigazione della console Amazon SNS. Scegliere un numero di origine che includa SMS nella colonna Capabilities (Funzionalità), quindi scegliere Pubblicazione di testo.

Invio di un messaggio (SDK)AWS

Per inviare un messaggio SMS utilizzando uno degli AWS SDK, utilizza l'operazione API in quell'SDK che corrisponde alla `Publish` richiesta nell'API Amazon SNS. Con questa richiesta, puoi inviare un messaggio SMS direttamente a un numero di telefono. Puoi anche utilizzare il parametro `MessageAttributes` per impostare i valori per i seguenti nomi attributo:

`AWS.SNS.SMS.SenderID`

Un ID personalizzato che contiene da 3 a 11 caratteri alfanumerici o caratteri trattino (-), tra cui almeno una lettera e nessuno spazio. L'ID mittente viene visualizzato come mittente del messaggio sul dispositivo ricevente. Ad esempio, puoi utilizzare il tuo marchio commerciale per aiutare a rendere più facilmente riconoscibile l'origine del messaggio.

Il supporto per gli ID mittente varia in base al paese o alla Regione. Nei messaggi recapitati a numeri di telefono degli Stati Uniti, ad esempio, non viene visualizzato l'ID mittente. Per un elenco di paesi e regioni che supportano gli ID mittente, consulta Regioni e paesi supportati.

Se non specifichi un ID mittente, il messaggio visualizzerà un codice lungo come ID mittente nei paesi o nelle regioni supportati. Per i paesi o le regioni che richiedono un ID mittente in forma alfabetica, il messaggio visualizza NOTICE (PREAVVISO) come ID mittente.

Questo attributo a livello di messaggio sovrascrive l'attributo a livello di account `DefaultSenderID`, che può essere impostato tramite la richiesta `SetSMSAttributes`.

`AWS.MM.SMS.OriginationNumber`

Una stringa personalizzata di 5-14 numeri, che può includere un segno + iniziale opzionale (+). Questa stringa di numeri viene visualizzata come numero di telefono del mittente sul dispositivo ricevente. La stringa deve corrispondere a un numero di origine configurato nel tuo AWS account per il paese di destinazione. Il numero di origine può essere un numero da 10 DLC, un numero verde, un codice lungo person-to-person (P2P) o un codice breve. Per ulteriori informazioni, consulta Numeri di origine.

Se non specifichi un numero di origine, Amazon SNS sceglie un numero di origine in base alla configurazione del tuo account. AWS

`AWS.SNS.SMS.MaxPrice`

L'importo massimo in dollari che sei disposto a spendere per inviare il messaggio SMS. Amazon SNS non invierà il messaggio nel caso in cui il costo dovesse superare il prezzo massimo.

Questo attributo non ha effetto se i costi month-to-date degli SMS hanno già superato la quota impostata per l'attributo. `MonthlySpendLimit` È possibile impostare `MonthlySpendLimit` utilizzando l'attributo `SetSMSAttributes`.

Quando invii il messaggio a un argomento Amazon SNS, a ogni messaggio inviato viene applicato il prezzo massimo a ognuno dei numeri di telefono sottoscritti a un dato argomento.

`AWS.SNS.SMS.SMSType`

Tipo di messaggio che intendi inviare:

- `Promotional` (impostazione predefinita) - Messaggi non critici, come i messaggi di marketing.
- `Transactional` – Messaggi critici che supportano le transazioni dei clienti, come i passcode monouso per l'autenticazione a più fattori.

Questo attributo a livello di messaggio sovrascrive l'attributo a livello di account `DefaultSMSType`, che può essere impostato tramite la richiesta `SetSMSAttributes`.

`AWS.MM.SMS.EntityId`

Questo attributo è necessario solo per l'invio di messaggi SMS ai destinatari in India.

ID entità o ID entità principale (PE) per l'invio di messaggi SMS ai destinatari in India. Questo ID è una stringa univoca di 1-50 caratteri fornita dalla Telecom Regulatory Authority of India (TRAI) per identificare l'entità registrata presso il TRAI.

`AWS.MM.SMS.TemplateId`

Questo attributo è necessario solo per l'invio di messaggi SMS ai destinatari in India.

Questo è il modello per inviare messaggi SMS ai destinatari in India. Questo ID è una stringa univoca fornita da TRAI di 1-50 caratteri che identificherà il modello registrato con il TRAI. L'ID modello deve essere associato all'ID mittente specificato per il messaggio.

Invio di un messaggio

Negli esempi di codice seguenti viene illustrato come pubblicare messaggi SMS utilizzando Amazon SNS.

.NET

AWS SDK for .NET

> **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.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
```

```
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for .NET .

C++

SDK per C++

> **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.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for C++ .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 .

Kotlin

SDK per Kotlin

> **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.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API di SDK AWS per Kotlin.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for PHP .

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
 must be
                             in E.164 format. For example, a United States phone
```

```
                        number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per l'API SDK for Python (Boto3)AWS .

Monitoraggio dell'attività SMS

Mediante il monitoraggio della tua attività SMS, puoi tenere traccia di numeri di telefono di destinazione, consegne riuscite o non riuscite, motivi della mancate consegne, costi e altre informazioni. Amazon SNS consente di riepilogare le statistiche nella console, inviare informazioni a d Amazon CloudWatch nonché inviare report di utilizzo di SMS giornalieri a un bucket Amazon S3 che hai specificato.

Argomenti

- Visualizzazione delle statistiche di consegna SMS
- Visualizzazione dei parametri e dei log Amazon CloudWatch per le consegne di SMS
- Visualizzazione dei report di utilizzo di SMS giornalieri

Visualizzazione delle statistiche di consegna SMS

Puoi utilizzare la console Amazon SNS per visualizzare le statistiche relative alla consegna dei tuoi SMS più recenti

1. Accedi alla console Amazon SNS.
2. Nel menu della console, imposta lo strumento di selezione della regione su una regione che supporti la messaggistica SMS.
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Sulla pagina Text messaging (SMS) (Preferenze messaggistica (SMS)) nella sezione Account stats (Statistiche account), visualizza i grafici delle consegne degli SMS promozionali e di vendita. Ogni tabella mostra i seguenti dati relativi ai 15 giorni precedenti:
   - Tasso di consegna (percentuale di successo)
   - Inviati (numero di tentativi di recapito)
   - Falliti (numero di mancate consegne)

Su questa pagina, puoi anche selezionare il pulsante Usage (Utilizzo) per andare al bucket Amazon S3 dove vengono salvati i report di utilizzo giornalieri. Per ulteriori informazioni, consulta . Visualizzazione dei report di utilizzo di SMS giornalieri.

Visualizzazione dei parametri e dei log Amazon CloudWatch per le consegne di SMS

Puoi utilizzare Amazon CloudWatch e Amazon CloudWatch Logs per monitorare la consegna degli SMS.

Argomenti

- Visualizzazione dei parametri di Amazon CloudWatch
- Visualizzazione di CloudWatch Logs
- Log di esempio per gli SMS inviati con successo
- Log di esempio per gli SMS falliti
- Ragioni per un SMS fallito

Visualizzazione dei parametri di Amazon CloudWatch

Amazon SNS raccoglie automaticamente i parametri relativi alle consegne dei messaggi SMS e li invia ad Amazon CloudWatch. Per monitorarle, affidati a CloudWatch e crea allarmi che ti avvertano quando una metrica supera una certa soglia. Per esempio, puoi monitorare i parametri CloudWatch per conoscere il tasso di consegna degli SMS e gli addebiti dall'inizio del mese alla data corrente.

Per ulteriori informazioni sul monitoraggio dei parametri CloudWatch, l'impostazione di allarmi e il tipo di parametri disponibili, consulta Monitoraggio di argomenti Amazon SNS tramite CloudWatch.

Visualizzazione di CloudWatch Logs

Puoi raccogliere informazioni sull'avvenuta o sulla mancata consegna dei messaggi SMS abilitando Amazon SNS a scrivere su Amazon CloudWatch Logs. Per ogni SMS inviato, Amazon SNS creerà un log con il costo del messaggio, lo stato di consegna e la motivazione in caso di fallimento, il tempo di sosta e altre informazioni.

Per abilitare e visualizzare CloudWatch Logs per i tuoi messaggi SMS

1. Accedi alla console Amazon SNS.
2. Nel menu della console, imposta lo strumento di selezione della regione su una regione che supporti la messaggistica SMS.
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Nella pagina Mobile text messaging (SMS) (Messaggi di testo mobili (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).
5. Nella pagina successiva, espandere la sezione Delivery status logging (Registrazione dello stato di consegna).
6. Per Success sample rate (Frequenza di campionamento riuscita), specifica la percentuale di consegne SMS riuscite per le quali Amazon SNS scriverà i log in CloudWatch Logs. Ad esempio:
   - Per generare dei log unicamente per le consegne non riuscite, imposta questo valore su 0.
   - Per generare dei log per il 10% delle consegne riuscite, imposta 10.

   Se non specifichi una percentuale, Amazon SNS genera dei log per tutte le consegne riuscite.
7. Per fornire le autorizzazioni necessarie, eseguire una delle seguenti operazioni:
   - Per creare un nuovo ruolo del servizio, scegli Creazione di un nuovo ruolo del servizio e poi Creazione di nuovi ruoli. Nella pagina successiva, scegli Abilita per consentire ad Amazon SNS di accedere in scrittura alle risorse del tuo account.
   - Per utilizzare un ruolo del servizio esistente, scegli Utilizzare il ruolo del servizio esistente e quindi incollare il nome ARN nella casella Ruolo IAM per consegne riuscite e non riuscite.

Il ruolo di servizio specificato deve consentire l'accesso in scrittura alle risorse dell'account. Per ulteriori informazioni sulla creazione di un ruolo IAM, consultare Creazione di un ruolo per un servizio AWS nella Guida per l'utente di IAM.

8. Scegliere Save changes (Salva modifiche).
9. Nella pagina Messaggi di testo per dispositivi mobili (SMS) vai alla pagina Registri dello stato di per visualizzare tutti i registri disponibili.

> **Note**
>
> A seconda del corriere del numero di telefono di destinazione, possono essere necessarie fino a 72 ore affinché i log delle consegne vengano visualizzati nella console Amazon SNS.

Log di esempio per gli SMS inviati con successo

Il log dello stato di consegna per un SMS inviato con successo sarà simile al seguente esempio:

```
{
    "notification": {
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "timestamp": "2016-06-28 00:40:34.558"
    },
    "delivery": {
        "phoneCarrier": "My Phone Carrier",
        "mnc": 270,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 310,
        "providerResponse": "Message has been accepted by phone carrier",
        "dwellTimeMs": 599,
        "dwellTimeMsUntilDeviceAck": 1344
    },
    "status": "SUCCESS"
}
```

Log di esempio per gli SMS falliti

Il log dello stato di consegna per un SMS fallito sarà simile al seguente esempio:

```
{
    "notification": {
        "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
        "timestamp": "2016-06-28 00:40:34.559"
    },
    "delivery": {
        "mnc": 0,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 0,
        "providerResponse": "Unknown error attempting to reach phone",
        "dwellTimeMs": 1420,
        "dwellTimeMsUntilDeviceAck": 1692
    },
    "status": "FAILURE"
}
```

Ragioni per un SMS fallito

La ragione per un SMS fallito viene fornita dall'attributo `providerResponse`. L'invio di un messaggio SMS potrebbe fallire per i seguenti motivi:

- Bloccato come spam dall'operatore telefonico
- La destinazione è in un elenco bloccato
- Il numero di telefono non è valido
- Il testo del messaggio non è valido
- L'operatore telefonico ha bloccato il messaggio
- L'operatore telefonico non è raggiungibile/disponibile
- Il telefono ha bloccato l'SMS
- Il telefono è in un elenco bloccato
- Il telefono non è raggiungibile/disponibile
- Il numero di telefono è stato escluso

- La consegna supera il costo massimo
- Errore sconosciuto nel tentativo di raggiungere il telefono

Visualizzazione dei report di utilizzo di SMS giornalieri

Puoi monitorare le consegne di SMS effettuando la sottoscrizione a report di utilizzo di SMS giornalieri a partire da Amazon SNS. Per ogni giorno in cui invii almeno un messaggio SMS, Amazon SNS fornisce un report di utilizzo in un file CSV per il bucket Amazon S3 specificato. Ci vogliono 24 ore prima che i report di utilizzo degli SMS siano disponibili nel bucket S3.

Argomenti

Informazioni contenute nei report di utilizzo giornalieri

Il report di utilizzo include le seguenti informazioni per ogni SMS inviato dal tuo account.

Tenere presente che il report non include messaggi che vengono inviati ai destinatari che hanno scelto di non ricevere i messaggi.

- Tempo di pubblicazione per messaggio (in UTC)
- ID messaggio
- Numero di telefono di destinazione
- Tipo di messaggio
- Stato della consegna
- Costo del messaggio (in USD)
- Numero di parte (un messaggio viene suddiviso in più parti se è troppo lungo per un unico messaggio)
- Numero totale di parti

> **Note**
>
> Se Amazon SNS non ha ricevuto il numero di parte, impostiamo il suo valore su zero.

Sottoscrizione ai report di utilizzo giornalieri

Per eseguire la sottoscrizione ai report di utilizzo giornalieri, devi creare un bucket Amazon S3 con le autorizzazioni appropriate.

Creazione di un bucket Amazon S3 per i report di utilizzo giornalieri

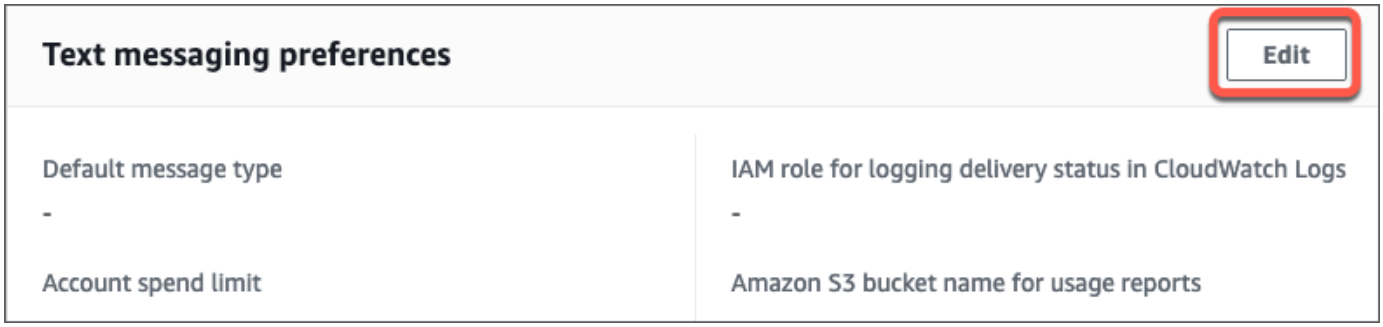
1. Da Account AWS che invia messaggi SMS, accedi alla console Amazon S3.
2. Scegliere Create Bucket (Crea bucket).
3. Per Bucket Name (Nome bucket), si consiglia di immettere un nome univoco per l'account e l'organizzazione. Ad esempio, utilizzare il modello `<my-bucket-prefix>-<account_id>-<org-id>`.

   Per informazioni sulle convenzioni e sulle restrizioni per i nomi di bucket, consulta Regole per la denominazione dei bucket nella Guida per l'utente di Amazon Simple Storage Service.
4. Scegliere Create (Crea).
5. Nella tabella All Buckets (Tutti i bucket), seleziona il bucket.
6. Nella sezione Permissions (Autorizzazioni), scegliere Bucket policy (Policy bucket).
7. Nella finestra Bucket Policy Editor (Editor policy bucket), specifica una policy che autorizzi il principale del servizio Amazon SNS a scrivere nel tuo bucket. Per un esempio, consultare Esempio di policy di bucket.

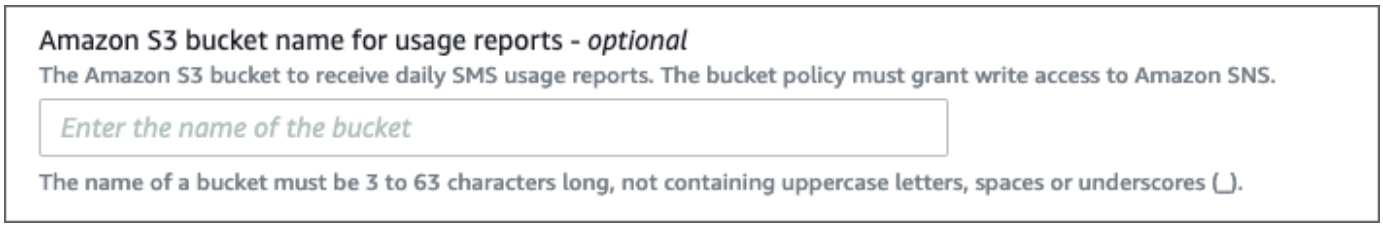
   Se si utilizza la policy di esempio, ricordarsi di sostituire *`my-s3-bucket`* con il nome bucket scelto nel passaggio 3.
8. Selezionare Salva.

Sottoscrizione ai report di utilizzo giornalieri

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
3. Nella pagina Text messaging (SMS) (Messaggi di testo (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).



4. Nella pagina Edit text messaging preferences (Modifica preferenze di messaggistica di testo), nella sezione Details (Dettagli), specificare Amazon S3 bucket name for usage reports (Nome del bucket Amazon S3 per i report di utilizzo).



5. Scegliere Save changes (Salva modifiche).

Esempio di policy di bucket

La policy seguente consente al principale del servizio Amazon SNS di eseguire le operazioni `s3:PutObject`, `s3:GetBucketLocation` e `s3:ListBucket`.

AWS fornisce strumenti per tutti i servizi che dispongono di entità principali del servizio a cui è stato consentito l'accesso alle risorse del tuo account. Quando il principale di una istruzione della policy del bucket Amazon S3 è unPrincipale del servizio AWS, è possibile utilizzare le `aws:SourceArn` o `aws:SourceAccount` chiavi di condizione globali per proteggersi dal problema del "confused deputy". Per limitare la regione e l'account da cui il bucket può ricevere report di utilizzo giornalieri, utilizzare `aws:SourceArn` come mostrato nell'esempio sottostante. Se non si desidera limitare le regioni che possono generare questi report, utilizzare `aws:SourceAccount` per limitare in base a quale account sta generando i report. Se non si conosce l'ARN della risorsa, utilizzare `aws:SourceAccount`.

Utilizzare l'esempio seguente che include la protezione "confused deputy" quando crei un bucket Amazon S3 per ricevere report di utilizzo di SMS giornalieri da Amazon SNS.

```
{
 "Version": "2008-10-17",
 "Statement": [{
```

```
  "Sid": "AllowPutObject",
  "Effect": "Allow",
  "Principal": {
   "Service": "sns.amazonaws.com"
  },
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::my-s3-bucket/*",
  "Condition": {
   "StringEquals": {
    "aws:SourceAccount": "account_id"
   },
   "ArnLike": {
    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
   }
  }
 },
 {
  "Sid": "AllowGetBucketLocation",
  "Effect": "Allow",
  "Principal": {
   "Service": "sns.amazonaws.com"
  },
  "Action": "s3:GetBucketLocation",
  "Resource": "arn:aws:s3:::my-s3-bucket",
  "Condition": {
   "StringEquals": {
    "aws:SourceAccount": "account_id"
   },
   "ArnLike": {
    "aws:SourceArn": "arn:aws:sns:region:account_id:*"
   }
  }
 },
 {
  "Sid": "AllowListBucket",
  "Effect": "Allow",
  "Principal": {
   "Service": "sns.amazonaws.com"
  },
  "Action": "s3:ListBucket",
  "Resource": "arn:aws:s3:::my-s3-bucket",
  "Condition": {
   "StringEquals": {
    "aws:SourceAccount": "account_id"
```

```
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  }
 ]
}
```

> **Note**
>
> Puoi pubblicare report sull'utilizzo nei bucket Amazon S3 di proprietà dell'Account AWS che è specificato nell'elemento `Condition` nella policy di Amazon S3. Per pubblicare i report sull'utilizzo in un bucket Amazon S3 che un altro Account AWS possiede, vedere Come posso copiare oggetti S3 da un altroAccount AWS?.

Esempio di report di utilizzo giornaliero

Dopo la sottoscrizione ai report di utilizzo giornalieri, Amazon SNS inserisce un file CSV con i dati sull'utilizzo nella seguente posizione:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Ogni file può contenere fino a 50.000 record. Se i record di un giorno superano questa quota, Amazon SNS aggiunge più file.

Di seguito viene riportato un esempio di report:

```
PublishTimeUTC,MessageId,DestinationPhoneNumber,MessageType,DeliveryStatus,PriceInUSD,PartNumbe
2016-05-10T03:00:29.476Z,96a298ac-1458-4825-
a7eb-7330e0720b72,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.90084,0,1
2016-05-10T03:00:29.561Z,1e29d394-
d7f4-4dc9-996e-26412032c344,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.34322,0,1
2016-05-10T03:00:30.769Z,98ba941c-afc7-4c51-
ba2c-56c6570a6c08,1XXX5550100,Transactional,Message has been accepted by phone
 carrier,0.27815,0,1
```

Gestione dei numeri di telefono e delle sottoscrizioni SMS

Amazon SNS offre diverse opzioni per gestire chi riceve messaggi SMS dal tuo account. Con frequenza limitata, puoi inserire nell'elenco di invio numeri di telefono di destinatari che hanno scelto di non ricevere i messaggi SMS dal tuo account. Per interrompere l'invio di messaggi a sottoscrizioni SMS, puoi rimuovere le sottoscrizioni o gli argomenti in cui effettui la pubblicazioni destinate alle sottoscrizioni.

Argomenti

Richiesta di non ricevere i messaggi SMS

Dove previsto da leggi e regolamenti locali (ad esempio negli Stati Uniti e in Canada), i destinatari di messaggi SMS possono usare i propri dispositivi per chiedere di essere esclusi dalla ricezione dei messaggi utilizzando una delle risposte seguenti:

- ARRET (francese)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

Per cancellarsi dalla ricezione di messaggi, il destinatario deve rispondere allo stesso Numero di origine utilizzato da Amazon SNS per consegnare il messaggio. Dopo la disattivazione, il destinatario non riceverà più i messaggi SMS recapitati dal destinatario Account AWS a meno che non indichi il numero di telefono.

Se il numero di telefono ha una sottoscrizione a un argomento Amazon SNS, l'esclusione dalla ricezione non comporta la rimozione della sottoscrizione, ma i messaggi SMS verranno consegnati alla sottoscrizione specificata solo se inserisci il numero di telefono nell'elenco di invio.

Gestione dei numeri di telefono e delle sottoscrizioni (console)

Per specificare quali numeri di telefono ricevono messaggi dal tuo account, puoi usare la console Amazon SNS.

Inserimento nell'elenco di invio di un numero di telefono escluso

Puoi visualizzare i numeri di telefono esclusi dalla ricezione di messaggi SMS dal tuo account e puoi inserirli nell'elenco di invio per riprendere la distribuzione dei messaggi.

Puoi inserire un numero di telefono nell'elenco di invio solo una volta ogni 30 giorni.

1. Accedi alla console Amazon SNS.
2. Nel menu della console, imposta lo strumento di selezione della regione su una regione che supporti la messaggistica SMS.
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Nella pagina Text messaging (SMS) (SMS) scegli View opted out phone numbers (Visualizza numeri di telefono esclusi). La pagina Opted out phone numbers (Numeri di telefono esclusi) mostra i numeri di telefono esclusi.
5. Seleziona la casella di controllo del numero di telefono da inserire nell'elenco di invio, quindi scegli Opt in (Inserisci). Il numero di telefono non risulterà più escluso e riceverà i messaggi SMS che invierai.

Eliminazione di una sottoscrizione SMS

Elimina una sottoscrizione SMS per interrompere l'invio di messaggi SMS al numero specificato quando effettui pubblicazioni nei tuoi argomenti.

1. Nel riquadro di navigazione, scegli Sottoscrizioni.
2. Seleziona le caselle di controllo delle sottoscrizioni da eliminare. Scegli Actions (Operazioni), quindi Delete Subscriptions (Elimina sottoscrizioni).
3. Nella finestra Delete (Elimina) scegli Delete (Elimina). Amazon SNS elimina la sottoscrizione visualizzandone poi la conferma.

Eliminazione di un argomento

Elimina un argomento quando non desideri più pubblicare messaggi per gli endpoint dotati di sottoscrizione all'argomento.

1. Nel pannello di navigazione, scegliere Argomenti.
2. Seleziona le caselle di controllo degli argomenti da eliminare. Scegli Actions (Operazioni), quindi Delete Topics (Elimina argomenti).
3. Nella finestra Delete (Elimina) scegli Delete (Elimina). Amazon SNS elimina l'argomento visualizzandone poi la conferma.

Gestione dei numeri di telefono e delle sottoscrizioni (SDK AWS)

Puoi utilizzare gli AWS SDK per effettuare richieste programmatiche ad Amazon SNS e gestire quali numeri di telefono possono ricevere messaggi SMS dal tuo account.

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento agli AWS SDK e agli strumenti.

Visualizzazione di tutti i numeri di telefono esclusi

Per visualizzare tutti i numeri di telefono esclusi, invia una richiesta `ListPhoneNumbersOptedOut` con l'API di Amazon SNS.

I seguenti esempi di codice mostrano come utilizzare. `ListPhoneNumbersOptedOut`

CLI

AWS CLI

Elencare le opzioni di opt-out dei messaggi SMS

Nell'esempio `list-phone-numbers-opted-out` seguente sono elencati i numeri di telefono per i quali sono abilitate le opzioni di opt-out per la ricezione di SMS.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
```

```
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Per i dettagli sull'API, consulta ListPhoneNumbersOptedOut AWS CLICommand Reference.

Java

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.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.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
```

```
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per i dettagli sull'API, consulta la ListPhoneNumbersOptedOutsezione AWS SDK for PHP API Reference.

Verifica dell'esclusione di un numero di telefono

Per verificare se un numero di telefono è escluso, invia una richiesta `CheckIfPhoneNumberIsOptedOut` con l'API di Amazon SNS.

I seguenti esempi di codice mostrano come utilizzare`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
   using System;
   using System.Threading.Tasks;
   using Amazon.SimpleNotificationService;
   using Amazon.SimpleNotificationService.Model;

   /// <summary>
   /// This example shows how to use the Amazon Simple Notification Service
   /// (Amazon SNS) to check whether a phone number has been opted out.
   /// </summary>
   public class IsPhoneNumOptedOut
   {
       public static async Task Main()
       {
           string phoneNumber = "+15551112222";

           IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

           await CheckIfOptedOutAsync(client, phoneNumber);
       }

       /// <summary>
       /// Checks to see if the supplied phone number has been opted out.
       /// </summary>
       /// <param name="client">The initialized Amazon SNS Client object used
       /// to check if the phone number has been opted out.</param>
       /// <param name="phoneNumber">A string representing the phone number
       /// to check.</param>
       public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
       {
           var request = new CheckIfPhoneNumberIsOptedOutRequest
           {
               PhoneNumber = phoneNumber,
           };

           try
           {
               var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

               if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
```

```
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for .NET API Reference.

CLI

AWS CLI

Controllo delle impostazioni di opt-out dei messaggi SMS per un numero di telefono

L'`check-if-phone-number-is-opted-out`esempio seguente verifica se al numero di telefono specificato è stata disattivata la ricezione di messaggi SMS dall' AWS account corrente.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Per i dettagli sull'API, vedere CheckIfPhoneNumberIsOptedOutin AWS CLI Command Reference.

Java

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.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

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

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for PHP API Reference.

Inserimento nell'elenco di invio di un numero di telefono escluso

Per inserire un numero di telefono nell'elenco di invio, invia una richiesta `OptInPhoneNumber` con l'API di Amazon SNS.

Puoi inserire un numero di telefono nell'elenco di invio solo una volta ogni 30 giorni.

Eliminazione di una sottoscrizione SMS

Per eliminare una sottoscrizione SMS da un argomento Amazon SNS, recupera l'ARN della sottoscrizione inviando una richiesta `ListSubscriptions` con l'API di &SNS;, quindi passa l'ARN a una richiesta `Unsubscribe`.

I seguenti esempi di codice mostrano come utilizzare`Unsubscribe`.

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Annulla l'iscrizione a un argomento tramite un ARN di sottoscrizione.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for .NET .

C++

SDK per C++

> **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.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Annullamento della sottoscrizione a un argomento

Nell'esempio `unsubscribe` seguente viene eliminata la sottoscrizione specificata a un argomento.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Questo comando non produce alcun output.

- Per informazioni dettagliate sulle API, consulta Unsubscribe nel Riferimento ai comandi AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for JavaScript .

Kotlin

SDK per Kotlin

> **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.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API di SDK AWS per Kotlin.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for PHP .

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per l'API SDK for Python (Boto3)AWS .

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta Unsubscribe nella documentazione di riferimento dell'SDK AWS per l'API SAP ABAP.

Eliminazione di un argomento

Per eliminare un argomento e tutte le relative sottoscrizioni, recupera l'ARN dell'argomento inviando una richiesta `ListTopics` con l'API di Amazon SNS, quindi passa l'ARN alla richiesta `DeleteTopic`.

I seguenti esempi di codice mostrano come usare`DeleteTopic`.

.NET

AWS SDK for .NET

> **Note**
>
> C'è 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 argomento in base all'ARN dell'argomento.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Eliminazione di un argomento SNS

Nell'esempio `delete-topic` seguente viene eliminato l'argomento SNS specificato.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta DeleteTopic AWS CLICommand Reference.

Go

SDK per Go V2

> **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.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for Go API Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Per i dettagli sull'API, DeleteTopicconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Per i dettagli sull'API, consulta DeleteTopic AWSSDK for Python (Boto3) API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per i dettagli sulle API, DeleteTopicconsulta AWS SDK for SAP ABAP API reference.

Regioni e paesi supportati

> **Important**
>
> A partire dal 31 agosto 2023, un numero dedicato come un numero 10DLC o un numero verde è necessario per inviare messaggi SMS negli Stati Uniti e nei suoi territori (Guam, Porto Rico, Isole Samoa americane e Isole Vergini US).

Attualmente, Amazon SNS supporta la messaggistica SMS nelle seguenti regioni: AWS

Nome Regione	Regione	Endpoint	Protocollo
Stati Uniti orientali (Ohio)	us-east-2	sns.us-east-2.amazonaws.com	HTTP e HTTPS
US East (N. Virginia)	us-east-1	sns.us-east-1.amazonaws.com	HTTP e HTTPS
US West (N. California)	us-west-1	sns.us-west-1.amazonaws.com	HTTP e HTTPS
US West (Oregon)	us-west-2	sns.us-west-2.amazonaws.com	HTTP e HTTPS
Africa (Cape Town)	af-south-1	sns.af-south-1.amazonaws.com	HTTP e HTTPS
Asia Pacific (Hyderabad)	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP e HTTPS
Asia Pacifico (Giacarta)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP e HTTPS

Nome Regione	Regione	Endpoint	Protocollo
Asia Pacifico (Melbourne)	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP e HTTPS
Asia Pacific (Mumbai)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP e HTTPS
Asia Pacifico (Osaka-Locale)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP e HTTPS
Asia Pacific (Singapore)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP e HTTPS
Asia Pacific (Sydney)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP e HTTPS
Asia Pacifico (Tokyo)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP e HTTPS
Canada (Central)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP e HTTPS
Europe (Frankfurt)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP e HTTPS
Europa (Irlanda)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP e HTTPS
Europe (London)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP e HTTPS
Europa (Milano)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP e HTTPS
Europe (Paris)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP e HTTPS
Europa (Spagna)	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP e HTTPS

Nome Regione	Regione	Endpoint	Protocollo
Europa (Stoccolma)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP e HTTPS
Europa (Zurigo)	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP e HTTPS
Israele (Tel Aviv)	il-central-1	sns.il-central-1.amazonaws.com	HTTP e HTTPS
Medio Oriente (Bahrein)	me-south-1	sns.me-south-1.amazonaws.com	HTTP e HTTPS
Medio Oriente (Emirati Arabi Uniti)	me-central-1	sns.me-central-1.amazonaws.com	HTTP e HTTPS
Sud America (São Paulo)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP e HTTPS
AWS GovCloud (Stati Uniti orientali)	us-gov-east-1	peccato. us-gov-east-1. amazonaws.com	HTTP e HTTPS
AWS GovCloud (Stati Uniti occidentali)	us-gov-west-1	peccato. us-gov-west-1. amazonaws.com	HTTP e HTTPS

Puoi utilizzare Amazon SNS per inviare SMS nei seguenti paesi e regioni:

> **Note**
>
> L'utilizzo degli ID mittente nei paesi supportati può migliorare il recapito degli SMS.

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
A						
Afghanistan	AF	93	No	No	Sì	No
Albania	AL	355	No	No	Sì	No
Algeria	DZ	213	No	No	Sì	No
Andorra	AD	376	No	No	Sì	No
Anguilla	AI	1-264	No	No	Sì	No
Antigua e Barbuda	AG	1-268	No	No	Sì	No
Argentina	AR	54	Sì	No	No	No
Armenia	AM	374	No	No	Sì	No
Aruba	AW	297	No	No	Sì	No
Australia	AU	61	No	Sì	Registrazione richiesta[1]	Sì
Austria	AT	43	Sì	Sì	Sì	Sì
Azerbaigian	AZ	994	No	No	Sì	No
B						
Bahamas	BS	1-242	No	No	No	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Bahrein	BH	973	No	No	Sì	No
Bangladesh	BD	880	No	No	Sì	No
Barbados	BB	1-246	No	No	Sì	No
Bielorussia	BY	375	No	No	Registrazione richiesta[1]	No
Belgio	BE	32	No	Sì	No	Sì
Belize	BZ	501	No	No	Sì	No
Bermuda	BM	1-441	No	No	Sì	No
Bhutan	BT	975	No	No	Sì	No
Bolivia	BO	591	No	No	Sì	No
Bosnia ed Erzegovina	BA	387	No	No	Sì	No
Botswana	BW	267	No	No	Sì	No
Brasile	BR	55	Sì	No	No	Sì
Brunei	BN	673	No	No	Sì	No
Bulgaria	BG	359	Sì	No	Sì	Sì
Burkina Faso	BF	226	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Burundi	BL	257	No	No	Sì	No
C						
Cambogia	KH	855	No	No	Sì	No
Camerun	CM	237	No	No	Sì	No
Canada	CA	1	Sì	Sì	No	Sì
Capo Verde	CV	238	No	No	Sì	No
Isole Cayman	KY	1-345	No	No	No	No
Repubblica Centrafricana	CF	236	No	No	Sì	No
Ciad	TD	235	No	No	Sì	No
Cile	CL	56	Sì	Sì	No	Sì
Cina	CN	86	Sì	No	No [2]	Sì
Colombia	CO	57	Sì	Sì	No	Sì
Comore	KM	269	No	No	Sì	No
Isole Cook	CK	682	No	No	Sì	Sì
Costa Rica	CR	506	No	No	No	No
Croazia	HR	385	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Cipro	CY	357	No	No	Sì	No
Cechia (Repubblica Ceca)	CZ	420	No	Sì	Sì	Sì
D						
Repubblica Democratica del Congo	CD	243	No	No	Sì	No
Danimarca	DK	45	Sì	Sì	Sì	Sì
Gibuti	DJ	253	No	No	Sì	No
Dominica	DM	1-767	No	No	Sì	No
Repubblica Dominicana	DO	1-809, 1-829, 1-849	Sì	No	No	Sì
E						
Ecuador	EC	593	Sì	No	No	Sì
Egitto	EG	20	Sì	No	Registrazione richiesta[1]	Sì
El Salvador	SV	503	No	No	No	No
Guinea Equatoriale	GQ	240	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Eritrea	ER	291	No	No	Sì	No
Estonia	EE	372	No	Sì	Sì	Sì
Etiopia	ET	251	No	No	Sì	No
F						
Isole Fær Øer	FO	298	No	No	Sì	No
Figi	FJ	679	No	No	Sì	No
Finlandia	FI	358	Sì	Sì	Sì	Sì
Francia	FR	33	Sì	No	Sì	Sì
Guyana francese	GF	594	No	No	Sì	No
Polinesia francese	PF	689	No	No	Sì	No
G						
Gabon	GA	241	No	No	Sì	No
Gambia	GM	220	No	No	Sì	No
Georgia	GE	995	No	No	Sì	No
Germania	DE	49	Sì	Sì	Sì	Sì
Ghana	GH	233	No	No	Sì	No
Gibilterra	GI	350	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Grecia	GR	30	No	No	Sì	No
Groenlandia	GL	299	No	No	Sì	No
Grenada	GD	1-473	No	No	Sì	No
Guadalupa	GP	590	No	No	Sì	No
Guam	GU	1-671	No	No	No	Sì
Guatemala	GT	502	No	No	No	No
Guernsey	GG	44-1481	No	No	Sì	No
Guinea	GN	224	No	No	Sì	No
Guinea-Bissau	GW	245	No	No	Sì	N/D
Guyana	GY	592	No	No	Sì	No
H						
Haiti	H	509	No	No	Sì	No
Honduras	HN	504	No	No	Sì	No
Hong Kong	HK	852	No	Sì	Sì	Sì
Ungheria	HU	36	No	Sì	No	Sì
I						
Islanda	IS	354	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
India	IN	91	Sì	Sì[4]	Registrazione richiesta[3]	Sì
Indonesia	ID	62	No	No	Sì	No
Iraq	IQ	964	No	No	Sì	No
Irlanda	IE	353	No	Sì	Sì	Sì
Isola di Man	IM	44-1624	No	No	Sì	No
Israele	IL	972	No	Sì	Sì	Sì
Italia	IT	39	Sì	Sì	Sì	Sì
Costa d'Avorio	CI	225	No	No	Sì	No
J						
Giamaica	JM	1-876	No	No	Sì	No
Giappone	JP	81	Sì	Sì	Sì	Sì
Jersey	JE	44-1534	No	Sì	Sì	Sì
Giordania	JO	962	No	No	Registrazione richiesta[1]	No
K						
Kazakistan	KZ	7	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Kenya	KE	254	No	No	Sì	No
Kosovo	XK	383	No	No	Sì	No
Kuwait	KW	965	No	No	Registrazione richiesta[1]	No
Kirghizistan	KG	996	No	No	Sì	No
L						
Laos	LA	856	No	No	Sì	No
Lettonia	LV	371	No	No	Sì	No
Libano	LB	961	No	No	Sì	No
Lesotho	LS	266	No	No	Sì	No
Liberia	LR	231	No	Sì	No	
Libia	LY	218	No	No	Sì	No
Liechtenstein	LI	423	No	No	Sì	No
Lituania	LT	370	No	Sì	Sì	Sì
Lussemburgo	LU	352	No	Sì	Sì	Sì
M						

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Macau	MO	853	No	No	Sì	No
Macedonia	MK	389	No	No	Sì	No
Madagascar	MG	261	No	No	Sì	No
Malawi	MW	265	No	No	Sì	No
Malesia	MY	60	Sì	No	No	Sì
Maldive	MV	960	No	No	Sì	No
Mali	ML	223	No	No	Sì	No
Malta	MT	356	No	No	Sì	No
Isole Marshall	MH	692	No	No	No	No
Martinica	MQ	596	No	No	Sì	No
Mauritania	MR	222	No	No	Sì	No
Mauritius	MU	230	No	No	Sì	No
Mayotte	YT	262	No	No	Sì	No
Messico	MX	52	Sì	No	No	Sì
Micronesia (Stati Federati di Micronesia)	FM	691	No	No	No	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Moldavia	MD	373	No	No	Sì	No
Monaco	MC	377	No	No	No	No
Mongolia	MN	976	No	No	Sì	No
Montenegro	ME	382	No	No	Sì	No
Montserrat	MS	1-664	No	No	Sì	No
Marocco	MA	212	Sì	No	Sì	Sì
Mozambico	MZ	258	No	No	No	No
Birmania	MM	95	No	Sì	Sì	Sì
N						
Namibia	NA	264	No	No	Sì	No
Nepal	NP	977	No	No	Sì	No
Paesi Bassi	NL	31	Sì	Sì	Sì	Sì
Antille Olandesi	AN	599	No	No	Sì	No
Nuova Caledonia	NC	687	No	No	Sì	No
Nuova Zelanda[6]	NZ	64	Sì	No	No	Sì

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Nicaragua	NI	505	No	No	No	No
Niger	NE	227	No	No	Sì	No
Nigeria	NG	234	No	No	Sì	No
Niue	NU	683	No	No	Sì	No
Norvegia	NO	47	No	Sì	Sì	Sì
O						
Oman	OM	968	No	No	No	N/D
P						
Pakistan	PK	92	No	No	Sì	N/D
Palestina	PS	970	No	No	Sì	No
Panama	PA	507	No	No	Sì	No
Papua Nuova Guinea	PG	675	No	No	Sì	No
Paraguay	PY	595	No	No	No	No
Perù	PE	51	Sì	No	No	Sì
Filippine	PH	63	No	Sì	Registrazione richiesta[1]	Sì
Polonia	PL	48	No	Sì	Sì	Sì

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Portogallo	PT	351	No	Sì	Sì	Sì
Porto Rico	PR	1-797, 1-939	No	No	No	Sì
Q						
Qatar	QA	974	No	No	Registrazione richiesta[1]	No
R						
Repubblica del Congo	CG	242	No	No	No	No
Riunione (Francia)	RE	262	No	No	Sì	No
Romania	RO	40	No	Sì	Sì	Sì
Russia	RU	7	Sì	No	Registrazione richiesta[1]	Sì
Ruanda	RW	250	No	No	Sì	No
S						
Saint Kitts e Nevis	KN	1-869	No	No	No	No
Santa Lucia	LC	1-758	No	No	No	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Samoa	WS	685	No	No	Sì	No
San Marino	SM	378	No	No	Sì	No
São Tomé e Príncipe	ST	239	No	No	Sì	No
Arabia Saudita	SA	966	No	Sì[4]	Registrazione richiesta[1]	No
Senegal	SN	221	No	No	Sì	No
Serbia	RS	381	No	No	Sì	No
Seychelles	SC	248	No	No	Sì	No
Sierra Leone	SL	232	No	No	Sì	No
Singapore	SG	65	Sì	Sì	Sì[5]	Sì
Slovacchia	SK	421	No	Sì	Sì	No
Slovenia	SI	386	No	No	Sì	No
Isole Salomone	SB	677	No	No	Sì	No
Somalia	SO	252	No	No	Sì	No
Sudafrica	ZA	27	Sì	Sì	No	Sì
Corea del Sud	KR	82	No	No	No	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Sudan del Sud	SS	211	No	No	Sì	No
Spagna	ES	34	Sì	Sì	Sì	Sì
Sri Lanka	LK	94	No	No	Registrazione richiesta[1]	No
Suriname	SR	597	No	No	Sì	No
Swaziland	SZ	268	No	No	Sì	No
Svezia	SE	46	Sì	Sì	Sì	Sì
Svizzera	CH	41	No	Sì	Sì	Sì
T						
Taiwan	TW	886	No	Sì	No	Sì
Tagikistan	TJ	992	No	No	Sì	No
Tanzania	TX	255	No	No	Sì	No
Tailandia	TH	66	No	Sì	Registrazione richiesta[1]	Sì
Timor Est	TL	670	No	No	Sì	No
Togo	TG	228	No	No	Sì	No
Tonga	TO	676	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
Trinidad e Tobago	TT	1-868	No	No	Sì	No
Tunisia	TN	216	No	No	Sì	No
Turchia	TR	90	No	No	Registrazione richiesta[1]	No
Turkmenistan	TM	993	No	No	No	No
Turks e Caicos	TC	1-649	No	No	Sì	No
Tuvalu	TC	688	No	No	Sì	No
U						
Uganda	UG	256	No	No	Sì	No
Ucraina	UA	380	No	Sì	Sì	Sì
Emirati Arabi Uniti (EAU)	AE	971	Sì	Sì	Registrazione richiesta[1]	Sì
Regno Unito	GB	44	Sì	Sì	Sì	Sì
Stati Uniti	US	1	Sì	Sì	No	Sì
Uruguay	UY	598	Sì	No	No	Sì
Uzbekistan	UZ	998	No	No	Sì	No

Paese o regione	Codice ISO	Codice di composizione	Supporta i codici brevi	Supporta i codici lunghi	Supporta ID mittente	Supporto SMS bidirezionali
V						
Vanuatu	VU	678	No	No	Sì	No
Venezuela	VE	58	No	No	No	No
Vietnam	VN	84	No	No	Registrazione richiesta[1]	No
Isole Vergini britanniche	VG	1-284	No	No	Sì	No
Isole Vergini, US	VI	1-340	No	No	No	Sì
W						
X						
Y						
Yemen	YE	967	No	No	Sì	No
Z						
Zambia	ZM	260	No	No	Sì	No
Zimbabwe	ZW	263	No	No	Sì	No

Note

1.

I mittenti devono utilizzare un ID mittente alfabetico pre-registrato. Per richiedere un ID mittente a AWS Support, vedere. Richiesta di ID mittente per la messaggistica SMS con Amazon SNS Alcuni paesi richiedono ai mittenti di soddisfare requisiti specifici o di rispettare determinate restrizioni per ottenere l'approvazione. In questi casi, AWS Support potrebbe contattarti per ulteriori informazioni dopo aver inviato la richiesta relativa all'ID mittente.

2. I mittenti devono utilizzare un modello pre-registrato per ogni tipo di messaggio che intendono inviare. Se un mittente non soddisfa questo requisito, i messaggi verranno bloccati. Per registrare un modello, apri una custodia Amazon SNS SMS con. AWS Support Quando crei il caso, fornisci le stesse informazioni utilizzate per richiedere un ID mittente. Per ulteriori informazioni, consulta Richiesta di ID mittente per la messaggistica SMS con Amazon SNS. Alcuni paesi richiedono ai mittenti di soddisfare requisiti addizionali specifici o di rispettare determinate restrizioni per ottenere l'approvazione. In questi casi, AWS Support potrebbe chiederti ulteriori informazioni.

   > **Note**
   >
   > Per inviare messaggi in Cina, devi prima registrare i tuoi modelli AWS Support per l'approvazione.

3. I mittenti devono utilizzare un ID mittente alfabetico pre-registrato. Sono necessarie ulteriori fasi di registrazione. Per ulteriori informazioni, consulta Requisiti per la registrazione dell'ID mittente per l'India.

4. I codici lunghi in questi paesi supportano solo la messaggistica in entrata. In altre parole, non è possibile utilizzare questi codici lunghi verso i tuoi destinatari, ma solo per ricevere messaggi dai destinatari. Questi codici lunghi sono utili per consentire ai destinatari di negare il consenso per la ricezione di messaggi se utilizzi un ID mittente alfabetico, poiché gli ID mittente supportano solo i messaggi in uscita.

5. Amazon SNS permette di inviare traffico SMS a Singapore utilizzando un ID mittente registrato nel Singapore SMS Sender ID Registry (SSIR), un registro creato dall'Autorità per lo sviluppo dei media di informazione e comunicazione (IMDA) di Singapore. Per ulteriori informazioni sui requisiti per l'utilizzo di un ID mittente di Singapore, consulta Requisiti per la registrazione dell'ID mittente per Singapore.

Puoi anche inviare traffico SMS a Singapore utilizzando ID mittente non registrati o tipi di identità di origine alternativi, come codici brevi o codici lunghi.

6. Senza un codice breve dedicato, Amazon SNS tenta comunque di inviare messaggi ai destinatari della Nuova Zelanda utilizzando un pool condiviso di codici brevi. A causa delle restrizioni dei carrier locali in merito ai numeri condivisi, la consegna su questi numeri condivisi viene effettuata in base al miglior tentativo. Pertanto, Amazon SNS consiglia vivamente di procurarsi un codice breve dedicato per tutto il traffico inviato in Nuova Zelanda. I messaggi contenenti URL devono essere inseriti nell'elenco consentiti tramite il processo dedicato di codice breve. Per ulteriori informazioni su come acquistare un codice breve, consulta Richiesta di codici brevi dedicati per la messaggistica SMS con Amazon SNS.

Best practice per il canale SMS

Gli utenti di telefonia mobile tendono ad avere scarsa tolleranza nei confronti dei messaggi SMS non sollecitati. Le campagne SMS non sollecitate otterranno quasi sempre tassi di risposta bassi e, di conseguenza, un ritorno sull'investimento ridotto.

Gli operatori di telefonia mobile, inoltre, controllano continuamente i mittenti di SMS in blocco e limitano o bloccano i messaggi provenienti da numeri che sono stati identificati come mittenti di messaggi non sollecitati.

L'invio di contenuti non sollecitati costituisce anche una violazione dell'Acceptable Use Policy (policy di utilizzo accettabile) di AWS. Il team di Amazon SNS controlla regolarmente le campagne SMS e potrebbe limitare o bloccare la possibilità di inviare messaggi se risulta che vengono inviati messaggi indesiderati.

In molti paesi, regioni e giurisdizioni, infine, sono previste sanzioni severe per l'invio di messaggi SMS non sollecitati. Ad esempio, negli Stati Uniti, il Telephone Consumer Protection Act (TCPA) stabilisce che i consumatori hanno diritto a $500-$1.500 in danni (pagati dal mittente) per ogni messaggio non richiesto che ricevono.

Questa sezione illustra diverse best practice che potrebbero rivelarsi utili per migliorare il coinvolgimento dei clienti ed evitare costose sanzioni. Non contiene tuttavia consulenza legale. Consulta sempre un avvocato per ottenere adeguati pareri legali.

Argomenti

- Conformità a leggi, normative e requisiti del gestore

- Acquisizione dell'autorizzazione
- Non inviare messaggi a elenchi obsoleti
- Controllo degli elenchi dei clienti
- Conservazione della documentazione
- Rendi i tuoi messaggi chiari, attendibili e concisi
- Invio di risposte appropriate
- Adattamento dell'invio al coinvolgimento
- Invio in orari appropriati
- Astensione dalla ripetizione in più canali
- Utilizzo di codici brevi dedicati
- Verifica i numeri di telefono di destinazione
- Progetta considerando la ridondanza
- Limiti e restrizioni degli SMS
- Gestione delle parole chiave di esclusione
- CreatePool
- PutKeyword
- Gestione delle impostazioni del numero
- Limiti relativi ai caratteri per gli SMS in Amazon SNS

Conformità a leggi, normative e requisiti del gestore

La violazione di leggi e normative in vigore nei luoghi di residenza dei clienti può comportare multe e sanzioni elevate. Per questo motivo, è essenziale conoscere le leggi relative alla messaggistica SMS in vigore in ogni paese o regione in cui l'azienda opera.

L'elenco seguente contiene link alle leggi chiave applicate alle comunicazioni tramite SMS nei principali mercati di tutto il mondo.

- Stati Uniti: alcuni tipi di messaggi SMS sono soggetti al Telephone Consumer Protection Act (TCPA) del 1991. Per ulteriori informazioni, consulta le regole e le normative nel sito Web della Federal Communications Commission.
- Regno Unito: alcuni tipi di messaggi SMS sono soggetti alla direttiva CE relativa alla vita privata e alle comunicazioni elettroniche del 2003. Per ulteriori informazioni, consulta What are PECR? nel sito Web dell'Information Commissioner's Office del Regno Unito.

- Unione Europea: alcuni tipi di messaggi SMS sono soggetti alla direttiva relativa alla vita privata e alle comunicazioni elettroniche del 2002, detta anche "direttiva ePrivacy". Per ulteriori informazioni, consulta il testo integrale della legge nel sito Web europa.eu.
- Canada: alcuni tipi di messaggi SMS sono soggetti al Fighting Internet and Wireless Spam Act, comunemente noto come CASL (Canada's Anti-Spam Law). Per ulteriori informazioni, consulta il testo integrale della legge nel sito Web del parlamento canadese.
- Giappone: alcuni tipi di messaggi SMS sono soggetti alla legge sulla regolamentazione della trasmissione di posta elettronica specifica. Per ulteriori informazioni, consulta Japan's Countermeasures Against Spam nel sito Web del Ministero degli Affari interni e delle Comunicazioni del Giappone.

In qualità di mittente, queste leggi possono essere applicabili anche se l'azienda o l'organizzazione non ha sede in uno di questi Paesi. Alcune delle leggi dell'elenco sono state originariamente create per e-mail o chiamate telefoniche non sollecitate, ma sono state interpretate o estese in modo da coprire anche i messaggi SMS. Altri paesi e regioni potrebbero disporre di legislazione specifica relativa alla trasmissione di messaggi SMS. Consulta un avvocato in ogni paese o regione in cui si trovano i tuoi clienti per ottenere consulenza legale.

In molti Paesi, i gestori locali hanno sostanzialmente l'autorità di determinare il tipo di traffico gestito nelle rispettive reti. Ciò significa che i gestori potrebbero applicare limitazioni ai contenuti SMS che non risultano conformi ai requisiti minimi delle leggi locali.

Acquisizione dell'autorizzazione

Non inviare mai messaggi a destinatari che non hanno richiesto esplicitamente di ricevere i tipi specifici di messaggi che intendi inviare. Non condividere elenchi di consenso esplicito, nemmeno tra organizzazioni all'interno della stessa azienda.

Se i destinatari possono registrarsi per ricevere i messaggi mediante un modulo online, aggiungi sistemi che impediscono che script automatizzati possano eseguire la sottoscrizione all'insaputa dei destinatari. È inoltre necessario limitare il numero di volte in cui un utente può inviare un numero di telefono in una singola sessione.

Quando ricevi una richiesta con consenso esplicito per i messaggi SMS, invia al destinatario un messaggio per chiedere conferma del desiderio di ricevere tali messaggi. Non inviare altri messaggi al destinatario finché non conferma la propria sottoscrizione. Un messaggio per la conferma della sottoscrizione potrebbe essere simile all'esempio seguente:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

Mantieni una documentazione della data, dell'ora e dell'origine di ogni richiesta con consenso esplicito e di ogni conferma. Potrebbe essere utile se un operatore o un ente normativo ne fa richiesta, nonché per eseguire i controlli di routine dell'elenco dei clienti.

Flusso di lavoro del consenso esplicito

In alcuni casi, ad esempio la registrazione mediante numeri verdi gratuiti o codici brevi negli Stati Uniti, i gestori di telefonia mobile richiedono di fornire modelli (mockup) o schermate dell'intero flusso di lavoro di consenso esplicito. I modelli o le schermate devono assomigliare il più possibile al flusso di lavoro del consenso esplicito che i destinatari completeranno.

I modelli o le schermate devono includere tutte le informazioni richieste elencate di seguito per mantenere il massimo livello di conformità.

Esposizione di informazioni richieste

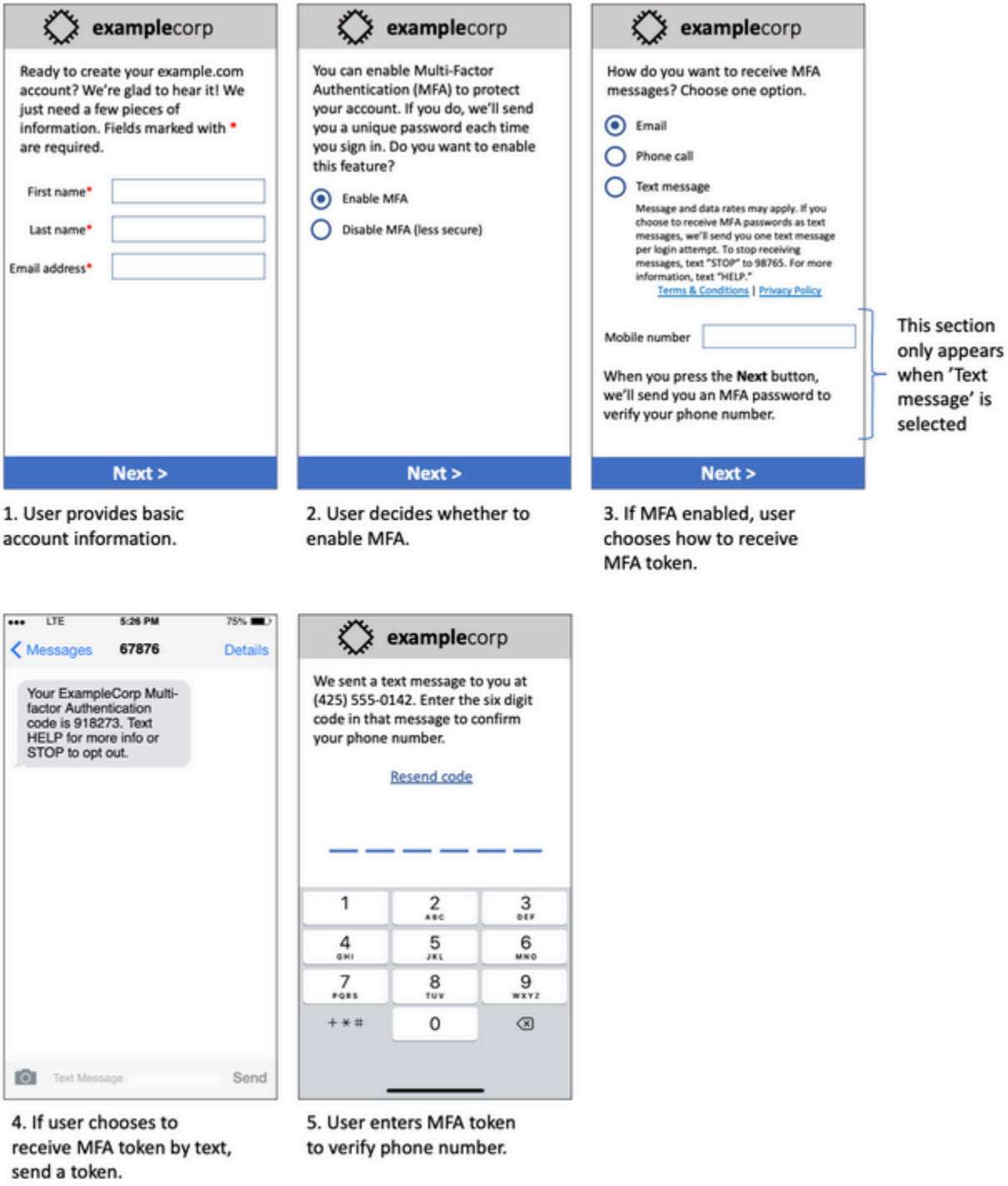
- Una descrizione del caso d'uso relativo alla messaggistica che invierai tramite il programma.
- La frase "È possibile che vengano applicate tariffe specifiche per messaggi e dati".
- Un'indicazione della frequenza con cui i destinatari riceveranno messaggi. Ad esempio, un programma di messaggistica ricorrente potrebbe indicare "un messaggio alla settimana". Un caso d'uso relativo alla password monouso o all'autenticazione a più fattori (MFA) potrebbe indicare "la frequenza dei messaggi può variare" o "un messaggio per tentativo di accesso".
- Collegamenti ai termini e alle condizioni d'uso e ai documenti relativi all'Informativa sulla privacy.

Motivi comuni di rifiuto per operazioni di consenso esplicito non conformi

- Se il nome dell'azienda fornito non corrisponde a quello fornito nel modello o nella schermata. Qualsiasi relazione non ovvia deve essere spiegata nella descrizione del flusso di lavoro del consenso esplicito.
- Se sembra che verrà inviato un messaggio al destinatario, ma non viene acquisito alcun consenso esplicito in merito. Il consenso esplicito è un requisito per tutti i messaggi.
- Se sembra che la ricezione di un messaggio SMS sia necessaria per iscriversi a un servizio. Ciò non è conforme se il flusso di lavoro non fornisce alcuna alternativa alla ricezione di un messaggio di consenso esplicito in un'altra forma, ad esempio e-mail o chiamata vocale.

- Se la lingua del consenso esplicito è rappresentata nei Termini del servizio. Le informazioni richieste devono sempre essere esposte al destinatario al momento del rilascio del consenso esplicito anziché essere contenute in un documento collegato relativo alla policy.
- Se un cliente ha fornito il consenso a ricevere un tipo di messaggio da te e tu gli invii altri tipi di messaggi di testo. Ad esempio, il cliente accetta di ricevere password monouso, ma riceve anche messaggi relativi a sondaggi e indagini.
- Se le informazioni richieste (elencate sopra) non vengono esposte ai destinatari.

L'esempio seguente è conforme ai requisiti dei gestori di telefonia mobile per un caso d'uso relativo all'autenticazione a più fattori.



Modello del caso d'uso relativo all'autenticazione a più fattori

Contiene testo e immagini finalizzati e mostra l'intero flusso di lavoro del consenso esplicito, completo di annotazioni. Nel flusso di lavoro del consenso esplicito, il cliente deve intraprendere azioni

distinte e intenzionali per fornire il proprio consenso a ricevere messaggi di testo e contiene tutte le informazioni richieste.

Altri tipi di flussi di lavoro del consenso esplicito

I gestori di telefonia mobile accetteranno anche flussi di lavoro del consenso esplicito esterni ad applicazioni e siti Web, come il consenso esplicito verbale o scritto, se conforme a quanto descritto sopra. Un flusso di lavoro del consenso esplicito conforme e uno script verbale o scritto acquisirà il consenso esplicito del destinatario a ricevere un tipo di messaggio specifico. Esempi di ciò includono lo script verbale utilizzato da un agente del supporto per acquisire il consenso prima della registrazione in un database di servizi o un numero di telefono elencato in un volantino promozionale. Per fornire un modello di questi tipi di flusso di lavoro del consenso esplicito, puoi fornire una schermata dello script di conenso, del materiale di marketing o del database in cui vengono raccolti i numeri. I gestori di telefonia mobile potrebbero avere ulteriori domande su questi casi d'uso se l'opzione di consenso esplicito non è chiara o se il caso d'uso supera determinati volumi.

Non inviare messaggi a elenchi obsoleti

Le persone cambiano spesso numeri di telefono. Un numero di telefono per il quale è stato acquisito il consenso all'invio di messaggi due anni fa potrebbe ora appartenere a un altro utente. Non utilizzare un vecchio elenco di numeri di telefono per un nuovo programma di messaggistica; se lo fai, è possibile che alcuni messaggi non vengano recapitati perché il numero non è più operativo e alcuni destinatari potrebbero non ricordare di avere precedentemente fornito il loro consenso.

Controllo degli elenchi dei clienti

Se invii campagne SMS ricorrenti, controlla regolarmente gli elenchi dei clienti. Tale controllo garantisce che i messaggi vengano inviati solo ai clienti che sono interessati a riceverli.

Quando controlli l'elenco, invia a ogni cliente che ha acconsentito esplicitamente un messaggio di promemoria della sottoscrizione con le informazioni per annullarla. Un messaggio di promemoria potrebbe essere simile all'esempio seguente:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

Conservazione della documentazione

Conserva la documentazione che indica quando ogni cliente ha richiesto di ricevere SMS e quali messaggi hai inviato a ciascun cliente. Molti paesi e regioni di tutto il mondo richiedono ai mittenti di

SMS di conservare tale documentazione in modo che sia facilmente recuperabile. Tali informazioni potrebbero inoltre essere richieste dagli operatori di telefonia mobile in qualsiasi momento. Le informazioni esatte da fornire variano a seconda del paese o della regione. Per ulteriori informazioni sui requisiti di conservazione della documentazione, esamina le normative sui messaggi SMS commerciali di ogni paese o regione in cui si trovano i tuoi clienti.

Talvolta, un operatore o un ente normativo ci chiede di fornire una prova del fatto che un cliente ha acconsentito a ricevere messaggi da te. In questi casi, AWS Support ti contatta con un elenco delle informazioni richieste dall'operatore o dall'ente. Se non sei in grado di fornire le informazioni necessarie, potremmo sospendere la tua possibilità di inviare ulteriori messaggi SMS.

Rendi i tuoi messaggi chiari, attendibili e concisi

Gli SMS sono uno strumento unico. Il limite di 160 caratteri per messaggio significa che i messaggi devono essere concisi. Le tecniche che potresti utilizzare in altri canali di comunicazione, come la posta elettronica, potrebbero non essere applicabili al canale SMS e potrebbero persino sembrare non attendibili o ingannevoli se utilizzate con i messaggi SMS. Se il contenuto dei messaggi non è in linea con le best practice, i destinatari potrebbero ignorarli; nel peggiore dei casi, i gestori di telefonia mobile potrebbero identificare i messaggi come spam e bloccare i messaggi futuri ricevuti dal tuo numero di telefono.

Questa sezione fornisce alcuni suggerimenti e idee per creare un corpo efficace del messaggio SMS.

Identificati come mittente

I destinatari devono essere in grado di capire subito che un messaggio proviene da te. I mittenti che si attengono a questa best practice includono un identificativo ("nome del programma") all'inizio di ogni messaggio.

Non fare questo:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Prova invece questo:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

Non cercare di far sembrare il tuo messaggio un messaggio personale

Alcuni esperti di marketing sono tentati di aggiungere un tocco personale ai loro messaggi SMS facendoli sembrare inviati da un individuo. Tuttavia, questa tecnica potrebbe essere fraintesa come un tentativo di phishing.

Non fare questo:

```
Hi, this is Jane. Did you know that you can save up to 50% at
Example.com? Click here for more info: https://www.example.com.
```

Prova invece questo:

```
ExampleCorp Offers: Save 25-50% on sale items at Example.com. Click here
to browse the sale: https://www.example.com. Text STOP to opt-out.
```

Fai attenzione quando parli di soldi

I truffatori spesso sfruttano il desiderio delle persone di risparmiare e ricevere denaro. Non far sembrare le offerte troppo belle per essere vere. Non usate il richiamo dei soldi per ingannare le persone. Non utilizzare simboli di valuta per fare riferimento al denaro.

Non fare questo:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Prova invece questo:

```
ExampleCorp Offers: Your ExampleCorp insurance policy gets you discounts
at 2300+ repair shops nationwide. More info at https://www.example.com.
Text STOP to opt-out.
```

Usa solo i caratteri necessari

I marchi sono spesso propensi a proteggere la propria immagine includendo simboli come ™ o ® nei rispettivi messaggi. Tuttavia, questi simboli non fanno parte del set standard di caratteri (noto come alfabeto GSM) che può essere incluso in un messaggio SMS di 160 caratteri. Quando invii un messaggio contenente uno di questi caratteri, il messaggio viene inviato automaticamente utilizzando un sistema di codifica dei caratteri diverso, che supporta solo 70 caratteri per parte del messaggio. Di

conseguenza, il messaggio potrebbe venire suddiviso in più parti. Poiché ti viene addebitato un costo per ogni parte del messaggio inviata, l'invio dell'intero messaggio potrebbe costarti più del previsto. Inoltre, i destinatari potrebbero ricevere più messaggi sequenziali, anziché un unico messaggio. Per ulteriori informazioni sulla codifica dei caratteri degli SMS, consulta Limiti relativi ai caratteri per gli SMS in Amazon SNS.

Non fare questo:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Prova invece questo:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> I due esempi precedenti sono pressoché identici, ma il primo contiene il simbolo del marchio registrato (®), che non fa parte dell'alfabeto GSM. Di conseguenza, il primo esempio viene inviato come messaggio in due parti, mentre il secondo esempio viene inviato come messaggio unico.

Usa collegamenti validi e sicuri

Se il messaggio include link, ricontrolla i link per assicurarti che funzionino. Verifica i link su un dispositivo esterno alla rete aziendale per assicurarti che vengano risolti correttamente. A causa del limite di 160 caratteri dei messaggi SMS, è possibile suddividere URL molto lunghi su più messaggi. È necessario utilizzare i domini di reindirizzamento per fornire URL abbreviati. Tuttavia, non dovresti usare servizi gratuiti di accorciamento dei link, come tinyurl.com o bitly.com, perché i gestori tendono a filtrare i messaggi che includono link su questi domini. Puoi tuttavia utilizzare servizi di abbreviazione dei link a pagamento, purché i tuoi link rimandino a un dominio dedicato all'uso esclusivo della tua azienda o organizzazione.

Non fare questo:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Prova invece questo:

```
ExampleCorp Offers: Today only, get an exclusive deal on an ExampleCorp
Widget. See https://a.co/cFKmaRG for more info. Text STOP to opt-out.
```

Limita il numero di abbreviazioni da utilizzare

La limitazione di 160 caratteri del canale SMS induce alcuni mittenti a credere di dover utilizzare ampiamente le abbreviazioni nei loro messaggi. Tuttavia, l'uso eccessivo di abbreviazioni può sembrare poco professionale per molti lettori e potrebbe indurre alcuni utenti a segnalare il tuo messaggio come spam. È possibile scrivere un messaggio coerente senza utilizzare un numero eccessivo di abbreviazioni.

Non fare questo:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Prova invece questo:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Invio di risposte appropriate

Quando un destinatario risponde ai tuoi messaggi, assicurati di rispondere con informazioni utili. Quando un cliente risponde a uno dei tuoi messaggi con la parola chiave "HELP", ad esempio, invia informazioni sul programma a cui ha effettuato la sottoscrizione, sul numero di messaggi che invierai ogni mese e sui modi in cui potrà contattarti per ulteriori informazioni. Una risposta HELP potrebbe essere simile all'esempio seguente:

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Quando un cliente risponde con la parola chiave "STOP", comunica al cliente che non riceverà ulteriori messaggi. Una risposta STOP potrebbe essere simile all'esempio seguente:

```
You're unsubscribed from ExampleCorp alerts. No more messages will be sent.
Reply HELP, email help@example.com, or call 425-555-0199 for more info.
```

Adattamento dell'invio al coinvolgimento

Le priorità dei clienti possono cambiare nel tempo. Se i clienti non ritengono più utili i tuoi messaggi, potrebbero cancellarsi completamente dalla ricezione o addirittura segnalare i tuoi messaggi come non sollecitati. Per questi motivi, è importante adattare le tue procedure di invio al coinvolgimento dei clienti.

Per i clienti che raramente interagiscono con i tuoi messaggi, dovresti adattare la frequenza dei messaggi. Se ai clienti coinvolti invii messaggi settimanali, ad esempio, potresti creare un riepilogo mensile separato per i clienti meno coinvolti.

Rimuovi infine dai tuoi elenchi i clienti che non sono affatto coinvolti. Questo evita che i tuoi messaggi generino frustrazione nei clienti e ti consente inoltre di risparmiare denaro e proteggere la tua reputazione come mittente.

Invio in orari appropriati

Invia messaggi solo durante il normale orario di ufficio diurno. Se invii messaggi all'ora di cena o nel cuore della notte, è probabile che i clienti annullino la sottoscrizione ai tuoi elenchi per evitare di essere disturbati. Non è inoltre opportuno inviare messaggi SMS quando i tuoi clienti non possono rispondere immediatamente.

Se invii campagne o percorsi a un pubblico molto vasto, ricontrolla la velocità di trasmissione effettiva dei numeri di origine. Dividi il numero di destinatari per la velocità di trasmissione effettiva per determinare il tempo necessario per inviare messaggi a tutti i destinatari.

Astensione dalla ripetizione in più canali

Se nelle tue campagne utilizzi più canali di comunicazione (ad esempio e-mail, SMS e notifiche push), non inviare lo stesso messaggio in ogni canale. Se invii lo stesso messaggio tramite più canali contemporaneamente, il tuo comportamento di invio verrà probabilmente percepito dai clienti come fastidioso anziché utile.

Utilizzo di codici brevi dedicati

Se utilizzi codici brevi, mantieni un codice breve separato per ogni marchio e per ciascun tipo di messaggio. Se l'azienda possiede due marchi, ad esempio, utilizza un codice breve separato per ciascuno. Analogamente, se invii messaggi sia transazionali che promozionali, utilizza un codice breve separato per ciascun tipo di messaggio. Per ulteriori informazioni sulla richiesta di codici brevi, consulta Richiesta di codici brevi dedicati per la messaggistica SMS con Amazon SNS.

Verifica i numeri di telefono di destinazione

Quando invii messaggi SMS tramite Amazon SNS, ti viene addebitato un costo per ogni parte di messaggio inviata. Il prezzo da pagare per parte di messaggio varia in base al Paese o all'area geografica del destinatario. Per le informazioni sui prezzi dei messaggi SMS, consulta la pagina dei prezzi di Amazon SNS.

Quando Amazon SNS accetta una richiesta di invio di un messaggio SMS, come risultato di una chiamata all'API SendMessages o come risultato del lancio di una campagna o di un percorso, ti viene addebitato un costo per l'invio di tale messaggio. Questa affermazione è vera anche se il destinatario previsto non riceve effettivamente il messaggio. Ad esempio, se il numero di telefono del destinatario non è più operativo o se il numero a cui hai inviato il messaggio non è un numero di cellulare valido, ti verrà comunque addebitato il costo dell'invio del messaggio.

Amazon SNS accetta richieste valide di invio di messaggi SMS e tenta di recapitarli. Per questo motivo, è necessario verificare che i numeri di telefono a cui si inviano messaggi siano numeri di cellulare validi. Puoi utilizzare il servizio di convalida dei numeri di telefono di Amazon SNS per determinare se un numero di telefono è valido e di che tipo è (cellulare, fisso o VoIP). Per ulteriori informazioni, consulta Convalida dei numeri di telefono in Amazon Pinpoint nella Guida per gli sviluppatori di Amazon Pinpoint.

Progetta considerando la ridondanza

Per i programmi di messaggistica mission critical, ti consigliamo di configurare Amazon SNS in più di una Regione AWS. Amazon SES è disponibile in diverse Regioni AWS. Per un elenco delle regioni in cui Amazon SNS è disponibile, consulta Riferimenti generali di AWS.

I numeri di telefono utilizzati per i messaggi SMS, inclusi codici brevi, codici lunghi, numeri gratuiti e numeri 10DLC, non possono essere replicati nelle Regioni AWS. Di conseguenza, per utilizzare Amazon SNS in più regioni, devi richiedere numeri di telefono distinti in ogni regione in cui desideri utilizzare Amazon SNS. Ad esempio, se utilizzi un codice breve per inviare messaggi SMS a destinatari negli Stati Uniti, devi richiedere codici brevi distinti in ciascuna Regione AWS che prevedi di utilizzare.

In alcuni Paesi, puoi anche utilizzare più tipi di numeri di telefono per una maggiore ridondanza. Ad esempio, negli Stati Uniti, puoi richiedere codici brevi, numeri 10DLC e numeri gratuiti. Ciascuno di questi tipi di numeri di telefono segue un percorso diverso per raggiungere il destinatario. La disponibilità di più tipi di numeri di telefono disponibili, nella stessa Regione AWS o su più Regioni AWS, fornisce un ulteriore livello di ridondanza, che può contribuire a migliorare la resilienza.

Limiti e restrizioni degli SMS

Per i limiti e le restrizioni relativi agli SMS, consulta i limiti e le restrizioni relativi agli SMS in Amazon Pinpoint nella Guida per l'utente di Amazon Pinpoint.

Gestione delle parole chiave di esclusione

I destinatari degli SMS possono utilizzare i propri dispositivi per disattivare i messaggi rispondendo con una parola chiave. Per ulteriori informazioni, consulta Richiesta di non ricevere i messaggi SMS.

CreatePool

Utilizza l'azione API `CreatePool` per creare un nuovo pool e associare un'identità di origine specificata al pool. Per ulteriori informazioni, consulta CreatePool nell'API SMS e vocale di Amazon Pinpoint.

PutKeyword

Usa l'azione API `PutKeyword` per creare o aggiornare una configurazione di parole chiave su un pool o un numero di telefono di emissione. Per ulteriori informazioni, consulta PutKeyword nell'API SMS e vocale di Amazon Pinpoint.

Gestione delle impostazioni del numero

Puoi utilizzare le opzioni nella sezione Impostazioni del numero della pagina delle impostazioni SMS e voce per gestire le impostazioni per i codici brevi e i codici lunghi dedicati che sono stati richiesti da AWS Support e assegnati all'account. Per ulteriori informazioni, consulta Gestione dlele impostazioni del numero nella Guida per l'utente di Amazon Pinpoint.

Limiti relativi ai caratteri per gli SMS in Amazon SNS

Un singolo messaggio SMS può contenere fino a 140 byte di informazioni. Il numero di caratteri che può essere incluso in un singolo messaggio SMS dipende dal tipo di caratteri che il messaggio contiene.

Se utilizza solo caratteri del set di caratteri GSM 03.38, noto anche come alfabeto GSM a 7 bit, il messaggio può contenere fino a 160 caratteri. Se il messaggio contiene caratteri non appartenenti al set di caratteri GSM 03.38, il messaggio può includere fino a 70 caratteri. Quando si invia un messaggio SMS, Amazon SNS determina automaticamente la codifica più efficiente da utilizzare.

Quando un messaggio contiene più caratteri del numero massimo, il messaggio viene suddiviso in più parti. Quando i messaggi vengono suddivisi in più parti, ogni parte contiene informazioni aggiuntive sulla parte del messaggio che lo precede. Quando il dispositivo del destinatario riceve parti del messaggio separate in questo modo, utilizza queste informazioni aggiuntive per garantire che tutte le parti del messaggio vengano visualizzate nell'ordine corretto. A seconda dell'operatore e del dispositivo mobile del destinatario, è possibile che più messaggi vengano visualizzati come un singolo messaggio o come una sequenza di messaggi separati. Di conseguenza, il numero di caratteri per ogni parte di un messaggio viene ridotto a 153 (per i messaggi contenenti solo caratteri GSM 03.38) o a 67 (per i messaggi contenenti altri caratteri). È possibile stimare quante parti di messaggio contiene il messaggio prima di inviarlo utilizzando gli strumenti di calcolo della lunghezza degli SMS, molti dei quali sono disponibili online. La dimensione massima supportata di qualsiasi messaggio è 1.600 caratteri GSM o 630 caratteri non GSM. Per ulteriori informazioni sul throughput (velocità di trasmissione effettiva) e sulla dimensione dei messaggi, consulta la pagina relativa ai limiti di caratteri per gli SMS in Amazon Pinpoint nella guida per l'utente di Amazon Pinpoint.

Per visualizzare il numero di parti per ogni messaggio inviato, è innanzitutto necessario abilitare le impostazioni relative allo streaming di eventi. Quando si esegue questa operazione, Amazon SNS produce un evento `_SMS.SUCCESS` quando il messaggio viene recapitato al fornitore di servizi mobili del destinatario. Il record di evento `_SMS.SUCCESS` contiene un attributo denominato `attributes.number_of_message_parts`. Questo attributo specifica il numero di parti di messaggio contenute nel messaggio.

> **Important**
>
> Quando si invia un messaggio che contiene più di una parte, viene addebitato il numero di parti contenute nel messaggio.

Set di caratteri GSM 03.38

Nella tabella seguente sono elencati tutti i caratteri presenti nel set di caratteri GSM 03.38. Se invii un messaggio che include solo i caratteri riportati nella tabella seguente, il messaggio può contenere fino a 160 caratteri.

Caratteri standard GSM 03.38												
A	B	C	D	E	F	G	H	I	J	K	L	M

Caratteri standard GSM 03.38												
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

In aggiunta ai caratteri riportati nella tabella precedente, il set di caratteri GSM 03.38 include diversi simboli. Ognuno di essi, tuttavia, viene conteggiato come due caratteri poiché include anche un carattere di escape non visibile:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Il set di caratteri GSM 03.38 include, infine, anche i caratteri non stampati seguenti:

- Carattere di spazio

- Controllo di avanzamento riga, che indica la fine di una riga di testo e l'inizio di un'altra
- Controllo di ritorno a capo, che passa all'inizio di una riga di testo (in genere dopo un carattere di avanzamento riga)
- Controllo di escape, che viene aggiunto automaticamente ai caratteri dell'elenco precedente

Messaggi di esempio

Questa sezione contiene vari messaggi SMS di esempio. Per ogni esempio, questa sezione mostra il numero totale di caratteri e il numero di parti del messaggio.

Esempio 1: messaggio lungo che contiene solo caratteri presenti nell'alfabeto GSM 03.38

Il seguente messaggio contiene solo caratteri presenti nell'alfabeto GSM 03.38.

```
Hello Carlos. Your Example Corp. bill of $100 is now available. Autopay is scheduled for next Thursday, April 9. To view the details of your bill, go to https://example.com/bill1.
```

Il messaggio precedente contiene 180 caratteri, quindi deve essere diviso in più parti. Quando un messaggio è diviso in più parti, ogni parte può contenere 153 caratteri GSM 03.38. Di conseguenza, questo messaggio viene inviato come messaggio in due parti.

Esempio 2: un messaggio che contiene caratteri multibyte

Il seguente messaggio contiene diversi caratteri cinesi, tutti al di fuori dell'alfabeto GSM 03.38.

```
##################################################·####1994#7#########
```

Il messaggio precedente contiene 71 caratteri. Tuttavia, poiché quasi tutti i caratteri nel messaggio sono al di fuori dell'alfabeto GSM 03.38, viene inviato come messaggio in due parti. Ciascuna di queste parti può contenere un massimo di 67 caratteri.

Esempio 3: messaggio che contiene un singolo carattere non GSM

Il seguente messaggio contiene un singolo carattere che non fa parte dell'alfabeto GSM 03.38. In questo esempio, il carattere è una virgoletta singola di chiusura (’), che è un carattere diverso da un normale apostrofo ('). Le applicazioni di elaborazione testi come Microsoft Word spesso sostituiscono automaticamente gli apostrofi con le virgolette singole di chiusura. Se si redigono messaggi SMS in Microsoft Word e si incollano in Amazon SNS, è necessario rimuovere questi caratteri speciali e sostituirli con apostrofi.

```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

Il messaggio precedente contiene 130 caratteri. Tuttavia, poiché contiene il carattere di virgoletta singola di chiusura, che non fa parte dell'alfabeto GSM 03.38, viene inviato come messaggio in due parti.

Se si sostituisce il carattere di virgoletta singola di chiusura in questo messaggio con un apostrofo (che fa parte dell'alfabeto GSM 03.38), il messaggio viene inviato come un messaggio singolo.

Notifiche push per dispositivi mobili

Con Amazon SNS hai la possibilità di inviare messaggi di notifica push direttamente alle app sui dispositivi mobili. I messaggi di notifica push inviati a un endpoint mobile possono essere visualizzati nell'app per dispositivi mobili come messaggi di avviso, aggiornamenti di notifiche o anche avvisi sonori.

Argomenti

Funzionamento delle notifiche all'utente

L'invio di messaggi di notifica push a desktop e dispositivi mobili viene eseguito mediante uno dei seguenti servizi di notifica push supportati:

- Amazon Device Messaging (ADM)
- Apple Push Notification Service (APNs) per iOS e Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Servizio di notifica push Microsoft per Windows Phone (MPNS)
- Windows Push Notification Services (WNS)

I servizi di notifica push, come APN e FCM, mantengono una connessione con ogni app e dispositivo mobile associato registrato per utilizzare il servizio. Quando si registra un'app e un dispositivo mobile, la notifica push restituisce un token di dispositivo. Amazon SNS utilizza il token di dispositivo per creare un endpoint mobile a cui può inviare direttamente messaggi di notifica push. Per consentire la comunicazione tra Amazon SNS e i differenti servizi di notifica push, devi inviare le tue credenziali del servizio di notifica push ad Amazon SNS affinché le utilizzi a tuo nome. Per ulteriori informazioni, consulta Panoramica del processo di notifica utente.

Oltre a inviare direttamente messaggi di notifica push, puoi anche utilizzare Amazon SNS per inviare messaggi a endpoint mobili che dispongono di una sottoscrizione a un argomento. Il concetto è lo stesso della sottoscrizione di altri tipi di endpoint, ad esempio Amazon SQS, HTTP/S, e-mail e SMS, a un argomento, come descritto in Che cos'è Amazon SNS?. La differenza è che Amazon SNS comunica mediante i servizi di notifica push affinché gli endpoint mobili con sottoscrizione ricevano i messaggi di notifica push inviati all'argomento.

Panoramica del processo di notifica utente

1. Ottenere le credenziali e il token del dispositivo per le piattaforme mobili che si desidera supportare.
2. Utilizzare le credenziali per creare un oggetto application platform (`PlatformApplicationArn`) utilizzando Amazon SNS. Per ulteriori informazioni, consulta Creazione di un'applicazione di piattaforma.
3. Utilizza le credenziali ottenute per richiedere un token per il dispositivo mobile e l'app dal servizio di notifiche push. Il token che ottieni rappresenta il tuo dispositivo e la tua app per dispositivi mobili.
4. Utilizzare il token dispositivo e `PlatformApplicationArn` per creare un oggetto endpoint piattaforma (`EndpointArn`) utilizzando Amazon SNS. Per ulteriori informazioni, consulta Creazione di un endpoint di piattaforma.

5. `EndpointArn` viene quindi utilizzato per pubblicare un messaggio in una app in un dispositivo mobile. Per ulteriori informazioni, consulta Pubblicazione in un dispositivo mobile e API Publish (Pubblica) nella Guida API di Amazon Simple Notification Service.

Configurare un'app mobile

Questa sezione descrive come utilizzare AWS Management Console le informazioni descritte in Prerequisiti per le notifiche utente Amazon SNS per configurare le applicazioni mobili.

Argomenti

Prerequisiti per le notifiche utente Amazon SNS

Per iniziare a utilizzare le notifiche push per dispositivi mobili Amazon SNS, è necessario quanto segue:

- Un set di credenziali per la connessione a uno dei servizi di notifica push supportati, ovvero ADM, APN, Baidu, FCM, MPNS, or WNS.
- Un token di dispositivo o un ID di registrazione per l'app per dispositivi mobili e il dispositivo.
- Amazon SNS configurato per l'invio di messaggi di notifica push a endpoint mobili.
- Un app per dispositivi mobili registrata e configurata per l'utilizzo di uno dei servizi di notifica push supportati.

La registrazione dell'applicazione a un servizio di notifica mobile comporta varie fasi. Amazon SNS necessita di alcune delle informazioni fornite al servizio di notifica push per inviare direttamente messaggi di notifica push all'endpoint mobile. In genere, hai bisogno delle credenziali necessarie per la connessione al servizio di notifica push, il token di dispositivo o l'ID di registrazione (che

rappresenta il dispositivo mobile e l'app) ricevuti dal servizio di notifica push e l'app per dispositivi mobili registrata al servizio di notifica push.

La forma esatta delle credenziali differisce a seconda della piattaforma mobile, ma in ogni caso, queste credenziali devono essere fornite durante una connessione alla piattaforma. Un set di credenziali viene generato per ogni app per dispositivi mobili e deve essere utilizzato per inviare un messaggio a qualsiasi istanza di quell'app.

I nomi specifici variano a seconda del servizio di notifica push utilizzato. Ad esempio, quando si utilizzano APN come servizio di notifica push, si necessita di un token di dispositivo. Quando invece utilizzi FCM, il token di dispositivo equivalente è denominato ID di registrazione. Il token di dispositivo o l'ID di registrazione è una stringa inviata all'applicazione dal sistema operativo del dispositivo mobile. Identifica in modo univoco un'istanza di un'app per dispositivi mobili eseguita su un determinato dispositivo mobile e può essere considerata come l'identificatore univoco di questa coppia app-dispositivo.

Amazon SNS archivia le credenziali (e altre impostazioni) come risorsa di applicazione di piattaforma. I token di dispositivo (anche in questo caso con altre impostazioni) sono rappresentati come oggetti denominati endpoint di piattaforma. Ogni endpoint di piattaforma appartiene a una specifica applicazione di piattaforma ed è possibile comunicare con ognuno di questi endpoint utilizzando le credenziali archiviate nell'applicazione di piattaforma corrispondente.

Le sezioni seguenti includono i prerequisiti per ogni servizio di notifica push supportato. Dopo aver ottenuto le informazioni sui prerequisiti, puoi inviare un messaggio di notifica push utilizzando la AWS Management Console o le API push per dispositivi mobili di Amazon SNS. Per ulteriori informazioni, consultare Panoramica del processo di notifica utente.

Creazione di un'applicazione di piattaforma

Per permettere a Amazon SNS di inviare messaggi di notifica agli endpoint mobili, direttamente o tramite sottoscrizioni a un argomento, devi prima creare un'applicazione della piattaforma. Dopo aver registrato l'app con AWS, la fase successiva è quella di creare un endpoint per l'app e il dispositivo mobile. Amazon SNS utilizza quindi l'endpoint per inviare i messaggi di notifica all'app e al dispositivo.

Per creare un'applicazione della piattaforma

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegli Mobile (Dispositivi mobili), e quindi scegli Push notifications (Notifiche push).

3. Nella sezione Platform applications (Applicazioni di piattaforma), scegli Create platform application (Crea applicazione di piattaforma).

   Per un elenco delle regioni AWS in cui è possibile creare applicazioni per dispositivi mobili, consulta Regioni supportate per applicazioni mobili.

4. Per Application name (Nome applicazione), inserisci un nome per rappresentare l'app.

   I nomi delle app devono essere composti unicamente da lettere ASCII maiuscole e minuscole, numeri, caratteri di sottolineatura, trattini e punti. Anche i nomi dovono essere 1–256 caratteri lunghi.

5. Per Push notification platform (Piattaforma notifiche push), scegli la piattaforma con cui l'app è registrata e inserisci le credenziali appropriate.

   > **Note**
   >
   > Se stai usando una delle piattaforme Apple Push Notification Service (Servizio notifiche push Apple) (APN), puoi scegliere fra l'autenticazione basata sul token o sul certificato, quindi scegli Choose file (Scegli file) per caricare il file .p8 o .p12 (esportato da Keychain Access) su Amazon SNS.

6. Selezionare Create platform application (Crea applicazione di piattaforma).

   In tal modo l'app viene registrata con Amazon SNS e viene creato un oggetto applicazione piattaforma per la piattaforma selezionata e quindi restituito un elemento PlatformApplicationArn corrispondente.

Creazione di un endpoint di piattaforma

Quando si esegue la registrazione di un'app e di un dispositivo mobile a un servizio di notifiche push, la notifica push restituisce un token di dispositivo. Amazon SNS utilizza il token di dispositivo per creare un endpoint mobile a cui può inviare direttamente messaggi di notifica push. Per ulteriori informazioni, consulta Prerequisiti per le notifiche utente Amazon SNS e Panoramica del processo di notifica utente.

Questa sezione descrive l'approccio consigliato per creare un endpoint di piattaforma.

Argomenti

- Creazione di un endpoint di piattaforma

- Pseudocodice
- AWS Esempio di SDK
- Risoluzione dei problemi

Creazione di un endpoint di piattaforma

Per inviare notifiche push a un'app con Amazon SNS, è necessario prima registrare il token di dispositivo di quell'app in Amazon SNS richiamando l'operazione di creazione di endpoint di piattaforma. Questa azione utilizza l'Amazon Resource Name (ARN) dell'applicazione di piattaforma e del token di dispositivo come parametri e restituisce l'ARN dell'endpoint di piattaforma creato.

L'`CreatePlatformEndpoint`azione esegue le seguenti operazioni:

- Se l'endpoint di piattaforma esiste, non lo crea di nuovo. Restituisce all'intermediario l'ARN dell'endpoint di piaffatorma esistente.
- Se l'endpoint di piattaforma con lo stesso token di dispositivo ma impostazioni differenti esiste, non lo crea di nuovo. Genera un'eccezione per l'intermediario.
- Se l'endpoint di piattaforma non esiste, lo crea. Restituisci all'intermediario l'ARN dell'endpoint di piattaforma appena creato.

Non devi chiamare immediatamente l'operazione di creazione di endpoint di piattaforma a ogni avvio di un'applicazione poiché questo approccio non fornisce sempre un endpoint funzionante. Ciò può verificarsi, ad esempio, quando un'app viene disinstallata e reinstallata nello stesso dispositivo e il relativo endpoint esiste ma è disattivato. Una procedura di registrazione corretta deve garantire quanto segue:

1. L'endpoint di piattaforma esiste già per la combinazione app-dispositivo.
2. Il token di dispositivo nell'endpoint di piattaforma è il token di dispositivo valido più recente.
3. L'endpoint di piattaforma è attivato e pronto all'uso.

Pseudocodice

Lo pseudocodice seguente descrive una pratica consigliata per la creazione di un endpoint di piattaforma funzionante, corrente e attivato in un'ampia gamma di condizioni di avvio. Questo approccio funziona indipendentemente se si tratta della prima registrazione dell'app, se l'endpoint di piattaforma per l'app esiste già, se l'endpoint di piattaforma è attivato, ha il token di dispositivo

corretto e così via. Non è un problema chiamarlo più volte in successione, in quanto non creerà endpoint di piattaforma duplicati o modificherà l'endpoint di piattaforma esistente se è già aggiornato e attivato.

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
  # this is a first-time registration
  call create platform endpoint
  store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
  # the platform endpoint was deleted
  call create platform endpoint with the latest device token
  store the returned platform endpoint ARN
else
  if (the device token in the endpoint does not match the latest one) or
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

Questo approccio può essere utilizzato ogni volta che l'app vuole registrarsi o ripetere la registrazione e anche in caso di notifica della modifica di un token di dispositivo a Amazon SNS. In quest'ultimo caso, è sufficiente chiamare l'operazione con il valore di token di dispositivo più recente. Alcuni punti da considerare in relazione a questo approccio:

- Esistono due casi in cui può chiamare l'operazione di creazione di endpoint di piattaforma. All'inizio, quando l'app non conosce il relativo ARN di endpoint di piattaforma, come avviene durante una prima registrazione, oppure nel caso in cui l'operazione iniziale di ottenimento degli attributi di endpoint non riesce con un'eccezione non trovato, come avverrebbe se l'applicazione conoscesse il relativo ARN di endpoint, ma questo fosse stato eliminato.
- L'operazione di ottenimento degli attributi di endpoint viene chiamata per verificare lo stato dell'endpoint di piattaforma anche se questo è stato appena creato. Ciò avviene quando l'endpoint di piattaforma esiste già ma è disattivato. In tal caso, l'operazione di creazione di endpoint di piattaforma riesce ma non attiva l'endpoint di piattaforma, di conseguenza devi ricontrollare lo stato dell'endpoint di piattaforma prima di indicare l'operazione come riuscita.

AWS Esempio di SDK

Il codice seguente mostra come implementare lo pseudo codice precedente utilizzando i client Amazon SNS forniti dagli SDK. AWS

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

CLI

AWS CLI

Creazione di un endpoint dell'applicazione della piattaforma

Nell'esempio `create-platform-endpoint` seguente viene creato un endpoint per l'applicazione della piattaforma indicata utilizzando il token specificato.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Output:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK per Java 2.x

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Per ulteriori informazioni, consulta Operazioni API push per dispositivi mobili.

Risoluzione dei problemi

Chiamata ripetuta dell'operazione di creazione di endpoint di piattaforma con un token di dispositivo obsoleto

Soprattutto per gli endpoint FCM, potreste pensare che sia meglio archiviare il primo token del dispositivo emesso dall'applicazione e quindi richiamare l'endpoint di creazione della piattaforma con quel token del dispositivo ogni volta all'avvio dell'applicazione. Ciò può sembrare corretto in quanto l'app non deve gestire lo stato del token di dispositivo e Amazon SNS aggiorna automaticamente il token di dispositivo al valore più recente. In verità, questa soluzione presenta alcuni seri inconvenienti:

- Amazon SNS si basa sul feedback di FCM per aggiornare i token di dispositivo obsoleti a quelli nuovi. FCM conserva le informazioni sui token di dispositivo obsoleti per qualche tempo, ma non indefinitamente. Una volta che FCM dimentica la connessione tra il vecchio token di dispositivo e il nuovo, Amazon SNS non sarà più in grado di aggiornare il token archiviato nell'endpoint di piattaforma al valore corretto, ma disattiverà l'endpoint di piattaforma.
- L'applicazione di piattaforma conterrà molteplici endpoint di piattaforma corrispondenti allo stesso token di dispositivo.
- Amazon SNS impone una quota al numero di endpoint di piattaforma che è possibile creare a partire dallo stesso token di dispositivo. Alla fine, la creazione di nuovi endpoint non riuscirà con un'eccezione di parametro non valido e il messaggio di errore seguente: "This endpoint is already registered with a different token." (Questo endpoint è già registrato con un altro token.).

Per ulteriori informazioni sulla gestione degli endpoint FCM, consulta. Gestione degli endpoint Firebase Cloud Messaging (FCM)

Riattivazione di un endpoint di piattaforma associato a un token di dispositivo non valido

Quando una piattaforma mobile (come APN o FCM) informa Amazon SNS che il token di dispositivo utilizzato nella richiesta di pubblicazione non è valido, Amazon SNS disattiva l'endpoint di piattaforma associato a quel token di dispositivo. Amazon SNS rifiuterà quindi le pubblicazioni successive in quel token di dispositivo. Sebbene si possa ritenere che la soluzione migliore consista semplicemente nel riattivare l'endpoint di piattaforma e continuare a pubblicare, nella maggior parte dei casi ciò non funzionerà: i messaggi pubblicati non verranno consegnati e l'endpoint di piattaforma sarà di nuovo disattivato poco tempo dopo.

Questo perché il token di dispositivo associato all'endpoint di piattaforma è effettivamente non valido. Le consegne a tale endpoint non possono riuscire in quanto non corrisponde più ad alcuna app installata. Al momento della pubblicazione successiva, la piattaforma mobile indicherà di nuovo aD Amazon SNS che il token di dispositivo non è valido e Amazon SNS disattiverà di nuovo l'endpoint di piattaforma.

Per riattivare un endpoint di piattaforma disattivato, è necessario associarlo a un token di dispositivo valido (con la chiamata dell'operazione di impostazione degli attributi di endpoint) e quindi attivarlo. Le consegne all'endpoint di piattaforma riusciranno solo dal momento dell'attivazione. L'unico caso in cui la riattivazione di un endpoint di piattaforma funziona senza l'aggiornamento del relativo token di dispositivo è quando un token di dispositivo associato a quell'endpoint che non era valido ridiventa valido. Ciò può verificarsi, ad esempio, quando un'app è stata disinstallata e quindi reinstallata nello

stesso dispositivo mobile e riceve lo stesso token di dispositivo. L'approccio descritto qui sopra effettua questa operazione, assicurandosi di riattivare un endpoint di piattaforma solo dopo aver verificato che il token di dispositivo ad esso associato è quello più recente disponibile.

Aggiunta token di dispositivo o ID di registrazione

Quando registri per la prima volta un'app e un dispositivo mobile con un servizio di notifica, come ad esempio Apple Push Notification Service (APN) e Firebase Cloud Messaging (FCM), i token di dispositivo o gli ID di registrazione vengono restituiti dal servizio di notifica. Quando aggiungi i token di dispositivo o gli ID di registrazione ad Amazon SNS, questi vengono usati con l'API `PlatformApplicationArn` per creare un endpoint per l'app e per il dispositivo. Quando Amazon SNS crea l'endpoint, viene restituito un elemento `EndpointArn`. L'`EndpointArn` indica ad Amazon SNS a quale app e dispositivo mobile inviare il messaggio di notifica.

Puoi aggiungere i token di dispositivo e gli ID di registrazione ad Amazon SNS con i seguenti metodi:

- Aggiunta manuale di un singolo token ad AWS utilizzando la AWS Management Console
- Caricamento di diversi token usando l'API `CreatePlatformEndpoint`
- Registrazione di token dai dispositivi che installeranno le tue app in futuro

Per aggiungere manualmente un token di dispositivo o un ID di registrazione

1. Accedi alla console Amazon SNS.
2. Scegli Mobile e poi Push notifications.
3. Nella sezione Platform applications, seleziona la tua applicazione, quindi scegli Edit. Se non hai già creato un'applicazione di piattaforma, creane una ora. Per istruzioni su come eseguire questa operazione, consultare Creazione di un'applicazione di piattaforma.
4. Scegli Add endpoints.
5. Nella casella Endpoint Token (Token di endpoint), immetti l'ID del token o l'ID di registrazione, a seconda del servizio di notifica. Ad esempio, con ADM e FCM immetti l'ID registrazione.
6. (Opzionale) Nella casella User Data (Dati utente), immetti le informazioni arbitrarie da associare all'endpoint. Amazon SNS non usa questi dati. I dati devono essere in formato UTF-8 e inferiori a 2 KB.
7. Infine, scegli Add Endpoints (Aggiungi endpoint).

   Ora con l'endpoint creato, puoi inviare messaggi direttamente a un dispositivo mobile o inviare messaggi a dispositivi mobili che hanno effettuato la sottoscrizione a un argomento.

Per caricare diversi tokens usando l'API `CreatePlatformEndpoint`

La procedura seguente mostra come utilizzare l'app di esempio Java (pacchetto `bulkupload`) fornita da AWS per caricare diversi token (token di dispositivo o ID di registrazione) in Amazon SNS. Puoi usare questa app di esempio per iniziare a caricare i token esistenti.

> **Note**
>
> Nei passaggi seguenti viene utilizzata l'IDE Java Eclipse. Nei passaggi si presuppone che tu abbia installato AWS SDK for Java e abbia le credenziali di sicurezza AWS per il tuo Account AWS. Per ulteriori informazioni, consulta AWS SDK for Java. Per ulteriori informazioni sulle credenziali, consulta Come ottenere le credenziali di sicurezza? in Riferimenti generali di AWS.

1. Download e decomprimi il file snsmobilepush.zip.
2. Crea un nuovo Progetto Java per in Eclipse.
3. Importa la cartella `SNSSamples` nella directory di primo livello del progetto Java appena creato. In Eclipse, fai clic con il pulsante destro del mouse sul nome del progetto Java e scegli Import (Importa), espandi General (Generale), scegli File System e quindi scegli Next (Avanti). Seleziona la cartella `SNSSamples`, scegli OK e quindi Finish (Termina).
4. Download una copia della libreria OpenCSV e aggiungila al percorso di compilazione del pacchetto `bulkupload`.
5. Apri il file `BulkUpload.properties` contenuto nel pacchetto `bulkupload`.
6. Aggiungi i seguenti elementi a `BulkUpload.properties`:
   - L'`ApplicationArn` a cui desideri aggiungere gli endpoint.
   - Il percorso assoluto per la posizione del file CSV contenente i token.
   - I nomi per i file CSV (come ad esempio `goodTokens.csv` e `badTokens.csv`) da creare per la registrazione dei token che Amazon SNS analizza correttamente o meno.
   - (Facoltativo) I caratteri per specificare il delimitatore e le virgolette nel file CSV contenente i token.
   - (Opzionale) Il numero di thread da utilizzare per creare gli endpoint contemporaneamente. Il valore predefinito è 1 thread.

L'elemento `BulkUpload.properties` completato si presenta in maniera analoga a quanto segue:

```
applicationarn:arn:aws:sns:us-west-2:111122223333:app/FCM/fcmpushapp
csvfilename:C:\\mytokendirectory\\mytokens.csv
goodfilename:C:\\mylogfiles\\goodtokens.csv
badfilename:C:\\mylogfiles\\badtokens.csv
delimiterchar:'
quotechar:"
numofthreads:5
```

7. Esegui l'applicazione BatchCreatePlatformEndpointSample.java per caricare i token in Amazon SNS.

   In questo esempio, gli endpoint creati per i token che sono stati caricati correttamente in Amazon SNS verrebbero registrati in `goodTokens.csv`, mentre i token con formato non valido verrebbero registrati in `badTokens.csv`. Inoltre, dovresti vedere i log STD OUT scritti nella console di Eclipse, contenenti contenuti simili ai seguenti:

   ```
   <1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-
   west-2:111122223333:app/FCM/fcmpushapp/165j2214-051z-3176-b586-138o3d420071
   <2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/mytokens.csv
   ```

Per registrare i token dai dispositivi che installeranno le tue app in futuro

Puoi utilizzare una delle seguenti due opzioni:

- Use the Amazon Cognito service: (Utilizza il servizio Amazon Cognito) la tua app per dispositivi mobili avrà bisogno di credenziali per creare endpoint associati all'applicazione della piattaforma Amazon SNS. Consigliamo di utilizzare credenziali temporanee che scadono dopo un periodo di tempo. Per la maggior parte degli scenari, ti consigliamo di utilizzare Amazon Cognito per creare credenziali di sicurezza temporanee. Per ulteriori informazioni, consulta la Guida per sviluppatori di Amazon Cognito . Se desideri ricevere una notifica quando un'app si registra con Amazon SNS, puoi registrarti per ricevere un evento che fornirà l'ARN del nuovo endpoint. Puoi anche usare l'API `ListEndpointByPlatformApplication` per ottenere l'elenco completo degli endpoint registrati con Amazon SNS.

- Usa un server proxy: se la tua infrastruttura applicativa è già configurata per le tue app mobili per effettuare chiamate e registrarti in ogni installazione, puoi continuare a utilizzare questa configurazione. Il tuo server fungerà da proxy e passerà il token del dispositivo alle notifiche push mobili ,Amazon SNS insieme a tutti i dati utente che desideri archiviare. A tale scopo, il server proxy si connetterà ad Amazon SNS utilizzando le tue credenziali AWS e userà la chiamata all'API `CreatePlatformEndpoint` per caricare le informazioni del token. Verrà restituito l'Amazon Resource Name (ARN) dell'endpoint appena creato che il server può archiviare per effettuare successive chiamate di pubblicazione ad Amazon SNS.

Metodi di autenticazione per Apple

Puoi autorizzare Amazon SNS a inviare notifiche push all'app iOS o macOS fornendo informazioni che identificano l'utente come sviluppatore dell'app. Per autenticarsi, fornire una chiave o un certificato durante la creazione di un'applicazione di piattaforma, entrambi possono essere ottenuti dal tuo account Apple Developer.

Chiave di firma dei token

Una chiave di firma privata utilizzata da Amazon SNS per firmare i token di autenticazione Push Notification Service (APN) di Apple.

Se si fornisce una chiave di firma, Amazon SNS utilizza un token per eseguire l'autenticazione con APN per ogni notifica push inviata. Con la chiave di firma, è possibile inviare notifiche push ad ambienti APN di produzione e sandbox.

La chiave di firma non ha scadenza e puoi utilizzare la stessa chiave di firma per più app. Per ulteriori informazioni, consulta Comunicare con gli APN utilizzando i token di autenticazione nella sezione Developer Account Help del sito Web Apple.

Certificate

Un certificato TLS che Amazon SNS utilizza per eseguire l'autenticazione con APN quando si inviano notifiche push. Si può ottenere il certificato dal proprio account sviluppatore Apple.

I certificati scadono dopo un anno. Alla scadenza, è necessario creare un nuovo certificato da fornire ad Amazon SNS. Per ulteriori informazioni, consulta Stabilire una connessione basata su certificato agli APN sul sito Web Apple Developer.

Per gestire le impostazioni APN utilizzando la Console di gestione di AWS

1. Accedi alla console Amazon SNS.
2. In Mobile, scegli Push notification (Notifiche push).
3. Seleziona la Applicazione per la quale si desidera modificare le impostazioni APN, quindi scegli Modificare.
4. Nella pagina Edit (Modificare), per Authentication type (Tipo di autenticazione), scegli Token (Token) o Certificate (Certificato).
5. Carica le credenziali appropriate per la chiave di firma dei certificati o dei token. Puoi ottenere queste informazioni dal tuo account sviluppatori Apple.
6. A seconda del tipo di autenticazione scelto, esegui una delle seguenti operazioni:
   - Se si sceglie Token (Token), fornisci le informazioni che seguono dal tuo account Apple Developer. Amazon SNS richiede queste informazioni per creare token di autenticazione.
     - Signing key (Chiave di firma): la chiave di firma del token di autenticazione dal tuo account Apple Developer, che scarichi come file.p8. Apple consente di scaricare la chiave di firma solo una volta.
     - Signing key ID (ID chiave di firma): l'ID assegnato alla chiave di firma. Amazon SNS richiede queste informazioni per creare token di autenticazione. Per trovare questo valore, scegli Certificates, IDs & Profiles (Certificati, ID e profili) e quindi scegli la propria chiave nella sezione Keys (Chiavi).
     - Team identifier (Identificatore team): l'ID assegnato al team di account sviluppatori Apple. Puoi trovare questo valore sulla pagina Membership (Appartenenza).
     - Bundle identifier (Identificatore bundle): l'ID assegnato all'app. Per trovare questo valore, scegli Certificates, IDs & Profiles (Certificati, ID e profili), scegli App IDs (ID app) nella sezione Identifiers (Identificatori), quindi scegli l'app.
   - Se si sceglie Certificate (Certificato), è necessario fornire le seguenti informazioni:
     - SSL certificate (Certificato SSL): il file .p12 per il certificato TLS. È possibile esportare questo file da Keychain Access dopo avere scaricato e installato il certificato dall'account sviluppatore Apple.
     - Certificate password (Password del certificato): se hai assegnato una password al certificato, specificala qui.
7. Al termine, scegliere Save changes (Salva modifiche).

Metodi di autenticazione di Firebase Cloud Messaging (FCM)

Questo argomento descrive come ottenere le credenziali API FCM (HTTP v1) richieste da Google da utilizzare con l' AWS API e il. AWS CLI AWS Management Console

Argomenti

> **Important**
>
> 20 giugno 2023 — Google ha reso obsoleta la propria API HTTP legacy di Firebase Cloud Messaging (FCM). Amazon SNS ora supporta la distribuzione a tutti i tipi di dispositivi utilizzando l'API HTTP v1 di FCM. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima API HTTP v1 di FCM entro il 1° giugno 2024 per evitare interruzioni.
> 18 gennaio 2024 — Amazon SNS ha introdotto il supporto per l'API HTTP v1 di FCM per la consegna di notifiche push mobili ai dispositivi Android.
> 26 marzo 2024 — Amazon SNS supporta l'API HTTP v1 FCM per dispositivi Apple e destinazioni Webpush. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima API HTTP v1 di FCM entro il 1° giugno 2024 per evitare interruzioni delle applicazioni.

Puoi autorizzare Amazon SNS a inviare notifiche push alle applicazioni fornendo informazioni che ti identificano come sviluppatore dell'app. Per l'autenticazione, fornisci una chiave API o un token durante la creazione di un'applicazione di piattaforma. Puoi ottenere le seguenti informazioni dalla console dell'applicazione Firebase:

Chiave API

La chiave API rappresenta le credenziali utilizzate per chiamare l'API Legacy di Firebase. Le API FCM legacy verranno rimosse da Google il 20 giugno 2024. Se attualmente utilizzi una chiave API come credenziali della piattaforma, puoi aggiornare le credenziali della piattaforma selezionando Token come opzione e caricando il file JSON associato all'applicazione Firebase.

Token

Quando si chiama l'API HTTP v1, viene utilizzato un token di accesso di breve durata. Questa è l'API consigliata da Firebase per l'invio di notifiche push. Per generare token di accesso, Firebase fornisce agli sviluppatori un set di credenziali sotto forma di file di chiave privata (noto anche come file service.json).

Prerequisito

Prima di poter iniziare a gestire le impostazioni FCM in Amazon SNS, devi ottenere le credenziali FCM service.json. Per ottenere le credenziali service.json, consulta Migrazione dalle API FCM legacy a HTTP v1 nella documentazione di Google Firebase.

Gestione delle impostazioni FCM (API)

Puoi creare notifiche push FCM utilizzando l'API. AWS Il numero e le dimensioni delle risorse Amazon SNS in un AWS account sono limitati. Per ulteriori informazioni, consulta gli endpoint e le quote di Amazon Simple Notification Service nella Riferimenti generali di AWS Guida.

Per creare una notifica push FCM insieme a un argomento AWS (API) di Amazon SNS

Quando utilizzi le credenziali chiave, `PlatformCredential` è `API key`. Quando utilizzi le credenziali token, `PlatformCredential` è un file di chiavi private in formato JSON:

- `CreatePlatformApplication`

Per recuperare un tipo di credenziale FCM per un argomento (API) di Amazon SNS esistente AWS

Recupera il tipo di credenziali, `"AuthenticationMethod": "Token"` o `"AuthenticationMethod": "Key"`:

- GetPlatformApplicationAttributes

Configurazione di un attributo FCM per un argomento Amazon SNS esistente (API AWS)

Configura l'attributo FCM:

- SetPlatformApplicationAttributes

Gestione elle impostazioni FCM (CLI)

È possibile creare notifiche push FCM utilizzando AWS Command Line Interface (CLI). Il numero e le dimensioni delle risorse Amazon SNS in un AWS account sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di una notifica push FCM insieme a un argomento SNS (AWS CLI)

Quando utilizzi le credenziali chiave, `PlatformCredential` è `API key`. Quando utilizzi le credenziali token, `PlatformCredential` è un file di chiavi private in formato JSON. Quando si utilizza la AWS CLI, il file deve essere in formato stringa e i caratteri speciali devono essere ignorati. Per formattare correttamente il file, Amazon SNS consiglia di utilizzare il seguente comando: `SERVICE_JSON=`jq @json <<< cat service.json``

- create-platform-application

Recupero di un tipo di credenziali FCM per un argomento Amazon SNS esistente (AWS CLI)

Recupera il tipo di credenziali, `"AuthenticationMethod": "Token"` o `"AuthenticationMethod": "Key"`:

- get-platform-application-attributes

Configurazione di un attributo FCM per un argomento Amazon SNS esistente (AWS CLI)

Configura l'attributo FCM:

- set-platform-application-attributes

Gestione delle impostazioni FCM (console)

Utilizza la procedura seguente per inserire le credenziali utilizzate dall'applicazione per connettersi a FCM.

1. Accedi alla console Amazon SNS.
2. In Mobile, scegli Push notification (Notifiche push).
3. Seleziona un'applicazione FCM esistente e scegli Modifica. Se non hai già creato un'applicazione di piattaforma, consulta Creazione di un'applicazione di piattaforma.

4. Nella pagina Modifica, per Credenziali Firebase Cloud Messaging, scegli Token o Chiave. Puoi ottenere le seguenti informazioni dalla console dell'applicazione Firebase.
   - Se scegli Token, carica un file della chiave privata valido. Il contenuto di questo file viene utilizzato per generare token di accesso di breve durata durante l'invio di notifiche.
   - Se scegli Chiave, inserisci la chiave API di Google.
5. Al termine, scegliere Save changes (Salva modifiche).

Argomenti correlati

- Utilizzo dei payload Google Firebase Cloud Messaging (FCM) v1 in Amazon SNS

Gestione degli endpoint Firebase Cloud Messaging (FCM)

Argomenti

- Gestione e manutenzione dei token dei dispositivi
- Rilevamento di token non validi
- Rimuovere i token obsoleti

Gestione e manutenzione dei token dei dispositivi

Puoi garantire la consegna delle notifiche push della tua applicazione mobile seguendo questi passaggi:

1. Archivia tutti i token del dispositivo, gli ARN degli endpoint Amazon SNS corrispondenti e i timestamp sul tuo server delle applicazioni.
2. Rimuovi tutti i token obsoleti ed elimina gli ARN endpoint Amazon SNS corrispondenti.

All'avvio iniziale dell'app, riceverai un token del dispositivo (noto anche come token di registrazione) per il dispositivo. Questo token del dispositivo viene coniato dal sistema operativo del dispositivo ed è collegato all'applicazione FCM. Una volta ricevuto questo token del dispositivo, puoi registrarlo con Amazon SNS come endpoint della piattaforma. Ti consigliamo di archiviare il token del dispositivo, l'ARN dell'endpoint della piattaforma Amazon SNS e il timestamp salvandoli sul tuo server delle applicazioni o su un altro archivio persistente. Per configurare l'applicazione FCM per recuperare e archiviare i token del dispositivo, consulta Recuperare e archiviare i token di registrazione nella documentazione di Google su Firebase.

È importante mantenere i token. up-to-date I token del dispositivo dell'utente possono cambiare nelle seguenti condizioni:

1. L'applicazione mobile viene ripristinata su un nuovo dispositivo.
2. L'utente disinstalla o aggiorna l'applicazione.
3. L'utente cancella i dati dell'applicazione.

Quando il token del dispositivo cambia, ti consigliamo di aggiornare l'endpoint Amazon SNS corrispondente con il nuovo token. Ciò consente ad Amazon SNS di continuare la comunicazione con il dispositivo registrato. Puoi farlo implementando il seguente pseudo codice all'interno della tua applicazione mobile. Descrive una pratica consigliata per la creazione e la manutenzione degli endpoint della piattaforma abilitati. Questo approccio può essere eseguito ogni volta che le applicazioni mobili vengono avviate o come processo pianificato in background.

Pseudocodice

Utilizza il seguente pseudo codice FCM per gestire e mantenere i token dei dispositivi.

```
retrieve the latest token from the mobile OS
if (endpoint arn not stored)
    # first time registration
    call CreatePlatformEndpoint
    store returned endpoint arn
endif

call GetEndpointAttributes on the endpoint arn

if (getting attributes encountered NotFound exception)
    #endpoint was deleted
    call CreatePlatformEndpoint
    store returned endpoint arn
else
    if (token in endpoint does not match latest) or
        (GetEndpointAttributes shows endpoint as disabled)
        call SetEndpointAttributes to set the
                    latest token and enable the endpoint
    endif
endif
```

Per ulteriori informazioni sui requisiti di aggiornamento dei token, consulta Aggiornare i token regolarmente nella documentazione di Google su Firebase.

Rilevamento di token non validi

Quando un messaggio viene inviato a un endpoint FCM v1 con un token di dispositivo non valido, Amazon SNS riceverà una delle seguenti eccezioni:

- `UNREGISTERED`(HTTP 404) — Quando Amazon SNS riceve questa eccezione, riceverai un evento di errore `FailureType` di `InvalidPlatformToken` consegna con un token di piattaforma *`FailureMessage`* of e uno of associato all'endpoint non valido. Amazon SNS disabiliterà l'endpoint della piattaforma quando una consegna fallisce, con questa eccezione.
- `INVALID_ARGUMENT`(HTTP 400) — Quando Amazon SNS riceve questa eccezione, significa che il token del dispositivo o il payload del messaggio non sono validi. Per ulteriori informazioni, consulta la documentazione di Google su ErrorCodeFirebase.

Poiché `INVALID_ARGUMENT` può essere restituito in entrambi i casi, Amazon SNS restituirà un corpo *`FailureType`* di *`InvalidNotification`* notifica e uno *`FailureMessage`* di non è valido. Quando ricevi questo errore, verifica che il payload sia corretto. Se è corretto, verifica che il token del dispositivo lo sia up-to-date. Amazon SNS non disattiverà l'endpoint della piattaforma quando una consegna fallisce, con questa eccezione.

Un altro caso in cui si verificherà un errore di `InvalidPlatformToken` consegna è quando il token del dispositivo registrato non appartiene all'applicazione che tenta di inviare il messaggio. In questo caso, Google restituirà un errore SENDER_ID_MISMATCH. Amazon SNS disabiliterà l'endpoint della piattaforma quando una consegna fallisce, con questa eccezione.

Tutti i codici di errore rilevati ricevuti dall'API FCM v1 sono disponibili CloudWatch quando configuri la registrazione dello stato di consegna per la tua applicazione.

Per ricevere gli eventi di consegna della tua applicazione, consulta. Eventi applicazione disponibili

Rimuovere i token obsoleti

I token vengono considerati obsoleti una volta che il recapito dei messaggi al dispositivo endpoint inizia a fallire. Amazon SNS imposta questi token obsoleti come endpoint disabilitati per l'applicazione della tua piattaforma. Quando pubblichi su un endpoint disabilitato, Amazon SNS restituirà `EventDeliveryFailure` un evento con `FailureType` `EndpointDisabled` of e

`FailureMessage` un endpoint disabilitato. Per ricevere eventi di consegna per la tua applicazione, consulta. Eventi applicazione disponibili

Quando ricevi questo errore da Amazon SNS, devi rimuovere o aggiornare il token obsoleto nell'applicazione della tua piattaforma.

Invio di notifiche push per dispositivi mobili

Questa sezione descrive come inviare un messaggio di notifica push.

Argomenti

- Pubblicazione in un argomento
- Pubblicazione in un dispositivo mobile
- Pubblicazione con payload specifici della piattaforma

Pubblicazione in un argomento

Puoi anche usare Amazon SNS per inviare i messaggi agli endpoint mobili che hanno effettuato la sottoscrizione a un argomento. Il concetto è lo stesso della sottoscrizione di altri tipi di endpoint, ad esempio Amazon SQS, HTTP/S, e-mail e SMS, a un argomento, come descritto in Che cos'è Amazon SNS?. La differenza è che Amazon SNS comunica attraverso servizi di notifica come Apple Push Notification Service (APNS) e Google Firebase Cloud Messaging (FCM). Attraverso i servizi di notifica, gli endpoint mobili sottoscritti ricevano le notifiche inviate all'argomento.

Pubblicazione in un dispositivo mobile

Invia i messaggi di notifica push Amazon SNS direttamente a un endpoint che rappresenta un'applicazione su un dispositivo mobile.

Per inviare un messaggio diretto

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegli Push notifications (Notifiche push).
3. Nella pagina Notifiche push per dispositivi mobili, nella sezione Applicazioni della piattaforma, scegli ad esempio il nome dell'applicazione ***MyApp***.
4. Nella ***MyApp***pagina, nella sezione Endpoints, scegli un endpoint, quindi scegli Pubblica messaggio.

5. Nella pagina Publish message to endpoint (Pubblica messaggio a un endpoint), inserisci il messaggio da visualizzare nell'applicazione sul dispositivo mobile e quindi scegli Publish message (Pubblica messaggio).

   Amazon SNS invia il messaggio di notifica al servizio di notifica della piattaforma che, a sua volta, invia il messaggio all'applicazione.

Pubblicazione con payload specifici della piattaforma

Puoi utilizzare le AWS Management Console o le API di Amazon SNS per inviare messaggi personalizzati con payload specifici della piattaforma ai dispositivi mobili. Per informazioni sull'utilizzo delle API Amazon SNS, consulta Operazioni API push per dispositivi mobili e il file `SNSMobilePush.java` in `snsmobilepush.zip`.

Argomenti

- Invio di messaggi in formato JSON
- Invio di messaggi specifici della piattaforma
- Invio di messaggi a un'applicazione su più piattaforme
- Invio di messaggi a APN come notifiche o notifiche in background
- Utilizzo dei payload Google Firebase Cloud Messaging (FCM) v1 in Amazon SNS

Invio di messaggi in formato JSON

Quando si inviano payload specifici della piattaforma, i dati devono essere stringhe coppia chiave-valore in formato JSON, con virgolette doppie con carattere di escape.

I seguenti esempi mostrano un messaggio personalizzato per la piattaforma FCM .

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

Invio di messaggi specifici della piattaforma

Oltre a inviare dati personalizzati come coppie chiave-valore, è possibile inviare coppie chiave-valore specifiche per la piattaforma.

L'esempio seguente mostra l'inclusione dei parametri FCM `time_to_live` e `collapse_key` dopo le coppie chiave-valore dei dati personalizzate nel parametro FCM `data`.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Per l'elenco delle coppie chiave-valore supportate in ciascuno dei servizi di notifica push supportati in Amazon SNS, consulta i collegamenti seguenti:

> **Important**
>
> Amazon SNS ora supporta l'API HTTP v1 Firebase Cloud Messaging (FCM) per l'invio di notifiche push mobili ai dispositivi Android.
> 26 marzo 2024 — Amazon SNS supporta l'API HTTP v1 FCM per dispositivi Apple e destinazioni Webpush. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima API HTTP v1 di FCM entro il 1° giugno 2024 per evitare interruzioni delle applicazioni.

- Payload Key Reference (Riferimento alla chiave di payload) nella documentazione
- Firebase Cloud Messaging HTTP Protocol (Protocollo HTTP di Firebase Cloud Messaging) nella documentazione .
- Invio di un messaggio nella documentazione

Invio di messaggi a un'applicazione su più piattaforme

Per inviare un messaggio a un'applicazione installata su dispositivi per più piattaforme, come FCM e APN, devi prima effettuare la sottoscrizione degli endpoint mobili a un argomento in Amazon SNS e quindi pubblicare il messaggio nell'argomento.

L'esempio seguente mostra un messaggio da inviare agli endpoint mobili che hanno effettuato la sottoscrizione su APN, FCM e ADM:

```
{
```

```
  "default": "This is the default message which must be present when publishing a
 message to a topic. The default message will only be used if a message is not present
 for
one of the notification platforms.",
  "APNS": "{\"aps\":{\"alert\": \"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"} }",
  "GCM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}",
  "ADM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}"
}
```

Invio di messaggi a APN come notifiche o notifiche in background

Amazon SNS può inviare messaggi ad APN come notifiche `alert` o `background` (per ulteriori informazioni, consulta Invio di aggiornamenti in background alla tua app nella documentazione APN).

- Una notifica APN `alert` informa gli utenti visualizzando un messaggio di avviso, riproducendo un suono o aggiungendo un badge all'icona dell'applicazione.
- Una notifica APN `background` si attiva o indica all'applicazione di agire sul contenuto della notifica, senza informare l'utente.

Specificare i valori di intestazione APN personalizzati

Ti consigliamo di specificare valori personalizzati per l'attributo del messaggio `AWS.SNS.MOBILE.APNS.PUSH_TYPE` riservato utilizzando l'azione dell'API Amazon `Publish` SNS AWS , gli SDK o il. AWS CLI Il seguente esempio di CLI imposta `content-available` su `1` e `apns-push-type` su `background` per l'argomento specificato.

```
aws sns publish \
--endpoint-url https://sns.us-east-1.amazonaws.com \
--target-arn arn:aws:sns:us-east-1:123456789012:endpoint/APNS_PLATFORM/MYAPP/1234a567-
bc89-012d-3e45-6fg7h890123i \
--message '{"APNS_PLATFORM":"{\"aps\":{\"content-available\":1}}"}' \
--message-attributes '{ \
  "AWS.SNS.MOBILE.APNS.TOPIC":
{"DataType":"String","StringValue":"com.amazon.mobile.messaging.myapp"}, \
  "AWS.SNS.MOBILE.APNS.PUSH_TYPE":{"DataType":"String","StringValue":"background"} \
  "AWS.SNS.MOBILE.APNS.PRIORITY":{"DataType":"String","StringValue":"5"}}', \
--message-structure json
```

Indurre l'intestazione del tipo push APN dal payload

Se non si imposta l'intestazione APN `apns-push-type`, Amazon SNS imposta l'intestazione su `alert` o `background` a seconda della chiave `content-available` nel dizionario `aps` della configurazione del payload APN formattato JSON.

> **Note**
>
> Amazon SNS è in grado di dedurre solo le intestazioni `alert` o `background`, anche se l'intestazione `apns-push-type` può essere impostata su altri valori.

- `apns-push-type` è impostato su `alert`.
  - Se il dizionario `aps` contiene `content-available` impostato su `1` e una o più chiavi che attivano le interazioni dell'utente.
  - Se il dizionario `aps` contiene `content-available` impostato su `0` o se la chiave `content-available` è assente.
  - Se il valore della chiave `content-available` non è un numero intero o un valore booleano.
- `apns-push-type` è impostato su `background`.
  - Se il dizionario `aps` contiene solo `content-available` impostato su `1` e nessun'altra chiave che attiva le interazioni dell'utente.

> **Important**
>
> Se Amazon SNS invia un oggetto di configurazione raw per APNA come notifica di solo sfondo, è necessario includere `content-available` impostato su `1` nel dizionario `aps`. Sebbene sia possibile includere chiavi personalizzate, il dizionario `aps` non deve contenere chiavi che attivino interazioni dell'utente (ad esempio avvisi, badge o suoni).

Di seguito è riportato un esempio di oggetto configurazione non elaborato.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

In questo esempio, Amazon SNS imposta l'intestazione APN `apns-push-type` su `background` per il messaggio. Quando Amazon SNS rileva che il dizionario `apn` contiene la chiave `content-available` impostata su `1` e non contiene altre chiavi che possono attivare interazioni utente imposta l'intestazione su `background`.

Utilizzo dei payload Google Firebase Cloud Messaging (FCM) v1 in Amazon SNS

Amazon SNS supporta l'utilizzo dell'API HTTP v1 di FCM per inviare notifiche a destinazioni Android, iOS e Webpush. Questo argomento fornisce esempi della struttura del payload durante la pubblicazione di notifiche push per dispositivi mobili utilizzando la CLI o l'API Amazon SNS.

Puoi includere i seguenti tipi di messaggi nel tuo payload quando invii una notifica FCM:

- Messaggio di dati: un messaggio di dati viene gestito dall'app client e contiene coppie chiave-valore personalizzate. Quando si crea un messaggio di dati, è necessario includere la `data` chiave con un oggetto JSON come valore, quindi inserire le coppie chiave-valore personalizzate.
- Messaggio di notifica o messaggio visualizzato: un messaggio di notifica contiene un set predefinito di chiavi gestite da FCM SDK. Queste chiavi variano a seconda del tipo di dispositivo a cui vengono consegnate. Per ulteriori informazioni sui tasti di notifica specifici della piattaforma, consulta quanto segue:
  - Tasti di notifica Android
  - Chiavi di notifica APNS
  - Chiavi di notifica Webpush

Per ulteriori informazioni sui tipi di messaggi FCM, consulta Tipi di messaggio nella documentazione di Firebase di Google.

Indice

Utilizzo della struttura di payload FCM v1 per inviare messaggi

Se stai creando un'applicazione FCM per la prima volta o desideri sfruttare le funzionalità di FCM v1, puoi scegliere di inviare un payload in formato FCM v1. A tale scopo, è necessario includere la chiave di primo livello. `fcmV1Message` Per ulteriori informazioni sulla creazione di payload FCM v1,

consulta Migrazione dalle API FCM precedenti a HTTP v1 e Personalizzazione di un messaggio tra piattaforme nella documentazione Firebase di Google.

Payload di esempio FCM v1 inviato ad Amazon SNS:

> **Note**
>
> Il valore della `GCM` chiave utilizzato nell'esempio seguente deve essere codificato come stringa quando si pubblica una notifica tramite Amazon SNS.

```
{
  "GCM": "{
     \"fcmV1Message\": {
        \"validate_only\" : false,
        \"message\" :
            {
            \"notification\": {
                \"title\": \"string\",
                \"body\": \"string\"
            },
            \"data\": {
                \"dataGen\": \"priority message\",
            },
            \"android\": {
                \"priority\": \"high\",
                \"notification\": {
                    \"body_loc_args\": [
                    \"string\"
                    ],
                    \"title_loc_args\": [
                    \"string\"
                    ],
                    \"sound\": \"string\",
                    \"title_loc_key\": \"string\",
                    \"title\": \"string\",
                    \"body\": \"string\",
                    \"click_action\": \"clicky_clacky\",
                    \"body_loc_key\": \"string\"
                },
                \"data\": {
                    \"dataAndroid\": \"priority message\",
```

```
            },
            \"ttl\": \"10023.32s\"
        },
        \"apns\": {
            \"payload\": {
                \"aps\": {
                    \"alert\": {
                        \"subtitle\": \"string\",
                        \"title-loc-args\": [
                        \"string\"
                        ],
                        \"title-loc-key\": \"string\",
                        \"loc-args\": [
                        \"string\"
                        ],
                        \"loc-key\": \"string\",
                        \"title\": \"string\",
                        \"body\": \"string\"
                    },
                    \"category\": \"Click\",
                    \"content-available\": 0,
                    \"sound\": \"string\",
                    \"badge\": 5
                }
            }
        },
        \"webpush\": {
            \"notification\": {
                \"badge\": \"5\",
                \"title\": \"string\",
                \"body\": \"string\"
            },
            \"data\": {
                \"dataWeb\": \"priority message\",
            }
        }
    }
  }"
}
```

Quando invii un payload JSON, assicurati di includere l'`message-structure`attributo nella richiesta e di impostarlo su. `json`

Esempio di CLI:

```
aws sns publish --topic $TOPIC_ARN --message '{"GCM": "{\"fcmV1Message\": {\"message\":
{\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"android\":{\"priority
\":\"high\",\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"data\":
{\"customAndroidDataKey\":\"custom key value\"},\"ttl\":\"0s\"},\"apns\":{\"payload
\":{\"aps\":{\"alert\":{\"title\":\"string\", \"body\":\"string\"},\"content-
available\":1,\"badge\":5}}},\"webpush\":{\"notification\":{\"badge\":\"URL\",\"body
\":\"Test\"},\"data\":{\"customWebpushDataKey\":\"priority message\"}},\"data\":
{\"customGeneralDataKey\":\"priority message\"}}}}", "default": "{\"notification\":
 {\"title\": \"test\"}"}' --region $REGION --message-structure json
```

Per ulteriori informazioni sull'invio di payload in formato FCM v1, consulta quanto segue nella documentazione Firebase di Google:

- Esegui la migrazione dalle API FCM precedenti a HTTP v1
- Informazioni sui messaggi FCM
- Risorsa REST: projects.messages

Utilizzo della struttura di payload legacy per inviare messaggi all'API FCM v1

Durante la migrazione a FCM v1, non è necessario modificare la struttura del payload utilizzata per le credenziali precedenti. Amazon SNS trasforma il tuo payload nella nuova struttura di payload FCM v1 e lo invia a Google.

Formato del payload del messaggio di input:

```
{
  "GCM": "{\"notification\": {\"title\": \"string\", \"body\": \"string\",
 \"android_channel_id\": \"string\", \"body_loc_args\": [\"string\"], \"body_loc_key\":
 \"string\", \"click_action\": \"string\", \"color\": \"string\", \"icon\": \"string
\", \"sound\": \"string\", \"tag\": \"string\", \"title_loc_args\": [\"string\"],
 \"title_loc_key\": \"string\"}, \"data\": {\"message\": \"priority message\"}}"
}
```

Messaggio inviato a Google:

```
{
  "message": {
    "token": "***",
    "notification": {
```

```
      "title": "string",
      "body": "string"
    },
    "android": {
      "priority": "high",
      "notification": {
        "body_loc_args": [
          "string"
        ],
        "title_loc_args": [
          "string"
        ],
        "color": "string",
        "sound": "string",
        "icon": "string",
        "tag": "string",
        "title_loc_key": "string",
        "title": "string",
        "body": "string",
        "click_action": "string",
        "channel_id": "string",
        "body_loc_key": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "apns": {
      "payload": {
        "aps": {
          "alert": {
            "title-loc-args": [
              "string"
            ],
            "title-loc-key": "string",
            "loc-args": [
              "string"
            ],
            "loc-key": "string",
            "title": "string",
            "body": "string"
          },
          "category": "string",
          "sound": "string"
```

```
        }
      }
    },
    "webpush": {
      "notification": {
        "icon": "string",
        "tag": "string",
        "body": "string",
        "title": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "data": {
      "message": "priority message"
    }
  }
}
```

Rischi potenziali

- La mappatura dalla versione precedente alla versione 1 non supporta l'Apple Push Notification Service (APNS) `headers` o le chiavi. `fcm_options` Se desideri utilizzare questi campi, invia un payload FCM v1.
- In alcuni casi, le intestazioni dei messaggi sono necessarie da FCM v1 per inviare notifiche silenziose ai dispositivi APN. Se al momento stai inviando notifiche silenziose ai tuoi dispositivi APN, queste non funzioneranno con l'approccio precedente. Ti consigliamo invece di utilizzare il payload FCM v1 per evitare problemi imprevisti. Per trovare un elenco delle intestazioni APN e per cosa vengono utilizzate, consulta Communicating with APNs nella Apple Developer Guide.
- Se utilizzi l'attributo TTL Amazon SNS per inviare la notifica, questo verrà aggiornato solo sul `android` campo. Se desideri impostare l'attributo TTL APNS, utilizza il payload FCM v1.
- Le `webpush` chiavi `androidapns`, e verranno mappate e compilate con tutte le chiavi pertinenti fornite. Ad esempio, se fornisci `title` una chiave condivisa tra tutte e tre le piattaforme, la mappatura FCM v1 popolerà tutte e tre le piattaforme con il titolo che hai fornito.
- Alcune chiavi condivise tra piattaforme prevedono tipi di valori diversi. Ad esempio, la `badge` chiave passata a `apns` prevede un valore intero, mentre la `badge` chiave passata a `webpush` prevede un valore String. Nei casi in cui si fornisce la `badge` chiave, la mappatura FCM v1 popolerà solo la chiave per la quale è stato fornito un valore valido.

Eventi di mancata consegna da parte di FCM

La tabella seguente fornisce il tipo di errore di Amazon SNS che corrisponde ai codici di errore/stato ricevuti da Google per le richieste di notifica FCM v1. Tutti i codici di errore osservati ricevuti dall'API FCM v1 sono disponibili CloudWatch quando configuri la registrazione dello stato di consegna per la tua applicazione.

Codice di errore/stato FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
`UNREGISTERED`	`InvalidPlatformToken`	Il token di piattaforma associato all'endpoint non è valido.	Il token del dispositivo collegato all'endpoint è obsoleto o non valido. Amazon SNS ha disabilitato il tuo endpoint. Aggiorna l'endpoint Amazon SNS al token del dispositivo più recente.
`INVALID_ARGUMENT`	`InvalidNotification`	Il corpo della notifica non è valido.	Il token del dispositivo o il payload del messaggio potrebbero non essere validi. Verifica che il payload del messaggio sia valido. Se il payload del messaggio è valido, aggiorna l'endpoint Amazon SNS al token del dispositivo più recente.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	Il token della piattaforma associato	L'applicazione della piattaforma associata al token del dispositi

Codice di errore/stato FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
		all'endpoint non è valido.	vo non dispone dell'autorizzazione per l'invio al token del dispositivo. Verifica di utilizzare le credenzia li FCM corrette nell'applicazione della piattaforma Amazon SNS.
`UNAVAILABLE`	`Dependenc yUnavailable`	La dipendenza non è disponibile.	FCM non è riuscito a elaborare la richiesta in tempo. Tutti i nuovi tentativi eseguiti da Amazon SNS non sono riusciti. Puoi archiviare questi messaggi in una coda di lettere morte (DLQ) e reindirizzarli in un secondo momento.
`INTERNAL`	`Unexpecte dFailure`	Guasto imprevisto; contatta Amazon. Frase di errore [Errore interno].	Il server FCM ha riscontrato un errore durante il tentativo di elaborare la richiesta . Tutti i nuovi tentativi eseguiti da Amazon SNS non sono riusciti. Puoi archiviare questi messaggi in una coda di lettere morte (DLQ) e reindirizzarli in un secondo momento.

Codice di errore/stato FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
`THIRD_PARTY_AUTH_ERROR`	`InvalidCredentials`	Le credenziali dell'applicazione della piattaforma non sono valide.	Non è stato possibile inviare un messaggio indirizzato a un dispositivo iOS o a un dispositivo Webpush. Verifica che le tue credenziali di sviluppo e produzione siano valide.
`QUOTA_EXCEEDED`	`Throttled`	Richiesta limitata da [gcm].	È stata superata la quota per la frequenza dei messaggi, la quota per la velocità dei messaggi del dispositivo o la quota relativa alla frequenza dei messaggi per argomento. Per informazioni su come risolvere questo problema, consulta la ErrorCodedocumentazione Firebase di Google.

Codice di errore/stato FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
`PERMISSION_DENIED`	`InvalidNotification`	Il corpo della notifica non è valido.	In caso di `PERMISSION_DENIED` eccezione, il chiamante (l'applicazione FCM) non è autorizzato a eseguire l'operazione specificata nel payload. Accedi alla console FCM e verifica che le tue credenziali abbiano abilitato le azioni API richieste.

Attributi per app

Amazon Simple Notification Service (Amazon SNS) fornisce supporto per la registrazione dello stato di consegna dei messaggi di notifica push. Dopo la configurazione degli attributi di applicazione, le voci di log sono inviate a CloudWatch Logs per i messaggi inviati da Amazon SNS a endpoint mobili. La registrazione dello stato di consegna dei messaggi consente di ottenere informazioni operative più precise, ad esempio:

- Sapere se un messaggio di notifica push è stato consegnato da Amazon SNS al servizio di notifica push.
- Identificare la risposta inviata dal servizio di notifica push a Amazon SNS.
- Determinare il tempo di attesa dei messaggi (il periodo di tempo tra il timestamp di pubblicazione e l'istante immediatamente precedente alla consegna a un servizio di notifica push).

Per configurare gli attributi di applicazione per lo stato di consegna dei messaggi, puoi utilizzare la AWS Management Console, i kit SDK (Software Development Kit) AWS o l'API di query.

Argomenti

- Configurazione degli attributi dello stato di consegna dei messaggi utilizzando la AWS Management Console
- Esempi di stato di consegna messaggi Amazon SNS CloudWatch log
- Configurazione degli attributi dello stato di consegna dei messaggi con i kit SDK AWS
- Codici di risposta della piattaforma

Configurazione degli attributi dello stato di consegna dei messaggi utilizzando la AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel riquadro di navigazione, scegliere Mobile (Dispositivi mobili), Push notifications (Notifiche push).
3. Dalla sezione Applicazioni di piattaforma selezionare l'applicazione contenente gli endpoint per cui si desidera ricevere i CloudWatch Logs.
4. Scegli Application Actions (Operazioni applicazione), quindi scegli Delivery status (Stato consegna).
5. Nella finestra di dialogo Delivery Status (Stato consegna), scegli Create IAM Roles (Crea ruoli IAM).

   Si viene reindirizzati alla console IAM.
6. Fai clic su Allow (Consenti) per consentire ad Amazon SNS l'accesso in scrittura a CloudWatch Logs a tuo nome.
7. Nella finestra di dialogo Delivery Status (Stato consegna), immetti un numero nel campo Percentage of Success to Sample (0-100) (Percentuale di consegne riuscite - 0-100) per la percentuale di messaggi inviati con successo per cui desideri ricevere CloudWatch Logs.

   > **Note**
   >
   > Dopo la configurazione degli attributi di applicazione per lo stato di consegna dei messaggi, tutte le consegne di messaggi non riuscite generano dei CloudWatch Logs.
8. Infine scegli Save Configuration (Salva configurazione). A questo punto puoi visualizzare e analizzare i CloudWatch Logs contenenti lo stato di consegna dei messaggi. Per ulteriori informazioni sull'utilizzo di CloudWatch e degli allarmi, consulta la documentazione di Amazon CloudWatch.

Esempi di stato di consegna messaggi Amazon SNS CloudWatch log

Dopo aver configurato gli attributi dello stato di consegna dei messaggi per un endpoint applicazione, verranno generati dei CloudWatch Logs. Di seguito vengono forniti degli esempi di log in formato JSON:

RIUSCITO


```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
    "token": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\":[{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```


NON RIUSCITO


```
{
  "status": "FAILURE",
  "notification": {
    "timestamp": "2015-01-26 23:29:35.678",
    "messageId": "c3ad79b0-8996-550a-8bfa-24f05989898f"
  },
  "delivery": {
    "statusCode": 8,
    "dwellTimeMs": 1451,
    "token": "examp1e29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
```

```
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken,
 id=1, cause=null)",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/
APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

Per un elenco dei codici di risposta dei servizi di notifica push, consulta Codici di risposta della piattaforma.

Configurazione degli attributi dello stato di consegna dei messaggi con i kit SDK AWS

I SDK AWS forniscono API in vari linguaggi per l'utilizzo degli attributi relativi allo stato di consegna dei messaggi con Amazon SNS.

L'esempio Java seguente mostra come utilizzare l'API `SetPlatformApplicationAttributes` per configurare attributi di applicazione per lo stato di consegna dei messaggi di notifica push. Puoi utilizzare i seguenti attributi per lo stato di consegna dei messaggi: `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn` e `SuccessFeedbackSampleRate`. Gli attributi `SuccessFeedbackRoleArn` e `FailureFeedbackRoleArn` sono utilizzati per consentire ad Amazon SNS l'accesso in scrittura per utilizzare CloudWatch Logs. L'attributo `SuccessFeedbackSampleRate` consente di specificare la percentuale della frequenza di campionamento (0-100) dei messaggi consegnati. Dopo la configurazione dell'attributo `FailureFeedbackRoleArn`, tutte le consegne di messaggi non riuscite generano dei CloudWatch Logs.

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
 SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-
west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

Per ulteriori informazioni su SDK per Java, consulta Nozioni di base su AWS SDK for Java.

Codici di risposta della piattaforma

Di seguito viene fornito un elenco dei collegamenti per i codici di risposta dei servizi di notifica push:

Servizio push per le notifiche	Codice di risposta
Amazon Device Messaging (ADM)	Vedi Formato di risposta nella documentazione ADM.
Apple Push Notification Service (APN)	Vedi Risposta HTTP/2 dagli APN in Comunicazione con gli APN nella Guida per la programmazione delle notifiche locali e in remoto.
Firebase Cloud Messaging (FCM)	Vedi Codici di risposta a messaggi di errore downstream nella documentazione di Firebase Cloud Messaging.
Servizio di notifica push Microsoft per Windows Phone (MPNS)	Vedi Push Notification Service Response Codes for Windows Phone 8 nella documentazione di Windows 8 Development.
Windows Push Notification Services (WNS)	Vedi "Codici di risposta" in Intestazioni delle richieste e delle risposte per il servizio di notifica Push (app di Windows Runtime) nella documentazione di sviluppo per Windows 8.

Eventi app per dispositivi mobili

Amazon SNS aiuta a capire come attivare le notifiche quando si verificano determinati eventi applicazione. Puoi intraprendere alcune azioni programmatiche su quell'evento. L'applicazione deve includere il supporto per un servizio di notifica push come Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) e Windows Push Notification Services (WNS). Puoi impostare le notifiche degli eventi dell'applicazione utilizzando la console Amazon SNS o AWS CLI gli AWS SDK.

Argomenti

- Eventi applicazione disponibili
- Invio di notifiche push per dispositivi mobili

Eventi applicazione disponibili

Le notifiche eventi applicazione tengono traccia degli eventi di creazione, eliminazione e aggiornamento dei singoli endpoint della piattaforma, nonché degli errori di consegna. Di seguito sono elencati i nomi degli attributi per gli eventi applicazione.

Nome attributo	Trigger di notifica
`EventEndpointCreated`	Viene aggiunto un nuovo endpoint della piattaforma all'applicazione.
`EventEndpointDeleted`	Viene eliminato un endpoint della piattaforma associato all'applicazione.
`EventEndpointUpdated`	Viene modificato un attributo degli endpoint della piattaforma associati all'applicazione.
`EventDeliveryFailure`	Una consegna a un qualsiasi endpoint della piattaforma associato all'applicazione restituisce un errore permanente. **Note** Per tenere traccia degli errori di consegna relativamente alle applicazioni della piattaforma, effettua la sottoscrizione agli eventi sullo stato di consegna dei messaggi per l'applicazione. Per ulteriori informazioni, consulta la pagina sull'utilizzo degli attributi di applicazione di Amazon SNS per lo stato di consegna dei messaggi.

È possibile associare qualsiasi attributo a un'applicazione che può quindi ricevere le notifiche di eventi.

Invio di notifiche push per dispositivi mobili

Per inviare notifiche di eventi dell'applicazione, specifica un argomento per ricevere le notifiche per ciascun tipo di evento. Come Amazon SNS invia le notifiche, l'argomento può indirizzarle agli endpoint che intraprenderanno un'azione programmatica.

> **Important**
>
> Le applicazioni a volume elevato creeranno un gran numero di notifiche di eventi dell'applicazione (ad esempio, decine di migliaia), che sovraccaricheranno gli endpoint destinati all'uso umano, come indirizzi e-mail, numeri di telefono e applicazioni mobili. Considera le seguenti linee guida quando invii notifiche di eventi dell'applicazione a un argomento:
>
> - Ogni argomento che riceve notifiche deve contenere solo sottoscrizioni per endpoint programmatici, come endpoint HTTP o HTTPS, code o funzioni Amazon SQS. AWS Lambda
> - Per ridurre la quantità di elaborazione attivata dalle notifiche, limitare le sottoscrizioni di ciascun argomento a un numero ridotto (ad esempio, cinque o meno).

Puoi inviare notifiche sugli eventi dell'applicazione utilizzando la console Amazon SNS, AWS Command Line Interface (AWS CLI) o gli AWS SDK.

AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegli Mobile (Dispositivi mobili), Push notifications (Notifiche push).
3. Nella pagina delle notifiche push per dispositivi mobili, nella sezione Applicazioni della piattaforma, scegli un'applicazione, quindi scegli Modifica.
4. Espandere la sezione Event notifications (Notifiche evento).
5. Seleziona Actions (Azioni), Configure events (Configura eventi).
6. Inserire gli ARN per gli argomenti da utilizzare per i seguenti eventi:
   - Creazione endpoint
   - Eliminazione endpoint
   - Aggiornamento endpoint
   - Errore di consegna
7. Seleziona Save changes (Salva modifiche).

AWS CLI

Esegui il comando set-platform-application-attributes.

L'esempio seguente imposta lo stesso argomento Amazon SNS per tutti e quattro gli eventi dell'applicazione:

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDK

Imposta le notifiche degli eventi dell'applicazione inviando una `SetPlatformApplicationAttributes` richiesta con l'API Amazon SNS utilizzando AWS un SDK.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, tra cui assistenza per iniziare e informazioni sulle versioni precedenti, consulta. Utilizzo di Amazon SNS con un SDK AWS

Operazioni API push per dispositivi mobili

Per usare le API push per dispositivi mobili Amazon SNS devi disporre dei prerequisiti per il servizio di notifica push, come ad esempio Apple Push Notification Service (APNs) e Firebase Cloud Messaging (FCM). Per ulteriori informazioni sui prerequisiti, consulta Prerequisiti per le notifiche utente Amazon SNS.

Per inviare un messaggio di notifica push a un'app e a un dispositivo mobile utilizzando le API, devi prima utilizzare l'operazione `CreatePlatformApplication`, che restituirà un attributo `PlatformApplicationArn`. L'attributo `PlatformApplicationArn` viene poi utilizzato da `CreatePlatformEndpoint`, che restituisce un attributo `EndpointArn`. Puoi utilizzare l'attributo `EndpointArn` insieme all'operazione `Publish` o inviare un messaggio di notifica a un'app e un

dispositivo mobile oppure è possibile utilizzare l'attributo `EndpointArn` insieme all'operazione `Subscribe` per effettuare la sottoscrizione a un argomento. Per ulteriori informazioni, consulta . Panoramica del processo di notifica utente.

Le API push mobili Amazon SNS sono come segue:

`CreatePlatformApplication`

Crea un oggetto applicazione di piattaforma per uno dei servizi di notifica push supportati, come APN e FCM, ai quali i dispositivi e le app mobili possono registrarsi. Restituisce un attributo `PlatformApplicationArn`, utilizzato dall'operazione `CreatePlatformEndpoint`.

`CreatePlatformEndpoint`

Crea un endpoint per un dispositivo e un'app mobile su uno dei servizi di notifica push supportati. `CreatePlatformEndpoint` utilizza l'attributo `PlatformApplicationArn` restituito dall'operazione `CreatePlatformApplication`. L'attributo `EndpointArn` restituito quando si utilizza `CreatePlatformEndpoint` viene poi utilizzato insieme all'operazione `Publish` per inviare un messaggio di notifica a un'app e a un dispositivo mobile.

`CreateTopic`

Crea un argomento su cui e possibile pubblicare i messaggi.

`DeleteEndpoint`

Elimina l'endpoint per un dispositivo e un'app mobile su uno dei servizi di notifica push supportati.

`DeletePlatformApplication`

Elimina un oggetto dell'applicazione di piattaforma.

`DeleteTopic`

Elimina un argomento e tutte le sue sottoscrizioni.

`GetEndpointAttributes`

Recupera gli attributi endpoint per un dispositivo e un'app mobile.

`GetPlatformApplicationAttributes`

Recupera gli attributi dell'oggetto della piattaforma.

`ListEndpointsByPlatformApplication`

Elenca gli endpoint e gli attributi degli endpoint per dispositivi e app mobili in un servizio di notifica push supportato.

`ListPlatformApplications`

Elenca gli oggetti dell'applicazione di piattaforma per i servizi di notifica push supportati.

`Publish`

Invia un messaggio di notifica a tutti gli endpoint sottoscritti di un argomento.

`SetEndpointAttributes`

Imposta gli attributi di un endpoint per un dispositivo e un'app mobile.

`SetPlatformApplicationAttributes`

Imposta gli attributi dell'oggetto della piattaforma.

`Subscribe`

Prepara la sottoscrizione di un endpoint inviando all'endpoint un messaggio di conferma. Per creare una sottoscrizione, il proprietario dell'endpoint deve chiamare l'operazione ConfirmSubscription con il token dal messaggio di conferma.

`Unsubscribe`

Elimina una sottoscrizione.

Errori dell'API per dispositivi mobili

Gli errori restituiti dalle API di Amazon SNS per push per dispositivi mobili sono elencati nella tabella seguente. Per ulteriori informazioni sulle API di Amazon SNS per push per dispositivi mobili, consulta Operazioni API push per dispositivi mobili.

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Application Name is null string (Nome applicazione è una stringa null)	Il nome di applicazione richiesto è impostato su null.	400	`CreatePlatformApplication`
Platform Name is null string (Nome piattaforma è una stringa null)	Il nome di piattaforma richiesto è impostato su null.	400	`CreatePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Platform Name is invalid (Nome piattaforma non valido)	Un valore non valido o fuori intervallo è stato fornito per il nome di piattaforma.	400	`CreatePlatformApplication`
APN - Principal is not a valid certificate (APn - Il principale non è un certificato valido)	Un certificato non valido è stato fornito per il principale APN, ovvero il certificato SSL. Per informazioni, consulta CreateTopic in Riferimento API di Amazon Simple Notification Service nella Guida di riferimento API Amazon Simple Notification Service.	400	`CreatePlatformApplication`
APN - Principal is a valid cert but not in a .pem format (APN - Il principale è un certificato valido ma non in formato .pem)	Un certificato valido non in formato .pem è stato fornito per il principale APN, ovvero il certificato SSL.	400	`CreatePlatformApplication`
APN - Prinicipal is an expired certificate (APN - Il principale è un certificato scaduto)	Un certificato scaduto è stato fornito per il principale APN, ovvero il certificato SSL.	400	`CreatePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
APN - Principal is not an Apple issued certificate (APN - Il principale non è un certificato emesso da Apple)	Un certificato non Apple è stato fornito per il principale APN, ovvero il certificato SSL.	400	`CreatePlatformApplication`
APN - Principal is not provided (APN - Principale non fornito)	Il principale APN, ovvero il certificato SSL, non è stato fornito.	400	`CreatePlatformApplication`
APN - Credential is not provided (APN; - Credenziale non fornita)	La credenziale APN, ovvero la chiave privata, non è stata fornita. Per informazioni, consulta CreateTopic in Riferimento API di Amazon Simple Notification Service nella Guida di riferimento API Amazon Simple Notification Service.	400	`CreatePlatformApplication`
APN - Credential are not in a valid .pem format (APN - Credenziale non in formato .pem valido)	La credenziale APN, ovvero la chiave privata, non è in un formato .pem valido.	400	`CreatePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
FCM — serverAPIKey is not provided (FCM - serverAPIKey non viene fornito)	La credenziale FCM, ovvero la chiave API, non è stata fornita. Per informazioni, consulta CreateTopic in Riferimento API di Amazon Simple Notification Service nella Guida di riferimento API Amazon Simple Notification Service.	400	`CreatePlatformApplication`
FCM — serverAPIKey è vuoto	La credenziale FCM, ovvero la chiave API, è vuota.	400	`CreatePlatformApplication`
FCM — serverAPIKey is a null string (FCM - serverAPIKey è una stringa null string)	La credenziale FCM, ovvero la chiave API, è null.	400	`CreatePlatformApplication`
FCM — serverAPIKey is invalid (serverAPIKey non valido)	La credenziale FCM, ovvero la chiave API, non è valida.	400	`CreatePlatformApplication`
ADM - clientsecret is not provided (ADM - clientsecret non fornita)	Il segreto client richiesto non è fornito.	400	`CreatePlatformApplication`
ADM - clientsecret is a null string (ADM - clientsecret è una stringa null)	La stringa richiesta per il segreto client è null.	400	`CreatePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
ADM - client_secret is empty string (ADM - client_secret è una stringa vuota)	La stringa richiesta per il segreto client è vuota.	400	`CreatePlatformApplication`
ADM - client_secret is not valid (ADM - client_secret non valida)	La stringa richiesta per il segreto client non è valida.	400	`CreatePlatformApplication`
ADM - client_id is empty string (ADM - client_id è una stringa vuota)	La stringa richiesta per l'ID client è vuota.	400	`CreatePlatformApplication`
ADM - clientId is not provided (ADM - clientId non fornita)	La stringa richiesta per l'ID client non è fornita.	400	`CreatePlatformApplication`
ADM - clientid is a null string (ADM - clientid è una stringa null)	La stringa richiesta per l'ID client è null.	400	`CreatePlatformApplication`
ADM - client_id is not valid (ADM - client_id non valida)	La stringa richiesta per l'ID client non è valida.	400	`CreatePlatformApplication`
EventEndpointCreated has invalid ARN format (Formato ARN di EventEndpointCreated non valido)	Il formato ARN di EventEndpointCreated non è valido.	400	`CreatePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventEndpointDeleted has invalid ARN format (Formato ARN di EventEndpointDeleted non valido)	Il formato ARN di EventEndpointDeleted non è valido.	400	`CreatePlatformApplication`
EventEndpointUpdated has invalid ARN format (Formato ARN di EventEndpointUpdated non valido)	Il formato ARN di EventEndpointUpdated non è valido.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure has invalid ARN format (Formato ARN di EventDeliveryAttemptFailure non valido)	Il formato ARN di EventDeliveryAttemptFailure non è valido.	400	`CreatePlatformApplication`
EventDeliveryFailure has invalid ARN format (Formato ARN di EventDeliveryFailure non valido)	Il formato ARN di EventDeliveryFailure non è valido.	400	`CreatePlatformApplication`
EventEndpointCreated is not an existing Topic (EventEndpointCreated non è un argomento esistente)	EventEndpointCreated non è un argomento esistente.	400	`CreatePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventEndpointDeleted is not an existing Topic (EventEndpointDeleted non è un argomento esistente)	EventEndpointDeleted non è un argomento esistente.	400	`CreatePlatformApplication`
EventEndpointUpdated is not an existing Topic (EventEndpointUpdated non è un argomento esistente)	EventEndpointUpdated non è un argomento esistente.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure is not an existing Topic (EventDeliveryAttemptFailure non è un argomento esistente)	EventDeliveryAttemptFailure non è un argomento esistente.	400	`CreatePlatformApplication`
EventDeliveryFailure is not an existing Topic (EventDeliveryFailure non è un argomento esistente)	EventDeliveryFailure non è un argomento esistente.	400	`CreatePlatformApplication`
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	`SetPlatformAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	400	`SetPlatformAttributes`
APN - Principal is not a valid certificate (APn - Il principale non è un certificato valido)	Un certificato non valido è stato fornito per il principale APN, ovvero il certificato SSL. Per informazioni, consulta CreateTopic in Riferimento API di Amazon Simple Notification Service nella Guida di riferimento API Amazon Simple Notification Service.	400	`SetPlatformAttributes`
APN - Principal is a valid cert but not in a .pem format (APN - Il principale è un certificato valido ma non in formato .pem)	Un certificato valido non in formato .pem è stato fornito per il principale APN, ovvero il certificato SSL.	400	`SetPlatformAttributes`
APN - Prinicipal is an expired certificate (APN - Il principale è un certificato scaduto)	Un certificato scaduto è stato fornito per il principale APN, ovvero il certificato SSL.	400	`SetPlatformAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
APN - Principal is not an Apple issued certificate (APN - Il principale non è un certificato emesso da Apple)	Un certificato non Apple è stato fornito per il principale APN, ovvero il certificato SSL.	400	`SetPlatformAttributes`
APN - Principal is not provided (APN - Principale non fornito)	Il principale APN, ovvero il certificato SSL, non è stato fornito.	400	`SetPlatformAttributes`
APN - Credential is not provided (APN; - Credenziale non fornita)	La credenziale APN, ovvero la chiave privata, non è stata fornita. Per informazioni, consulta CreateTopic in Riferimento API di Amazon Simple Notification Service nella Guida di riferimento API Amazon Simple Notification Service.	400	`SetPlatformAttributes`
APN - Credential are not in a valid .pem format (APN - Credenziale non in formato .pem valido)	La credenziale APN, ovvero la chiave privata, non è in un formato .pem valido.	400	`SetPlatformAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
FCM — serverAPIKey is not provided (FCM - serverAPIKey non viene fornito)	La credenziale FCM, ovvero la chiave API, non è stata fornita. Per informazioni, consulta CreateTopic in Riferimento API di Amazon Simple Notification Service nella Guida di riferimento API Amazon Simple Notification Service.	400	`SetPlatformAttributes`
FCM — serverAPIKey is a null string (FCM - serverAPIKey è una stringa null string)	La credenziale FCM, ovvero la chiave API, è null.	400	`SetPlatformAttributes`
ADM - clientId is not provided (ADM - clientId non fornita)	La stringa richiesta per l'ID client non è fornita.	400	`SetPlatformAttributes`
ADM - clientid is a null string (ADM - clientid è una stringa null)	La stringa richiesta per l'ID client è null.	400	`SetPlatformAttributes`
ADM - clientsecret is not provided (ADM - clientsecret non fornita)	Il segreto client richiesto non è fornito.	400	`SetPlatformAttributes`
ADM - clientsecret is a null string (ADM - clientsecret è una stringa null)	La stringa richiesta per il segreto client è null.	400	`SetPlatformAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventEndpointUpdated has invalid ARN format (Formato ARN di EventEndpointUpdated non valido)	Il formato ARN di EventEndpointUpdated non è valido.	400	`SetPlatformAttributes`
EventEndpointDeleted has invalid ARN format (Formato ARN di EventEndpointDeleted non valido)	Il formato ARN di EventEndpointDeleted non è valido.	400	`SetPlatformAttributes`
EventEndpointUpdated has invalid ARN format (Formato ARN di EventEndpointUpdated non valido)	Il formato ARN di EventEndpointUpdated non è valido.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure has invalid ARN format (Formato ARN di EventDeliveryAttemptFailure non valido)	Il formato ARN di EventDeliveryAttemptFailure non è valido.	400	`SetPlatformAttributes`
EventDeliveryFailure has invalid ARN format (Formato ARN di EventDeliveryFailure non valido)	Il formato ARN di EventDeliveryFailure non è valido.	400	`SetPlatformAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventEndpointCreated is not an existing Topic (EventEndpointCreated non è un argomento esistente)	EventEndpointCreated non è un argomento esistente.	400	`SetPlatformAttributes`
EventEndpointDeleted is not an existing Topic (EventEndpointDeleted non è un argomento esistente)	EventEndpointDeleted non è un argomento esistente.	400	`SetPlatformAttributes`
EventEndpointUpdated is not an existing Topic (EventEndpointUpdated non è un argomento esistente)	EventEndpointUpdated non è un argomento esistente.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure is not an existing Topic (EventDeliveryAttemptFailure non è un argomento esistente)	EventDeliveryAttemptFailure non è un argomento esistente.	400	`SetPlatformAttributes`
EventDeliveryFailure is not an existing Topic (EventDeliveryFailure non è un argomento esistente)	EventDeliveryFailure non è un argomento esistente.	400	`SetPlatformAttributes`
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	`GetPlatformApplicationAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	403	`GetPlatformApplicationAttributes`
Token specified is invalid (Token specificato non valido)	Il token specificato non è valido.	400	`ListPlatformApplications`
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	`ListEndpointsByPlatformApplication`
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	404	`ListEndpointsByPlatformApplication`
Token specified is invalid (Token specificato non valido)	Il token specificato non è valido.	400	`ListEndpointsByPlatformApplication`
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	`DeletePlatformApplication`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	403	`DeletePlatformApplication`
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	`CreatePlatformEndpoint`
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	404	`CreatePlatformEndpoint`
Token is not specified (Token non specificato)	Il token non è specificato.	400	`CreatePlatformEndpoint`
Token is not of correct length (Lunghezza token non corretta)	La lunghezza del token non è corretta.	400	`CreatePlatformEndpoint`
Customer User data is too large (Dimensione dati utente cliente troppo grande)	La dimensione dei dati utente cliente non può essere superiore a 2048 byte nella codifica UTF-8.	400	`CreatePlatformEndpoint`
Endpoint ARN is invalid (ARN endpoint non valido)	L'ARN dell'endpoint non è valido.	400	`DeleteEndpoint`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Endpoint ARN is valid but does not belong to the user (ARN endpoint valido ma non appartiene all'utente)	L'ARN dell'endpoint è valido ma non appartiene all'utente.	403	`DeleteEndpoint`
Endpoint ARN is invalid (ARN endpoint non valido)	L'ARN dell'endpoint non è valido.	400	`SetEndpointAttributes`
Endpoint ARN is valid but does not belong to the user (ARN endpoint valido ma non appartiene all'utente)	L'ARN dell'endpoint è valido ma non appartiene all'utente.	403	`SetEndpointAttributes`
Token is not specified (Token non specificato)	Il token non è specificato.	400	`SetEndpointAttributes`
Token is not of correct length (Lunghezza token non corretta)	La lunghezza del token non è corretta.	400	`SetEndpointAttributes`
Customer User data is too large (Dimensione dati utente cliente troppo grande)	La dimensione dei dati utente cliente non può essere superiore a 2048 byte nella codifica UTF-8.	400	`SetEndpointAttributes`
Endpoint ARN is invalid (ARN endpoint non valido)	L'ARN dell'endpoint non è valido.	400	`GetEndpointAttributes`

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Endpoint ARN is valid but does not belong to the user (ARN endpoint valido ma non appartiene all'utente)	L'ARN dell'endpoint è valido ma non appartiene all'utente.	403	`GetEndpointAttributes`
Target ARN is invalid (ARN destinazione non valido)	L'ARN di destinazione non è valido.	400	`Publish`
Target ARN is valid but does not belong to the user (ARN destinazione valido ma non appartiene all'utente)	L'ARN di destinazione è valido ma non appartiene all'utente.	403	`Publish`
Message format is invalid (Formato messaggio non valido)	Il formato del messaggio non è valido.	400	`Publish`
Message size is larger than supported by protocol/end-service (Dimensione messaggio superiore a quella supportata da protocollo/servizio finale)	La dimensione del messaggio è superiore a quella supportata dal protocollo/servizio finale.	400	`Publish`

Utilizzo dell'attributo di messaggio TTL (Time To Live) Amazon SNS per notifiche push per dispositivi mobili

Amazon Simple Notification Service (Amazon SNS) ti aiuta a impostare un attributo messaggio TTL (Time To Live) per messaggi di notifica push per dispositivi mobili. Questa funzionalità si aggiunge alla capacità esistente di impostare il TTL all'interno del corpo del messaggio Amazon SNS per i servizi di notifica push mobili che lo supportano, come Amazon Device Messaging (ADM) e Firebase Cloud Messaging (FCM) durante l'invio ad Android.

L'attributo di messaggio TTL è utilizzato per specificare i metadati di scadenza strutturati su un messaggio. Questo ti consente di specificare il tempo a disposizione del servizio di notifica push, come Apple Push Notification Service (APNs) o FCM, per consegnare il messaggio all'endpoint. Se per qualche motivo (per esempio il dispositivo mobile è stato disattivato) il messaggio non può essere consegnato entro il TTL specificato, verrà eliminato senza ulteriori tentativi di consegna. Per specificare il TTL all'interno degli attributi dei messaggi, puoi utilizzare i kit di sviluppo AWS software (SDK) o l' AWS Management Console API di interrogazione.

Argomenti

Gli attributi di messaggio TTL riservati ai servizi di notifica push

Di seguito è riportato un elenco degli attributi dei messaggi TTL per i servizi di notifica push che è possibile utilizzare per impostare quando si utilizzano gli AWS SDK o l'API di interrogazione:

Servizio di notifiche push	Attributo di messaggio TTL
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple Push Notification Service (APNs)	`AWS.SNS.MOBILE.APNS.TTL`
Apple Push Notification Service Sandbox (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`

Servizio di notifiche push	Attributo di messaggio TTL
Firebase Cloud Messaging (FCM per l'invio ad Android)	`AWS.SNS.MOBILE.FCM.TTL`
Windows Push Notification Services (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Ogni servizio di notifica push gestisce il TTL in modo diverso. Amazon SNS offre una vista astratta del TTL su tutti i servizi di notifica push, il che rende più semplice specificare il TTL. Quando utilizzi il TTL AWS Management Console per specificare (in secondi), devi inserire il valore TTL una sola volta e Amazon SNS calcolerà il TTL per ciascuno dei servizi di notifica push selezionati durante la pubblicazione del messaggio.

Il TTL è relativo all'ora di pubblicazione. Prima di consegnare un messaggio di notifica push a un servizio di notifica push specifico, Amazon SNS calcola il tempo di sosta (il tempo tra il timestamp di pubblicazione e l'istante immediatamente precedente il passaggio a un servizio di notifica push) per la notifica push e passa il TTL rimanente al servizio di notifica push specifico. Se il TTL è più breve del tempo di sosta, Amazon SNS non proverà a pubblicare.

Se specifichi un TTL per un messaggio di notifica push, il valore TTL deve essere un numero intero positivo, a meno che il valore di non `0` abbia un significato specifico per il servizio di notifica push, ad esempio con APN e FCM (quando si invia ad Android). Se il valore TTL è impostato su `0` e il servizio di notifica push non ha un significato specifico per `0`, Amazon SNS non consegnerà il messaggio. Per ulteriori informazioni sul parametro TTL impostato su `0` quando si utilizza APN, consulta Tabella A-3 identificatori elemento per notifiche remote all'interno della documentazione API provider binario.

Ordine di precedenza per determinare il TTL

La precedenza che Amazon SNS utilizza per determinare il TTL per un messaggio di notifica push è basata sul seguente ordine, in cui il numero più basso ha la priorità più alta:

1. TTL attributo di messaggio
2. TTL testo messaggio
3. TTL di default per il servizio di notifica push (varia in base al servizio)
4. TTL di default Amazon SNS (4 settimane)

Se imposti diversi valori TTL (uno negli attributi di messaggio e un altro nel corpo del messaggio) per lo stesso messaggio, Amazon SNS modificherà il TTL nel corpo del messaggio in modo che corrisponda al TTL specificato nell'attributo del messaggio stesso.

Specificare il TTL utilizzando AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegli Mobile (Dispositivi mobili), Push notifications (Notifiche push).
3. Nella pagina Mobile push notifications (Notifiche push per dispositivi mobili), nella sezione Platform applications (Applicazioni di piattaforma), selezionare un'applicazione.
4. Nella ***MyApplication***pagina, nella sezione Endpoints, scegli un endpoint dell'applicazione, quindi scegli Pubblica messaggio.
5. Nella sezione Message details (Dettagli messaggio), immettere il TTL (numero di secondi a disposizione del servizio di notifica push per recapitare il messaggio all'endpoint).
6. Seleziona Publish message (Pubblica messaggio).

Regioni supportate per applicazioni mobili

Al momento, puoi creare applicazioni per dispositivi mobili nelle seguenti regioni:

- Stati Uniti orientali (Ohio)
- Stati Uniti orientali (Virginia settentrionale)
- Stati Uniti occidentali (California settentrionale)
- Stati Uniti occidentali (Oregon)
- Africa (Città del Capo)
- Asia Pacifico (Hong Kong)
- Asia Pacifico (Giacarta)
- Asia Pacifico (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)

- Canada (Centrale)
- Europa (Francoforte)
- Europa (Irlanda)
- Europa (Londra)
- Europa (Milano)
- Europa (Parigi)
- Europa (Stoccolma)
- Medio Oriente (Bahrein)
- Medio Oriente (Emirati Arabi Uniti)
- Sud America (São Paulo)
- AWS GovCloud (US-West)

Best practice per le notifiche push per dispositivi mobili

Questa sezione illustra diverse best practice che potrebbero rivelarsi utili per migliorare il coinvolgimento dei clienti.

Gestione di endpoint

I problemi di consegna potrebbero verificarsi in situazioni in cui i token del dispositivo cambiano a causa dell'azione di un utente sul dispositivo (ad esempio un'app viene reinstallata sul dispositivo), oppure aggiornamenti dei certificati che interessano i dispositivi in esecuzione su una particolare versione iOS. Questa non è una best practice consigliata da Apple per registrarsi con APN ogni volta che viene avviata la tua app.

Poiché il token del dispositivo non cambia ogni volta che un'app viene aperta da un utente, è possibile utilizzare l'API idempotente `CreatePlatformEndpoint`. Tuttavia, ciò può introdurre duplicati per lo stesso dispositivo nei casi in cui il token stesso non è valido o se l'endpoint è valido ma disabilitato (ad esempio, una mancata corrispondenza tra gli ambienti di produzione e sandbox).

Può essere utilizzato un meccanismo di gestione dei token del dispositivo, come quello nello pseudocodice.

Per informazioni sulla gestione e la manutenzione dei token dei dispositivi FCM v1, vedere. Gestione degli endpoint Firebase Cloud Messaging (FCM)

Registrazione dello stato della consegna

Per monitorare lo stato di consegna delle notifiche push, ti consigliamo di abilitare la registrazione dello stato di consegna per l'applicazione della piattaforma Amazon SNS. In questo modo è possibile risolvere i problemi di consegna, poiché i log contengono codici di risposta di provider restituiti dal servizio push della piattaforma. Per informazioni dettagliate sull'abilitazione della registrazione dello stato di consegna, vedere Come posso accedere ai registri di consegna degli argomenti Amazon SNS per le notifiche push?.

Notifiche degli eventi

Per gestire gli endpoint in modo che siano basati su eventi, è possibile utilizzare la funzionalità Notifiche eventi. Ciò consente all'argomento Amazon SNS configurato di eseguire il fanout degli eventi agli abbonati, come una funzione Lambda, per eventi applicativi della piattaforma di creazione, eliminazione, aggiornamenti e errori di consegna degli endpoint.

Notifiche e-mail

Questa pagina descrive come iscrivere un indirizzo e-mail a un argomento di Amazon SNS utilizzando AWS Management Console AWS SDK for Java, o. AWS SDK for .NET

> **Note**
>
> - Non è possibile personalizzare il testo del messaggio e-mail. La funzione di recapito e-mail ha lo scopo di fornire avvisi interni di sistema, non messaggi di marketing.
> - La sottoscrizione diretta agli endpoint di e-mail è supportata solo per gli argomenti standard.
> - La velocità effettiva di recapito delle e-mail è regolata in base alle Quote Amazon SNS.

> **Important**
>
> Per impedire ai destinatari della mailing list di annullare la sottoscrizione di tutti i destinatari dalle e-mail dell'argomento Amazon SNS, consulta Configurare un abbonamento e-mail che richiede l'autenticazione per annullare l'iscrizione da AWS Support.

Per iscrivere un indirizzo e-mail a un argomento di Amazon SNS utilizzando il AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione sinistro scegli Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
   a. Per ARN argomento scegli l'Amazon Resource Name (ARN) di un argomento.
   b. Per Protocol, scegliere Email.
   c. Per Endpoint, immettere l'indirizzo e-mail.
   d. (Facoltativo) Per configurare un criterio filtro, espandere la sezione Policy di filtro per sottoscrizione. Per ulteriori informazioni, consulta Policy di filtro degli abbonamenti Amazon SNS.
   e. (Facoltativo) Per abilitare il filtro basato sul payload, imposta `Filter Policy Scope` su `MessageBody`. Per ulteriori informazioni, consulta Ambito delle policy di filtro per le sottoscrizioni Amazon SNS.
   f. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta Code DLQ (DLQ) Amazon SNS.
   g. Scegli Create Subscription (Crea sottoscrizione).

      La console crea la sottoscrizione e apre la pagina Dettagli.

È necessario confermare la sottoscrizione prima che l'indirizzo e-mail possa iniziare a ricevere messaggi.

Per confermare una sottoscrizione

1. Controlla la tua casella di posta elettronica e scegli Conferma sottoscrizione nell'e-mail di Amazon SNS.
2. Amazon SNS apre il browser Web e visualizza una conferma dell'abbonamento con il tuo ID abbonamento.

Per iscrivere un indirizzo e-mail a un argomento di Amazon SNS utilizzando un SDK AWS

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta File di configurazione e credenziali condivisi nella Guida di riferimento per SDK AWS e strumenti.

I seguenti esempi di codice mostrano come utilizzare. `Subscribe`

.NET

AWS SDK for .NET

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };
```

```
            var response = await client.SubscribeAsync(request);

            return response;
        }
```

Iscrivi una coda a un argomento con filtri opzionali.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for .NET .

C++

SDK per C++

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
```

```
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Sottoscrivi un'applicazione mobile a un argomento.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

Sottoscrivi una funzione Lambda a un argomento.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

Sottoscrivi una coda SQS a un argomento.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

Iscriviti con un filtro a un argomento.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
```

```
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
```

```
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";
```

```
        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for C++ .

CLI

AWS CLI

Effettuare la sottoscrizione a un argomento

Attraverso il comando `subscribe` seguente viene effettuata la sottoscrizione all'argomento specificato utilizzando un indirizzo e-mail.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Output:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Per informazioni dettagliate sulle API, consulta Subscribe nel Riferimento ai comandi AWS CLI .

Go

SDK per Go V2

> **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.

Iscrivi una coda a un argomento con filtri opzionali.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for Go .

Java

SDK for 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.

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 .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Sottoscrivi un'applicazione mobile a un argomento.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
```

```
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Sottoscrivi una funzione Lambda a un argomento.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
```

```
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Sottoscrivi una coda SQS a un argomento.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
```

```
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Iscriviti con un filtro a un argomento.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for JavaScript .

Kotlin

SDK per Kotlin

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
```

```
    }
}
```

Sottoscrivi una funzione Lambda a un argomento.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento degli SDK AWS per Kotlin.

PHP

SDK for PHP

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Sottoscrivi un endpoint HTTP a un argomento.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for PHP .

Python

SDK for Python (Boto3)

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
```

```
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API dell'SDK for Python (Boto3)AWS .

Ruby

SDK for Ruby

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
```

```
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for Ruby .

Rust

SDK per Rust

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per SAP ABAP

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta Subscribe nella documentazione di riferimento dell'SDK AWS per l'API SAP ABAP.

Esempi di codice per Amazon SNS che utilizzano SDK AWS

I seguenti esempi di codice mostrano come usare Amazon SNS con un kit di sviluppo AWS software (SDK).

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le operazioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati e negli esempi tra servizi.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica richiamando più funzioni all'interno dello stesso servizio.

Esempi cross-service: applicazioni di esempio che funzionano su più servizi Servizi AWS.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Nozioni di base

Hello Amazon SNS

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon SNS.

.NET

AWS SDK for .NET

> **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.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
```

```
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

Codice per il file CMake C MakeLists .txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)
```

```
# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine hello_sns.cpp.

```
#include <aws/core/Aws.h>
```

```
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
```

```
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Per i dettagli sull'API, consulta API ListTopicsReference AWS SDK for C++ .

Go

SDK per Go V2

> **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 main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
```

```
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for Go API Reference.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Inizializza un client SNS ed elenca gli argomenti nel tuo account.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});
```

```
  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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 aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform
```

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- Per i dettagli sull'API, ListTopicsconsulta AWS SDK for Kotlin API reference.

Esempi di codice

- Azioni per Amazon SNS tramite SDK AWS
  - Utilizzo CheckIfPhoneNumberIsOptedOut con un AWS SDK o una CLI
  - Utilizzo ConfirmSubscription con un AWS SDK o una CLI
  - Utilizzo CreateTopic con un AWS SDK o una CLI
  - Utilizzo DeleteTopic con un AWS SDK o una CLI
  - Utilizzo GetSMSAttributes con un AWS SDK o una CLI
  - Utilizzo GetTopicAttributes con un AWS SDK o una CLI
  - Utilizzo ListPhoneNumbersOptedOut con un AWS SDK o una CLI
  - Utilizzo ListSubscriptions con un AWS SDK o una CLI
  - Utilizzo ListTopics con un AWS SDK o una CLI
  - Utilizzo Publish con un AWS SDK o una CLI
  - Utilizzo SetSMSAttributes con un AWS SDK o una CLI

- Utilizzo SetSubscriptionAttributes con un AWS SDK o una CLI
- Utilizzo SetSubscriptionAttributesRedrivePolicy con un AWS SDK o una CLI
- Utilizzo SetTopicAttributes con un AWS SDK o una CLI
- Utilizzo Subscribe con un AWS SDK o una CLI
- Utilizzo TagResource con un AWS SDK o una CLI
- Utilizzo Unsubscribe con un AWS SDK o una CLI

- Scenari per Amazon SNS che utilizzano SDK AWS
  - Crea un endpoint di piattaforma per le notifiche push di Amazon SNS utilizzando un SDK AWS
  - Crea e pubblica su un argomento FIFO Amazon SNS utilizzando un SDK AWS
  - Pubblica messaggi SMS su un argomento Amazon SNS utilizzando un SDK AWS
  - Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un SDK AWS
  - Pubblica un messaggio di testo SMS Amazon SNS utilizzando un SDK AWS
  - Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS
- Esempi serverless per Amazon SNS che utilizzano SDK AWS
  - Richiamo di una funzione Lambda da un trigger Amazon SNS
- Esempi di servizi multipli per Amazon SNS che utilizzano SDK AWS
  - Costruisci un'applicazione per inviare dati a una tabella DynamoDB
  - Costruzione di un'applicazione per la pubblicazione e la sottoscrizione che traduce i messaggi
  - Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette
  - Creazione di un'applicazione Amazon Textract explorer
  - Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS
  - Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS
  - Utilizzo di un'API Gateway per richiamare una funzione Lambda
  - Utilizzo degli eventi pianificati per richiamare una funzione Lambda

Azioni per Amazon SNS tramite SDK AWS

I seguenti esempi di codice mostrano come eseguire singole azioni Amazon SNS con AWS gli SDK. Questi estratti richiamano l'API Amazon RDS e sono estratti di codice da programmi più grandi che

devono essere eseguiti nel contesto. Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Gli esempi seguenti includono solo le operazioni più comunemente utilizzate. Per un elenco completo, consultare la Documentazione di riferimento all'API di Servizio di notifica semplice Amazon (Amazon SNS).

Esempi

- Utilizzo CheckIfPhoneNumberIsOptedOut con un AWS SDK o una CLI
- Utilizzo ConfirmSubscription con un AWS SDK o una CLI
- Utilizzo CreateTopic con un AWS SDK o una CLI
- Utilizzo DeleteTopic con un AWS SDK o una CLI
- Utilizzo GetSMSAttributes con un AWS SDK o una CLI
- Utilizzo GetTopicAttributes con un AWS SDK o una CLI
- Utilizzo ListPhoneNumbersOptedOut con un AWS SDK o una CLI
- Utilizzo ListSubscriptions con un AWS SDK o una CLI
- Utilizzo ListTopics con un AWS SDK o una CLI
- Utilizzo Publish con un AWS SDK o una CLI
- Utilizzo SetSMSAttributes con un AWS SDK o una CLI
- Utilizzo SetSubscriptionAttributes con un AWS SDK o una CLI
- Utilizzo SetSubscriptionAttributesRedrivePolicy con un AWS SDK o una CLI
- Utilizzo SetTopicAttributes con un AWS SDK o una CLI
- Utilizzo Subscribe con un AWS SDK o una CLI
- Utilizzo TagResource con un AWS SDK o una CLI
- Utilizzo Unsubscribe con un AWS SDK o una CLI

Utilizzo **`CheckIfPhoneNumberIsOptedOut`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

> **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.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
```

```
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for .NET API Reference.

CLI

AWS CLI

Controllo delle impostazioni di opt-out dei messaggi SMS per un numero di telefono

L'`check-if-phone-number-is-opted-out`esempio seguente verifica se al numero di telefono specificato è stata disattivata la ricezione di messaggi SMS dall' AWS account corrente.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Per i dettagli sull'API, vedere CheckIfPhoneNumberIsOptedOutin AWS CLI Command Reference.

Java

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.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

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

        final String usage = """
```

```
                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
```

```
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per i dettagli sull'API, consulta la CheckIfPhoneNumberIsOptedOutsezione AWS SDK for PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ConfirmSubscription`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`ConfirmSubscription`.

CLI

AWS CLI

Conferma di una sottoscrizione

Attraverso il comando `confirm-subscription` seguente viene completato il processo di conferma avviato effettuando la sottoscrizione a un argomento SNS denominato `my-`

`topic`. Il parametro --token proviene dal messaggio di conferma inviato all'endpoint di notifica specificato nella chiamata subscribe.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --token
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

Output:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Per i dettagli sull'API, consulta ConfirmSubscription AWS CLICommand Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
```

```
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la ConfirmSubscriptionsezione AWS SDK for JavaScript API Reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
```

```
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la ConfirmSubscriptionsezione AWS SDK for PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`CreateTopic`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`CreateTopic`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- Creazione e pubblicazione su un argomento FIFO
- Pubblicazione di messaggi nelle code

.NET

AWS SDK for .NET

> **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 argomento con un nome di specifico.

```
using System;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;
```

```
    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

Crea un nuovo argomento con un nome e attributi FIFO e di deduplicazione specifici.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }

        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- Per i dettagli sull'API, consulta la CreateTopicsezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
```

```
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la CreateTopicsezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Creazione di un argomento SNS

Nell'esempio `create-topic` seguente viene creato un argomento SNS denominato `my-topic`.

```
aws sns create-topic \
    --name my-topic
```

Output:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Per ulteriori informazioni, consulta Using the AWS Command Line Interface with Amazon SQS e Amazon SNS nella Command Line Interface AWS User Guide.

- Per i dettagli sull'API, consulta AWS CLI Command CreateTopicReference.

Go

SDK per Go V2

> **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.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
```

```
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Per i dettagli sull'API, consulta la CreateTopicsezione AWS SDK for Go API Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la CreateTopicsezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- Per i dettagli sull'API, CreateTopicconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per i dettagli sull'API, consulta la CreateTopicsezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- Per i dettagli sull'API, consulta CreateTopic AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **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 class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
```

```
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, consulta la CreateTopicsezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **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.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- Per i dettagli sulle API, consulta la CreateTopicguida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

> **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.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- Per i dettagli sulle API, CreateTopicconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`DeleteTopic`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`DeleteTopic`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Pubblicazione di messaggi nelle code

.NET

AWS SDK for .NET

> **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.

Elimina un argomento in base all'ARN dell'argomento.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Eliminazione di un argomento SNS

Nell'esempio `delete-topic` seguente viene eliminato l'argomento SNS specificato.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta DeleteTopic AWS CLICommand Reference.

Go

SDK per Go V2

> **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.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for Go API Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- Per i dettagli sull'API, DeleteTopicconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la DeleteTopicsezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- Per i dettagli sull'API, consulta DeleteTopic AWSSDK for Python (Boto3) API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per i dettagli sulle API, DeleteTopicconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`GetSMSAttributes`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`GetSMSAttributes`.

C++

SDK per C++

> **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.

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
```

```
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta GetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Elencare gli attributi predefiniti dei messaggi SMS

Nell'esempio `get-sms-attributes` seguente vengono elencati gli attributi predefiniti per l'invio di messaggi SMS.

```
aws sns get-sms-attributes
```

Output:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Per informazioni dettagliate sull'API, consulta GetSMSAttributes nel Riferimento ai comandi AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.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 informazioni dettagliate sulle API, consulta GetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
```

```
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta GetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for JavaScript .

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
```

```
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta GetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for PHP .

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`GetTopicAttributes`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`GetTopicAttributes`.

.NET

AWS SDK for .NET

> **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.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
 topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
```

```
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- Per i dettagli sull'API, consulta la GetTopicAttributessezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la GetTopicAttributessezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Recupero degli attributi di un argomento

Nell'esempio `get-topic-attributes` seguente vengono visualizzati gli attributi per l'argomento specificato.

```
aws sns get-topic-attributes \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

Output:

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "1",
        "DisplayName": "my-topic",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy
\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,
\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,
\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "123456789012",
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID
\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":
\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes
\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-
topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
\"0123456789012\"}}}]}",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
        "SubscriptionsPending": "0"
    }
}
```

- Per i dettagli sull'API, consulta GetTopicAttributes AWS CLICommand Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la GetTopicAttributessezione AWS SDK for JavaScript API Reference.

SDK per JavaScript (v2)

> **Note**
>
> C'è altro da fare. GitHub Trova l'esempio completo e scopri come configurarlo ed eseguirlo nel AWS Code Examples Repository.

Importare l'SDK e i moduli client e chiamare l'API.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
```

```
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la GetTopicAttributessezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {

    val request = GetTopicAttributesRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Per i dettagli sull'API, GetTopicAttributesconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la GetTopicAttributessezione AWS SDK for PHP API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per i dettagli sulle API, GetTopicAttributesconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListPhoneNumbersOptedOut`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Elencare le opzioni di opt-out dei messaggi SMS

Nell'esempio `list-phone-numbers-opted-out` seguente sono elencati i numeri di telefono per i quali sono abilitate le opzioni di opt-out per la ricezione di SMS.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Per i dettagli sull'API, consulta ListPhoneNumbersOptedOut AWS CLICommand Reference.

Java

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.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.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per i dettagli sull'API, consulta la ListPhoneNumbersOptedOutsezione AWS SDK for PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListSubscriptions`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`ListSubscriptions`.

.NET

AWS SDK for .NET

> **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.

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon.SimpleNotificationService;
   using Amazon.SimpleNotificationService.Model;

   /// <summary>
   /// This example will retrieve a list of the existing Amazon Simple
   /// Notification Service (Amazon SNS) subscriptions.
   /// </summary>
   public class ListSubscriptions
   {
       public static async Task Main()
       {
           IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

           Console.WriteLine("Enter a topic ARN to list subscriptions for a
specific topic, " +
                             "or press Enter to list subscriptions for all
topics.");
           var topicArn = Console.ReadLine();
           Console.WriteLine();

           var subscriptions = await GetSubscriptionsListAsync(client,
topicArn);

           DisplaySubscriptionList(subscriptions);
       }

       /// <summary>
```

```
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
            {
                var paginateAllSubscriptions =
 client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
 paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }
```

```
        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- Per i dettagli sull'API, consulta la ListSubscriptionssezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
```

```
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                      << outcome.GetError().GetMessage()
                      <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
```

```
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Per i dettagli sull'API, consulta la ListSubscriptionssezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Elencare le sottoscrizioni SNS

L'`list-subscriptions`esempio seguente mostra un elenco degli abbonamenti SNS presenti nel tuo AWS account.

```
aws sns list-subscriptions
```

Output:

```
{
    "Subscriptions": [
        {
            "Owner": "123456789012",
            "Endpoint": "my-email@example.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
            "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
        }
    ]
}
```

- Per i dettagli sull'API, consulta AWS CLI Command ListSubscriptionsReference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la ListSubscriptionssezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun listSNSSubscriptions() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- Per i dettagli sull'API, ListSubscriptionsconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la ListSubscriptionssezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
```

```
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- Per i dettagli sull'API, consulta ListSubscriptions AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **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 class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = "SNS_TOPIC_ARN" # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, consulta la ListSubscriptionssezione AWS SDK for Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.

```
   TRY.
       oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
       DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
       MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
     CATCH /aws1/cx_rt_generic.
       MESSAGE 'Unable to list subscribers.' TYPE 'E'.
   ENDTRY.
```

- Per i dettagli sulle API, ListSubscriptionsconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`ListTopics`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`ListTopics`.

.NET

AWS SDK for .NET

> **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.

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon.SimpleNotificationService;
   using Amazon.SimpleNotificationService.Model;

   /// <summary>
   /// Lists the Amazon Simple Notification Service (Amazon SNS)
   /// topics for the current account.
   /// </summary>
   public class ListSNSTopics
   {
       public static async Task Main()
       {
           IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

           await GetTopicListAsync(client);
       }

       /// <summary>
       /// Retrieves the list of Amazon SNS topics in groups of up to 100
       /// topics.
       /// </summary>
       /// <param name="client">The initialized Amazon SNS client object used
       /// to retrieve the list of topics.</param>
       public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
       {
           // If there are more than 100 Amazon SNS topics, the call to
           // ListTopicsAsync will return a value to pass to the
           // method to retrieve the next 100 (or less) topics.
           string nextToken = string.Empty;

           do
           {
               var response = await client.ListTopicsAsync(nextToken);
               DisplayTopicsList(response.Topics);
               nextToken = response.NextToken;
           }
           while (!string.IsNullOrEmpty(nextToken));
       }
```

```
        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for .NET API Reference.

C++

SDK per C++

> **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.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;
```

```
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for C++ API Reference.

CLI

AWS CLI

Elenco degli argomenti SNS

L'`list-topics`esempio seguente elenca tutti gli argomenti SNS del tuo AWS account.

```
aws sns list-topics
```

Output:

```
{
```

```
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Per i dettagli sull'API, consulta ListTopics AWS CLICommand Reference.

Go

SDK per Go V2

> **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 main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/sns"
 "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
```

```
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for Go API Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun listSNSTopics() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- Per i dettagli sull'API, ListTopicsconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for PHP API Reference.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- Per i dettagli sull'API, consulta ListTopics AWSSDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

> **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.

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'

def list_topics?(sns_client)
```

```
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, consulta la ListTopicssezione AWS SDK for Ruby API Reference.

Rust

SDK per Rust

> **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.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
    let resp = client.list_topics().send().await?;
```

```
    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- Per i dettagli sulle API, consulta la ListTopicsguida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

> **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.
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- Per i dettagli sulle API, ListTopicsconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`Publish`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`Publish`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- Creazione e pubblicazione su un argomento FIFO
- Pubblicazione di un SMS
- Pubblicazione di messaggi nelle code

.NET

AWS SDK for .NET

> **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.

Pubblicare un messaggio in un argomento.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

Pubblica un messaggio in un argomento con opzioni di gruppo, duplicazione e attributo.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");
```

```
        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);
```

```
                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

Applica le selezioni dell'utente all'azione di pubblicazione.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
```

```
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for .NET .

C++

SDK per C++

> **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.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
```

```
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Pubblica un messaggio con un attributo.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);
```

```
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Esempio 1: pubblicazione di un messaggio in un argomento

Nell'esempio `publish` viene pubblicato il messaggio indicato in un argomento SNS specificato. Il messaggio proviene da un file di testo che consente di includere interruzioni di riga.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

Contenuto di `message.txt`.

```
Hello World
Second Line
```

Output:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

Esempio 2: pubblicare un messaggio SMS su un numero di telefono

Nell'esempio `publish` seguente viene pubblicato il messaggio `Hello world!` sul numero di telefono`+1-555-555-0100`.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

Output:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- Per ulteriori informazioni sulle API, consulta Publish nel Riferimento ai comandi AWS CLI .

Go

SDK per Go V2

> **Note**
>
> C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
```

```
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for Go .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                 if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
```

```
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Pubblica un messaggio in un argomento con opzioni di gruppo, duplicazione e attributo.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
```

```
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
```

```
    }
  }
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for JavaScript .

Kotlin

SDK per Kotlin

> **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.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API di SDK AWS per Kotlin.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for PHP .

PowerShell

Utensili per PowerShell

Esempio 1: Questo esempio mostra la pubblicazione di un messaggio con una sola riga MessageAttribute dichiarata.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

Esempio 2: Questo esempio mostra la pubblicazione di un messaggio con più messaggi MessageAttributes dichiarati in anticipo.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- Per i dettagli sull'API, vedere Publish in AWS Tools for PowerShell Cmdlet Reference.

Python

SDK per Python (Boto3)

> **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.

Pubblicare un messaggio con attributi in modo che una sottoscrizione possa filtrare in base agli attributi.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
```

```
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

Pubblicare un messaggio che assume forme diverse in base al protocollo del sottoscrittore.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
```

```
        :param default_message: The default version of the message. This version
is
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per l'API SDK for Python (Boto3)AWS .

Ruby

SDK per Ruby

> **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.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content
```

```
  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for Ruby .

Rust

SDK per Rust

> **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.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);
```

```
    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta Publish nella documentazione di riferimento dell'SDK AWS per l'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **SetSMSAttributes** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`SetSMSAttributes`.

C++

SDK per C++

> **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.

Come utilizzare Amazon SNS per impostare l'attributo DefaultSMType.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
```

```
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Impostazione degli attributi dei messaggi SMS

Nell'esempio `set-sms-attributes` seguente l'ID mittente predefinito per i messaggi SMS viene impostato su `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Questo comando non produce alcun output.

- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nel Riferimento ai comandi AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
}
```

- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
```

```
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for JavaScript .

PHP

SDK per PHP

> **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.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta SetSMSAttributes nella Documentazione di riferimento per le API AWS SDK for PHP .

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`SetSubscriptionAttributes`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`SetSubscriptionAttributes`.

CLI

AWS CLI

Impostazione degli attributi della sottoscrizione

Nell'esempio `set-subscription-attributes` seguente viene impostato l'attributo `RawMessageDelivery` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene impostato un attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene rimosso l'attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta SetSubscriptionAttributes AWS CLICommand Reference.

Java

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.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.

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- Per i dettagli sull'API, consulta SetSubscriptionAttributes AWSSDK for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **SetSubscriptionAttributesRedrivePolicy** con un AWS SDK o una CLI

Il seguente esempio di codice mostra come utilizzare`SetSubscriptionAttributesRedrivePolicy`.

Java

SDK per Java 1.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.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`SetTopicAttributes`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`SetTopicAttributes`.

CLI

AWS CLI

Impostazione di un attributo per un argomento

Nell'esempio `set-topic-attributes` seguente vengono impostati gli attributi `DisplayName` per l'argomento specificato.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta SetTopicAttributes AWS CLICommand Reference.

Java

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.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.

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
```

```
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per i dettagli sull'API, consulta la SetTopicAttributessezione AWS SDK for JavaScript API Reference.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?)
 {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Per i dettagli sull'API, SetTopicAttributesconsulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la SetTopicAttributessezione AWS SDK for PHP API Reference.

Ruby

SDK per Ruby

> **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.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                   Sid: "__default_statement_ID",
                   Effect: "Allow",
                   Principal: { "AWS": "*" },
                   Action: ["SNS:Publish"],
                   Resource: topic_arn,
                   Condition: {
                     ArnEquals: {
                       "AWS:SourceArn": resource_arn
                     }
                   }
                 }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per i dettagli sull'API, consulta la SetTopicAttributessezione AWS SDK for Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

> **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.

```
TRY.
    lo_sns->settopicattributes(
        iv_topicarn = iv_topic_arn
        iv_attributename  = iv_attribute_name
        iv_attributevalue = iv_attribute_value
    ).
    MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, SetTopicAttributesconsulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`Subscribe`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`Subscribe`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- Creazione e pubblicazione su un argomento FIFO
- Pubblicazione di messaggi nelle code

.NET

AWS SDK for .NET

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

Iscrivi una coda a un argomento con filtri opzionali.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for .NET .

C++

SDK per C++

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Sottoscrivi un'applicazione mobile a un argomento.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Sottoscrivi una funzione Lambda a un argomento.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Sottoscrivi una coda SQS a un argomento.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::SNS::SNSClient snsClient(clientConfiguration);

           Aws::SNS::Model::SubscribeRequest request;
           request.SetTopicArn(topicARN);
           request.SetProtocol("sqs");
           request.SetEndpoint(queueARN);

           Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

           if (outcome.IsSuccess()) {
               Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
               std::cout << "The queue '" << queueName
                         << "' has been subscribed to the topic '"
                         << "'" << topicName << "'" << std::endl;
               std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                         << std::endl;
               subscriptionARNS.push_back(subscriptionARN);
           }
           else {
               std::cerr << "Error with TopicsAndQueues::Subscribe. "
                         << outcome.GetError().GetMessage()
                         << std::endl;

               cleanUp(topicARN,
                       queueURLS,
                       subscriptionARNS,
                       snsClient,
                       sqsClient);

               return false;
           }
```

Sottoscrivi un argomento con un filtro.

```
       static const Aws::String TONE_ATTRIBUTE("tone");
       static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                      "sincere"};
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;
```

```
                    request.AddAttributes("FilterPolicy", jsonPolicy);
                }
                else {
                    std::cout
                            << "Because you did not select any attributes, no
 filter "
                            << "will be added to this subscription." <<
 std::endl;
                }
            }
        }  // if (isFifoTopic)
        Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

        if (outcome.IsSuccess()) {
            Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
            std::cout << "The queue '" << queueName
                      << "' has been subscribed to the topic '"
                      << "'" << topicName << "'" << std::endl;
            std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                      << std::endl;
            subscriptionARNS.push_back(subscriptionARN);
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Subscribe. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
```

```
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";
```

```
        result = jsonPolicyStream.str();
    }

    return result;
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for C++ .

CLI

AWS CLI

Effettuare la sottoscrizione a un argomento

Attraverso il comando `subscribe` seguente viene effettuata la sottoscrizione all'argomento specificato utilizzando un indirizzo e-mail.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Output:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Per informazioni dettagliate sulle API, consulta Subscribe nel Riferimento ai comandi AWS CLI .

Go

SDK per Go V2

> **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.

Iscrivi una coda a un argomento con filtri opzionali.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for Go .

Java

SDK for 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.

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 .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";
```

```
/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

Sottoscrivi un'applicazione mobile a un argomento.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
```

```
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Sottoscrivi una funzione Lambda a un argomento.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
```

```
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Sottoscrivi una coda SQS a un argomento.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
```

```
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

Sottoscrivi un argomento con un filtro.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
```

```
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for JavaScript .

Kotlin

SDK per Kotlin

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
```

```
    }
}
```

Sottoscrivi una funzione Lambda a un argomento.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento degli SDK AWS per Kotlin.

PHP

SDK for PHP

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

Sottoscrivi un endpoint HTTP a un argomento.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for PHP .

Python

SDK for Python (Boto3)

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
```

```
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API dell'SDK for Python (Boto3)AWS .

Ruby

SDK for Ruby

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
```

```
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento sulle API AWS SDK for Ruby .

Rust

SDK per Rust

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- Per informazioni dettagliate sulle API, consulta Sottoscrizione nella Documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per SAP ABAP

> **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.

Sottoscrivi un indirizzo email a un argomento.

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta Subscribe nella documentazione di riferimento dell'SDK AWS per l'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`TagResource`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`TagResource`.

CLI

AWS CLI

Aggiungere un tag a un argomento

Nell'esempio `tag-resource` seguente viene aggiunto un tag di metadati all'argomento Amazon SNS specificato.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta TagResource AWS CLICommand Reference.

Java

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.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.

Kotlin

SDK per Kotlin

> **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.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }

    val tag2 = Tag {
        key = "Environment"
```

```
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- Per i dettagli sull'API, TagResourceconsulta AWS SDK for Kotlin API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **`Unsubscribe`** con un AWS SDK o una CLI

I seguenti esempi di codice mostrano come utilizzare`Unsubscribe`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- Pubblicazione di messaggi nelle code

.NET

AWS SDK for .NET

> **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.

Annulla l'iscrizione a un argomento tramite un ARN di sottoscrizione.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for .NET .

C++

SDK per C++

> **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.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for C++ .

CLI

AWS CLI

Annullamento della sottoscrizione a un argomento

Nell'esempio `unsubscribe` seguente viene eliminata la sottoscrizione specificata a un argomento.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Questo comando non produce alcun output.

- Per informazioni dettagliate sulle API, consulta Unsubscribe nel Riferimento ai comandi AWS CLI .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 .

JavaScript

SDK per JavaScript (v3)

> **Note**
>
> C'è 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 il client in un modulo separato ed esportarlo.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

Importare l'SDK e i moduli client e chiamare l'API.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for JavaScript.
- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for JavaScript .

Kotlin

SDK per Kotlin

> **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.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API di SDK AWS per Kotlin.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per le API AWS SDK for PHP .

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- Per informazioni dettagliate sulle API, consulta Annullamento della sottoscrizione nella Documentazione di riferimento per l'API SDK for Python (Boto3)AWS .

SAP ABAP

SDK per SAP ABAP

> **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.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta Unsubscribe nella documentazione di riferimento dell'SDK AWS per l'API SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per Amazon SNS che utilizzano SDK AWS

I seguenti esempi di codice mostrano come implementare scenari comuni in Amazon SNS con AWS SDK. Questi scenari mostrano come eseguire attività specifiche richiamando più funzioni all'interno di Amazon SNS. Ogni scenario include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire il codice.

Esempi

- Crea un endpoint di piattaforma per le notifiche push di Amazon SNS utilizzando un SDK AWS
- Crea e pubblica su un argomento FIFO Amazon SNS utilizzando un SDK AWS
- Pubblica messaggi SMS su un argomento Amazon SNS utilizzando un SDK AWS
- Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un SDK AWS
- Pubblica un messaggio di testo SMS Amazon SNS utilizzando un SDK AWS
- Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS

Crea un endpoint di piattaforma per le notifiche push di Amazon SNS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come creare un endpoint di piattaforma per notifiche push di Amazon SNS.

CLI

AWS CLI

Creazione di un endpoint dell'applicazione della piattaforma

Nell'esempio `create-platform-endpoint` seguente viene creato un endpoint per l'applicazione della piattaforma indicata utilizzando il token specificato.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Output:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

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.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
```

```
                    platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Crea e pubblica su un argomento FIFO Amazon SNS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come creare un argomento FIFO Amazon SNS e pubblicare un messaggio su di esso.

Java

SDK per Java 2.x

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

In questo esempio

- viene creato un argomento Amazon SNS FIFO, due code FIFO Amazon SQS e una coda Standard.
- viene effettuata la sottoscrizione all'argomento e pubblicato un messaggio nell'argomento.

Il test verifica la ricezione del messaggio in ogni coda. L'esempio completo mostra anche l'aggiunta di policy di accesso e l'eliminazione delle risorse alla fine.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
```

```
                "   fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
```

```
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
```

```
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Java 2.x .
  - CreateTopic
  - Pubblicare
  - Subscribe

Python

SDK per Python (Boto3)

> **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.

Crea un argomento FIFO, sottoscrivi una coda FIFO di Amazon SQS all'argomento e pubblica un messaggio su un argomento Amazon SNS.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
```

```
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
```

```
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
```

```
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
```

```
                }
            )
            logger.info("Added trust policy to the queue.")
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
```

```
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per Python (Boto3).
  - CreateTopic
  - Pubblicare
  - Subscribe

SAP ABAP

SDK per SAP ABAP

> **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.

Crea un argomento FIFO, sottoscrivi una coda Amazon SQS FIFO all'argomento e pubblica un messaggio su un argomento Amazon SNS.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
```

```
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes
      ).
        ov_message_id = lo_result->get_messageid( ).                    "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- Per informazioni dettagliate sulle API, consulta i seguenti argomenti nella Documentazione di riferimento delle API SDK AWS per SAP ABAP.
  - CreateTopic
  - Pubblicare
  - Subscribe

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica messaggi SMS su un argomento Amazon SNS utilizzando un SDK AWS

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.

Java

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);
        }
    }
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un SDK AWS

L'esempio di codice seguente mostra come pubblicare un messaggio di grandi dimensioni su Amazon SNS utilizzando Amazon S3 per archiviare il payload del messaggio.

Java

SDK per Java 1.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 pubblicare un messaggio di grandi dimensioni, utilizza Amazon SNS Extended Client Library per Java. Il messaggio che invii fa riferimento a un oggetto Amazon S3 contenente il contenuto effettivo del messaggio.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;
```

```
                // Message threshold controls the maximum message size that will
be allowed to
                // be published
                // through SNS using the extended client. Payload of messages
exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);
```

```
                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3
 when the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration
 snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

 .withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
 AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it
 exceeds the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration
 = new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica un messaggio di testo SMS Amazon SNS utilizzando un SDK AWS

Negli esempi di codice seguenti viene illustrato come pubblicare messaggi SMS utilizzando Amazon SNS.

.NET

AWS SDK for .NET

> **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.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }
```

```
        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for .NET .

C++

SDK per C++

> **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.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for C++ .

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 .

Kotlin

SDK per Kotlin

> **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.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API di SDK AWS per Kotlin.

PHP

SDK per PHP

> **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.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
```

```
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for PHP.
- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per le API AWS SDK for PHP .

Python

SDK per Python (Boto3)

> **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.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
 must be
                             in E.164 format. For example, a United States phone
```

```
                        number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- Per informazioni dettagliate sulle API, consulta Pubblicazione nella Documentazione di riferimento per l'API SDK for Python (Boto3)AWS .

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano 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.

.NET

AWS SDK for .NET

> **Note**
>
> C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
/// <summary>
/// Console application to run a workflow scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
```

```
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
```

```
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues workflow is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the workflow.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this workflow, you will create an SNS topic and
subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
```

```
                        $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                        $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
                        $"An AWS Identity and Access Management (IAM) policy must
be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
```

```
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
 subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
 SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
 {_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
```

```
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
 filters.");
            for (int i = 0; i < _tones.Length; i++)
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
```

```
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
```

```
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }
```

```
        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
```

```
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
```

```
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Crea una classe che avvolge le operazioni di Amazon SQS.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;
```

```
    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }

    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
```

```
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }

    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
```

```
                                    "\"Effect\": \"Allow\"," +
                                    "\"Principal\": {" +
                                        $"\"Service\": " +
                                            "\"sns.amazonaws.com\"" +
                                           "}," +
                                    "\"Action\": \"sqs:SendMessage\"," +
                                    $"\"Resource\": \"{queueArn}\"," +
                                     "\"Condition\": {" +
                                          "\"ArnEquals\": {" +
                                              $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                           "}" +
                                       "}" +
                                  "}]" +
                               "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
```

```
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
_amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
}
```

Crea una classe che avvolge le operazioni di Amazon SNS.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
```

```
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
```

```
        }

        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
```

```
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }


    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for .NET .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Pubblicare
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

C++

SDK per C++

> **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.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
```

```
   printAsterisksLine();
   std::cout << "In this workflow, you will create an SNS topic and subscribe "
             << NUMBER_OF_QUEUES <<
             " SQS queues to the topic." << std::endl;
   std::cout
           << "You can select from several options for configuring the topic and
the subscriptions for the "
           << NUMBER_OF_QUEUES << " queues." << std::endl;
   std::cout << "You can then post to the topic and see the results in the
queues."
             << std::endl;

   Aws::SNS::SNSClient snsClient(clientConfiguration);

   printAsterisksLine();

   std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
             << std::endl;
   std::cout
           << "FIFO topics deliver messages in order and support deduplication
and message filtering."
           << std::endl;
   bool isFifoTopic = askYesNoQuestion(
           "Would you like to work with FIFO topics? (y/n) ");

   bool contentBasedDeduplication = false;
   Aws::String topicName;
   if (isFifoTopic) {
       printAsterisksLine();
       std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                 << std::endl;
       std::cout
               << "Deduplication IDs are either set in the message or
automatically generated "
               << "from content using a hash function." << std::endl;
       std::cout
               << "If a message is successfully published to an SNS FIFO topic,
any message "
               << "published and determined to have the same deduplication ID, "
               << std::endl;
       std::cout
               << "within the five-minute deduplication interval, is accepted
but not delivered."
```

```
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
```

```
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
```

```
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
```

```
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
```

```
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
```

```
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;
```

```
                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
```

```
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
```

```
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
 outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
```

```
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);
```

```
                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}


bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

```
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for C++ .

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Pubblicare
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Go

SDK per Go V2

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel Repository di esempi di codice AWS.

Esegui uno scenario interattivo al prompt dei comandi.

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
```

```
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic() (string, string, bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }
```

```
 topicArn, err := runner.snsActor.CreateTopic(topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ordinal string, isFifoTopic bool)
 (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)
```

```
err = runner.sqsActor.AttachSendMessagePolicy(queueUrl, queueArn, topicArn)
if err != nil {
 panic(err)
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n"
+
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following
tones: %v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(topicArn, queueArn,
filterPolicy)
```

```
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(topicArn string, isFifoTopic bool,
 contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }
```

```
  err := runner.snsActor.Publish(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }
```

```
  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
```

```
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor

 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic()
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
```

```
  resources.Cleanup()
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definisci una struttura che racchiuda le azioni di Amazon SNS utilizzate in questo esempio.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
```

```
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
```

```
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
```

```
 return err
}
```

Definisci una struttura che racchiude le azioni di Amazon SQS utilizzate in questo esempio.

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
```

```
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(queueUrl string, queueArn string,
 topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(context.TODO(),
 &sqs.SetQueueAttributesInput{
```

```
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
```

```
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(queueUrl string, messages []types.Message)
 error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(context.TODO(),
 &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Pubblicare
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

JavaScript

SDK per (v3) JavaScript

> **Note**
>
> C'è 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 è il punto di ingresso per questo flusso di lavoro.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
```

```
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Il codice precedente fornisce le dipendenze necessarie e avvia il flusso di lavoro. La sezione successiva contiene il blocco principale dell'esempio.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;
```

```
/**
 * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
 * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
 * @param {import('../../libs/prompter.js').Prompter} prompter
 * @param {import('../../libs/logger.js').Logger} logger
 */
constructor(snsClient, sqsClient, prompter, logger) {
  this.snsClient = snsClient;
  this.sqsClient = sqsClient;
  this.prompter = prompter;
  this.logger = logger;
}

async welcome() {
  await this.logger.log(MESSAGES.description);
}

async confirmFifo() {
  await this.logger.log(MESSAGES.snsFifoDescription);
  this.isFifo = await this.prompter.confirm({
    message: MESSAGES.snsFifoPrompt,
  });

  if (this.isFifo) {
    this.logger.logSeparator(MESSAGES.headerDedup);
    await this.logger.log(MESSAGES.deduplicationNotice);
    await this.logger.log(MESSAGES.deduplicationDescription);
    this.autoDedup = await this.prompter.confirm({
      message: MESSAGES.deduplicationPrompt,
    });
  }
}

async createTopic() {
  await this.logger.log(MESSAGES.creatingTopics);
  this.topicName = await this.prompter.input({
    message: MESSAGES.topicNamePrompt,
  });
  if (this.isFifo) {
    this.topicName += ".fifo";
    this.logger.logSeparator(MESSAGES.headerFifoNaming);
    await this.logger.log(MESSAGES.appendFifoNotice);
  }
```

```
    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;

    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );
```

```
    const { Attributes } = await this.sqsClient.send(
      new GetQueueAttributesCommand({
        QueueUrl: response.QueueUrl,
        AttributeNames: ["QueueArn"],
      }),
    );

    this.queues.push({
      queueName,
      queueArn: Attributes.QueueArn,
      queueUrl: response.QueueUrl,
    });

    await this.logger.log(
      MESSAGES.queueCreatedNotice
        .replace("${QUEUE_NAME}", queueName)
        .replace("${QUEUE_URL}", response.QueueUrl)
        .replace("${QUEUE_ARN}", Attributes.QueueArn),
    );
  }
}

async attachQueueIamPolicies() {
  for (const [index, queue] of this.queues.entries()) {
    const policy = JSON.stringify(
      {
        Statement: [
          {
            Effect: "Allow",
            Principal: {
              Service: "sns.amazonaws.com",
            },
            Action: "sqs:SendMessage",
            Resource: queue.queueArn,
            Condition: {
              ArnEquals: {
                "aws:SourceArn": this.topicArn,
              },
            },
          },
        ],
      },
      null,
```

```
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
```

```
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
```

```
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
```

```
              },
            },
          }
        : {}),
    }),
  );

  const publishAnother = await this.prompter.confirm({
    message: MESSAGES.publishAnother,
  });

  if (publishAnother) {
    await this.publishMessages();
  }
}

async receiveAndDeleteMessages() {
  for (const queue of this.queues) {
    const { Messages } = await this.sqsClient.send(
      new ReceiveMessageCommand({
        QueueUrl: queue.queueUrl,
      }),
    );

    if (Messages) {
      await this.logger.log(
        MESSAGES.messagesReceivedNotice.replace(
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      );
      console.log(Messages);

      await this.sqsClient.send(
        new DeleteMessageBatchCommand({
          QueueUrl: queue.queueUrl,
          Entries: Messages.map((message) => ({
            Id: message.MessageId,
            ReceiptHandle: message.ReceiptHandle,
          })),
        }),
      );
    } else {
      await this.logger.log(
```

```
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });

    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
```

```
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella Documentazione di riferimento delle API AWS SDK for JavaScript .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Pubblicare
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi serverless per Amazon SNS che utilizzano SDK AWS

I seguenti esempi di codice mostrano come usare Amazon SNS con AWS SDK.

Esempi

- Richiamo di una funzione Lambda da un trigger Amazon SNS

Richiamo di una funzione Lambda da un trigger Amazon SNS

I seguenti esempi di codice mostrano come implementare una funzione Lambda che riceve un evento attivato dal ricevimento di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

.NET

AWS SDK for .NET

> **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 .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri
```

```
namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }

    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK per Go V2

> **Note**
>
> C'è dell'altro 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 Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDK per Java 2.x

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un evento SNS con Lambda utilizzando Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;


import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
```

```
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

SDK per (v3 JavaScript)

> **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.

Consumo di un evento SNS con JavaScript Lambda che utilizza.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
```

```
    throw err;
  }
}
```

Consumo di un evento SNS con TypeScript Lambda che utilizza.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK per PHP

> **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.

Utilizzo di un evento SNS con Lambda tramite PHP.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK per Python (Boto3)

> **Note**
>
> C'è dell'altro 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 Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK per Ruby

> **Note**
>
> C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un evento SNS con Lambda utilizzando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

Rust

SDK per Rust

> **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.

Utilizzo di un evento SNS con Lambda tramite Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
```

```
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di servizi multipli per Amazon SNS che utilizzano SDK AWS

Le seguenti applicazioni di esempio utilizzano AWS gli SDK per combinare Amazon SNS con altri. Servizi AWS Ogni esempio include un collegamento a GitHub, dove puoi trovare istruzioni su come configurare ed eseguire l'applicazione.

Esempi

- Costruisci un'applicazione per inviare dati a una tabella DynamoDB
- Costruzione di un'applicazione per la pubblicazione e la sottoscrizione che traduce i messaggi
- Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette
- Creazione di un'applicazione Amazon Textract explorer
- Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS
- Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS
- Utilizzo di un'API Gateway per richiamare una funzione Lambda
- Utilizzo degli eventi pianificati per richiamare una funzione Lambda

Costruisci un'applicazione per inviare dati a una tabella DynamoDB

Gli esempi di codice riportati di seguito mostrano come costruire un'applicazione che invia dati a una tabella Amazon DynamoDB e che ti avvisa quando un utente aggiorna la tabella.

Java

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

JavaScript

SDK per JavaScript (v3)

Questo esempio mostra come creare un'app che consenta agli utenti di inviare dati a una tabella Amazon DynamoDB e un messaggio di testo all'amministratore utilizzando Amazon Simple Notification Service (Amazon SNS).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Questo esempio è anche disponibile nella Guida per lo sviluppatore di AWS SDK for JavaScript v3 .

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

Kotlin

SDK per Kotlin

Mostra come creare un'applicazione Android nativa che invia dati utilizzando l'API Kotlin di Amazon DynamoDB e invia un messaggio di testo utilizzando l'API Kotlin di Amazon SNS.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Costruzione di un'applicazione per la pubblicazione e la sottoscrizione che traduce i messaggi

I seguenti esempi di codice mostrano come creare un'applicazione con funzionalità di sottoscrizione e pubblicazione e che traduce i messaggi.

.NET

AWS SDK for .NET

Mostra come utilizzare l'API .NET 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, consulta l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Simple Notification Service per creare un'applicazione Web con funzionalità di sottoscrizione e pubblicazione. Inoltre, questa applicazione di esempio traduce anche i messaggi.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su GitHub.

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire l'esempio che utilizza l'API Java Async, vedi l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Kotlin

SDK per Kotlin

Mostra come utilizzare l'API Kotlin di Amazon SNS per creare un'applicazione 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 creare un'app web, guarda l'esempio completo su. GitHub

Per il codice sorgente completo e le istruzioni su come creare un'app Android nativa, guarda l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette

Nell'esempio di codice seguente viene illustrato come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

.NET

AWS SDK for .NET

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, consulta 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

C++

SDK per C++

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

Java

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

JavaScript

SDK per JavaScript (v3)

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

Kotlin

SDK per Kotlin

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

PHP

SDK per PHP

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

Rust

SDK per Rust

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

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione Amazon Textract explorer

Gli esempi di codice seguenti mostrano come esplorare l'output di Amazon Textract tramite un'applicazione interattiva.

JavaScript

SDK per JavaScript (v3)

Mostra come utilizzare per AWS SDK for JavaScript creare un'applicazione React che utilizza Amazon Textract per estrarre dati dall'immagine di un documento e visualizzarli in una pagina Web interattiva. Questo esempio viene eseguito in un browser Web e richiede, come credenziali, un'identità autenticata Amazon Cognito. Utilizza Amazon Simple Storage Service (Amazon S3) per l'archiviazione e per le notifiche esegue il polling di una coda di Servizio di coda semplice Amazon (Amazon SQS) sottoscritta a un argomento Servizio di notifica semplice Amazon (Amazon SNS).

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 Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK per Python (Boto3)

Mostra come utilizzarlo AWS SDK for Python (Boto3) con Amazon Textract per rilevare elementi di testo, moduli e tabelle nell'immagine di un documento. L'immagine di input e l'output di Amazon Textract sono mostrati in un'applicazione Tkinter che consente di esplorare gli elementi rilevati.

- Invia un'immagine del documento ad Amazon Textract ed esplora l'output degli elementi rilevati.
- Invia immagini direttamente ad Amazon Textract o tramite un bucket Amazon Simple Storage Service (Amazon S3).
- Utilizza le API asincrone per avviare un processo che pubblica una notifica in un argomento Amazon Simple Notification Service (Amazon SNS) al suo termine.

- Esegue il polling di una coda Amazon Simple Queue Service (Amazon SQS) per un messaggio di completamento del processo e visualizza i risultati.

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 S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come rilevare persone e oggetti in un video con Amazon Rekognition.

Python

SDK per Python (Boto3)

Usa Amazon Rekognition per rilevare volti, oggetti e persone nei video avviando processi di rilevamento asincrono. Questo esempio, inoltre, configura Amazon Rekognition per notificare un argomento Amazon Simple Notification Service (Amazon SNS) al completamento dei processi e sottoscrive una coda Amazon Simple Queue Service (Amazon SQS) all'argomento. Quando la coda riceve un messaggio su un processo, questo viene recuperato e vengono restituiti i risultati.

Questo esempio è visualizzato al meglio su. GitHub Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su GitHub.

Servizi utilizzati in questo esempio

- Amazon Rekognition

- Amazon SNS
- Amazon SQS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano 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.

Java

SDK per Java 2.x

Illustrazione della messaggistica con argomenti e code utilizzando Amazon Simple Notification Service (Amazon SNS) e Amazon Simple Queue Service (Amazon SQS).

Per il codice sorgente completo e le istruzioni che illustrano la messaggistica con argomenti e code in Amazon SNS e Amazon SQS, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon SQS

Kotlin

SDK per Kotlin

Illustrazione della messaggistica con argomenti e code utilizzando Amazon Simple Notification Service (Amazon SNS) e Amazon Simple Queue Service (Amazon SQS).

Per il codice sorgente completo e le istruzioni che illustrano la messaggistica con argomenti e code in Amazon SNS e Amazon SQS, consulta l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon SQS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. Utilizzo di Amazon SNS con un SDK AWS Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo di un'API Gateway per richiamare una funzione Lambda

I seguenti esempi di codice mostrano come creare una AWS Lambda funzione richiamata da Amazon API Gateway.

Java

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

JavaScript

SDK per JavaScript (v3)

Mostra come creare una AWS Lambda funzione utilizzando l'API di JavaScript runtime Lambda. 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

Questo esempio è anche disponibile nella Guida per lo sviluppatore di AWS SDK for JavaScript v3 .

Servizi utilizzati in questo esempio

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo degli eventi pianificati per richiamare una funzione Lambda

I seguenti esempi di codice mostrano come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

Java

SDK per Java 2.x

Mostra come creare un evento EventBridge pianificato da Amazon che richiami una AWS Lambda funzione. Configura EventBridge per utilizzare un'espressione cron per pianificare quando viene richiamata la funzione Lambda. In questo esempio, viene creata una funzione

Lambda utilizzando l'API di runtime Lambda Java. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. Questo esempio dimostra come creare un'app che invia un messaggio di testo via mobile ai tuoi dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

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, crei una funzione Lambda utilizzando l'API JavaScript Lambda runtime. 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

Questo esempio è anche disponibile nella Guida per lo sviluppatore di AWS SDK for JavaScript v3 .

Servizi utilizzati in questo esempio

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consultaUtilizzo di Amazon SNS con un SDK AWS. Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Sicurezza di Amazon SNS

Questa sezione fornisce informazioni sulla sicurezza di Amazon SNS, l'autenticazione e il controllo degli accessi, oltre alla sintassi delle policy di accesso Amazon SNS.

Argomenti

Protezione dei dati

Il modello di responsabilità condivisa di AWS si applica alla protezione in Amazon Simple Notification Service. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che esegue tutto l'Cloud AWS. L'utente è responsabile di mantenere il controllo sui contenuti ospitati su questa infrastruttura. Questo contenuto include la configurazione della protezione e le attività di gestione per i servizi AWS utilizzati. Per ulteriori informazioni sulla privacy dei dati, consultare 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 negli AWS.

Per garantire la protezione dei dati, ti suggeriamo di proteggere le credenziali Account AWS e di configurare singoli account utente con AWS Identity and Access Management (IAM). In questo modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere il suo lavoro. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Utilizza SSL/TLS per comunicare con le risorse AWS. È consigliabile TLS 1.2 o versioni successive.
- Configura la registrazione delle API e delle attività degli utenti con AWS CloudTrail.

- Utilizza le soluzioni di crittografia AWS, insieme a tutti i controlli di sicurezza di default all'interno dei servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, ad esempio Amazon Macie, che aiutano a individuare e proteggere i dati personali archiviati in Amazon S3.
- Se si richiedono moduli crittografici convalidati FIPS 140-2 quando si accede ad AWS tramite una CLI o un'API, utilizzare un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.
- Protezione dei dati dei messaggi
  - La protezione dei dati dei messaggi è una nuova importante funzionalità di Amazon SNS
  - Usa la funzionalità di protezione dei dati dei messaggi per analizzare i messaggi alla ricerca di informazioni riservate o sensibili
  - Puoi garantire il controllo dei messaggi per tutti i contenuti che passano attraverso l'argomento
  - Puoi controllare l'accesso ai contenuti dei messaggi pubblicati sull'argomento e dei messaggi recapitati dall'argomento

> **Important**
>
> Ti suggeriamo vivamente di non inserire mai informazioni identificative sensibili, ad esempio i numeri di account dei clienti, in campi a formato libero, ad esempio un campo Name (Nome). Sono inclusi i casi in cui usi Amazon SNS o altri servizi Amazon Web Services tramite la console, l'API, la AWS CLI o gli SDK AWS. I dati inseriti nei tag o nei campi in formato libero utilizzati per i nomi possono essere utilizzati per i registri di fatturazione o 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.

Nelle sezioni seguenti vengono fornite ulteriori informazioni sulla protezione dei contenuti in Amazon SNS.

Argomenti

Crittografia dei dati

La protezione dei dati ha lo scopo di proteggere i dati sia in transito (durante la trasmissione verso e da Amazon SNS), sia quando sono inattivi (ovvero quando sono archiviati su disco nei data center Amazon SNS). È possibile proteggere i dati in transito tramite il protocollo Secure Sockets Layer (SSL) o tramite la crittografia lato client. Per impostazione predefinita, Amazon SNS archivia messaggi e file utilizzando la crittografia del disco. Puoi proteggere i dati inattivi richiedendo ad Amazon SNS di crittografare i tuoi messaggi prima di salvarli nel file system crittografato dei suoi data center. Amazon SNS consiglia di utilizzare SSE per una crittografia ottimizzata dei dati.

Argomenti

Crittografia a riposo

La crittografia lato server (SSE) consente di archiviare dati sensibili in argomenti crittografati proteggendo il contenuto dei messaggi negli argomenti di Amazon SNS utilizzando chiavi gestite in (). AWS Key Management Service AWS KMS

SSE esegue la crittografia dei messaggi non appena vengono ricevuti da Amazon SNS. I messaggi vengono archiviati in forma crittografata e decrittografati solo quando vengono inviati.

- Per informazioni su come gestire SSE utilizzando la AWS Management Console o AWS SDK for Java (impostando l'attributo `KmsMasterKeyId` tramite le operazioni API `CreateTopic` e `SetTopicAttributes`), consulta Abilitazione della crittografia lato server (SSE) per un argomento Amazon SNS.
- Per informazioni su come creare argomenti crittografati utilizzando AWS CloudFormation e impostando la proprietà `KmsMasterKeyId` tramite la risorsa `AWS::SNS::Topic`, consulta la AWS CloudFormation Guida Utente.

> **Important**
>
> Tutte le richieste agli argomenti con la funzione SSE abilitata devono utilizzare HTTPS e Signature Version 4.
> Per informazioni sulla compatibilità di altri servizi con argomenti crittografati, consulta la documentazione del servizio.
> Amazon SNS supporta solo chiavi KMS di crittografia simmetrica. Non è possibile utilizzare nessun altro tipo di chiave KMS per crittografare le risorse del servizio. Per informazioni su come determinare se una chiave KMS è simmetrica o asimmetrica, consultare Identifying asymmetric KMS keys (Individuazione di chiavi KMS asimmetriche).

AWS KMS combina hardware e software sicuri e altamente disponibili per offrire un sistema di gestione delle chiavi a misura di cloud. Quando usi Amazon SNS con AWS KMS, anche le chiavi di dati che eseguono la crittografia dei dati del messaggio vengono crittografate e archiviate con i dati che proteggono.

Di seguito sono elencati i vantaggi derivanti dall'uso di AWS KMS:

- È possibile creare e gestire la AWS KMS key in modo autonomo.
- Puoi anche utilizzare le chiavi KMS gestite da AWS per Amazon SNS, univoche per ogni account e regione.
- Gli standard di sicurezza in AWS KMS consentono di soddisfare i requisiti di conformità correlati alla crittografia.

Per ulteriori informazioni, consulta Cos'è AWS Key Management Service? nella Guida per gli sviluppatori AWS Key Management Service.

Argomenti

- Ambito della crittografia
- Termini chiave

Ambito della crittografia

SSE esegue la crittografia del corpo di un messaggio in un argomento Amazon SNS.

SSE non esegue la crittografia di quanto segue:

- Metadata dell'argomento (nome e attributo dell'argomento)
- Metadati del messaggio (oggetto, ID messaggio, time stamp e attributo)
- Policy di protezione dei dati
- Parametri per argomento

> **Note**
>
> - Un messaggio viene crittografato solo se inviato dopo che la crittografia di una coda.viene abilitata. Amazon SNS non crittografa i messaggi in backlog.
> - Tutti i messaggi crittografati restano crittografati anche se la crittografia del relativo argomento è disabilitata.

Termini chiave

I seguenti termini chiave possono aiutarti a comprendere meglio le funzionalità di SSE . Per una descrizione dettagliata, consulta laDocumentazione di riferimento per l'API Amazon Simple Notification Service.

Chiave di dati

La chiave di crittografia dei dati (DEK) responsabile della crittografia dei contenuti dei messaggi Amazon SNS.

Per ulteriori informazioni, consulta Chiave dati nella AWS Key Management ServiceGuida per sviluppatori e Pacchetto Crittografia nella AWS Encryption SDKGuida per sviluppatori.

ID AWS KMS key

L'alias, l'ARN dell'alias, l'ID della chiave o l'ARN di una chiave AWS KMS key o una chiave AWS KMS personalizzata nel proprio account o in un altro account. Anche se l'alias della chiave AWS KMS gestita da AWS per Amazon SNS è sempre `alias/aws/sns`, l'alias di una chiave AWS KMS personalizzata può essere, ad esempio, `alias/MyAlias`. Puoi utilizzare queste chiavi AWS KMS per proteggere i messaggi negli argomenti Amazon SNS.

> **Note**
>
> Ricorda quanto segue:

- La prima volta che si utilizza la AWS Management Console per specificare la chiave KMS gestita da AWS per Amazon SNS per un argomento, AWS KMS crea la chiave KMS gestita da AWS per Amazon SNS.
- In alternativa, la prima volta che si utilizza l'operazione `Publish` in un argomento con SSE abilitato, AWS KMS crea la chiave KMS gestita da AWS per Amazon SNS.

È possibile creare chiavi AWS KMS, definire le policy che controllano il modo in cui le chiavi AWS KMS possono essere utilizzate e controllare l'utilizzo di AWS KMS tramite la sezione AWS KMS keys della console AWS KMS o l'operazione `CreateKey` AWS KMS. Per ulteriori informazioni, consultare AWS KMS keys e Creating Keys(Creazione di chiavi) nella Guida per gli sviluppatori di AWS Key Management Service. Per altri esempi di AWS KMS identificatori, consulta l'AWS Key Management ServiceAPI KeyIdReference. Per ulteriori informazioni su come individuare gli identificatori AWS KMS, consulta Trovare l'ID e l'ARN delle chiavi nella Guida per gli sviluppatori di AWS Key Management Service.

Important

Non vi sono costi aggiuntivi per l'utilizzo di AWS KMS. Per ulteriori informazioni, consulta Stima dei costi AWS KMS e Prezzi di AWS Key Management Service.

Gestione delle chiavi

Nelle sezioni seguenti vengono fornite informazioni sull'utilizzo delle chiavi gestite in AWS Key Management Service (AWS KMS). Per saperne di più su

Note

Amazon SNS supporta solo chiavi KMS di crittografia simmetrica. Non è possibile utilizzare nessun altro tipo di chiave KMS per crittografare le risorse del servizio. Per informazioni su come determinare se una chiave KMS è simmetrica o asimmetrica, consultare Identifying asymmetric KMS keys (Individuazione di chiavi KMS asimmetriche).

Argomenti

- Stima dei costi AWS KMS

- Configurazione delle autorizzazioni per AWS KMS
- Errori AWS KMS

Stima dei costi AWS KMS

Per prevedere i costi e capire meglio la fatturazione AWS, è utile sapere quanto spesso Amazon SNS usa la AWS KMS key.

> **Note**
>
> Anche se la seguente formula può darti un'idea molto precisa dei costi previsti, i costi effettivi potrebbero essere più elevati a causa della natura diffusa di Amazon SNS.

Per calcolare il numero di richieste API (R) per argomento, usa la formula seguente:

```
R = B / D * (2 * P)
```

B è il periodo di fatturazione (in secondi).

D è il periodo di riutilizzo della chiave dati (in secondi–Amazon SNS riutilizza una chiave dati per un massimo di 5 minuti).

P è il numero di principali per la pubblicazione che effettuano invii all'argomento Amazon SNS.

Di seguito vengono riportati esempi di calcolo. Per informazioni dettagliate sui prezzi, consulta Prezzi di AWS Key Management Service.

Esempio 1: calcolo del numero di chiamate API AWS KMS per 1 editore e 1 argomento

Questo esempio assume quanto segue:

- Il periodo di fatturazione è compreso tra il 1° e il 31 gennaio (2.678.400 secondi).
- Il periodo di riutilizzo della chiave di dati è di 5 minuti (300 secondi).
- C'è 1 argomento.
- C'è un principale per la pubblicazione.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Esempio 2: calcolo del numero di chiamate API AWS KMS per più editori e 2 argomenti

Questo esempio assume quanto segue:

- Il periodo di fatturazione è compreso tra il 1° e il 28 febbraio (2.419.200 secondi).
- Il periodo di riutilizzo della chiave di dati è di 5 minuti (300 secondi).
- Ci sono 2 argomenti.
- Il primo argomento ha 3 principali per la pubblicazione.
- Il secondo argomento ha 5 principali per la pubblicazione.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Configurazione delle autorizzazioni per AWS KMS

Prima di utilizzare SSE, è necessario configurare le policy AWS KMS key per consentire la crittografia degli argomenti e la crittografia e la decrittografia dei messaggi. Per esempi e ulteriori informazioni sulle autorizzazioni diAWS KMS, consulta AWS KMSAutorizzazioni API: Documentazione su operazioni e risorse nellaAWS Key Management ServiceGuida per sviluppatori. Per dettagli su come configurare un argomento Amazon SNS con crittografia lato server, consulta Configura un argomento Amazon SNS con crittografia lato server.

> **Note**
>
> È possibile gestire le autorizzazioni per le chiavi KMS utilizzando le policy IAM. Per ulteriori informazioni, consulta Uso di policy IAM con AWS KMS.
> Sebbene sia possibile configurare le autorizzazioni globali per inviare e ricevere da Amazon SNS, AWS KMS richiede esplicitamente la denominazione dell'ARN completo delle KMS in regioni specifiche nella sezione `Resource` di una policy IAM.

È necessario assicurarsi che le policy della chiave AWS KMS key consentano le autorizzazioni necessarie. Per eseguire questa operazione, denomina i principali che producono e utilizzano messaggi crittografati in Amazon SNS come utenti nella policy della chiave KMS.

In alternativa, è possibile specificare le operazioni AWS KMS richieste e l'ARN KMS in una policy IAM assegnata ai principali che pubblicano ed effettuano la sottoscrizione per ricevere messaggi

crittografati in Amazon SNS. Per ulteriori informazioni, consulta Gestione dell'accesso a AWS KMS nella Guida per sviluppatori di AWS Key Management Service.

Se stai selezionando una chiave gestita dal cliente per l'argomento Amazon SNS e stai utilizzando alias per controllare l'accesso alle chiavi KMS utilizzando le policy IAM o le policy delle chiavi KMS con la chiave di condizione `kms:ResourceAliases`, assicurati che alla chiave gestita dal cliente selezionata sia associato anche un alias. Per ulteriori informazioni sull'uso degli alias per controllare l'accesso alle chiavi KMS, consulta Using aliases to control access to KMS keys (Utilizzo degli alias per controllare l'accesso alle chiavi KMS) nella Guida per gli sviluppatori di AWS Key Management Service.

Consentire a un utente di inviare messaggi a un argomento con SEE

L'editore deve avere le autorizzazioni `kms:GenerateDataKey*` e `kms:Decrypt` per la chiave AWS KMS key.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Abilitare la compatibilità tra le origini eventi dai servizi AWS e gli argomenti crittografati

Diversi servizi AWS pubblicano eventi in argomenti Amazon SNS. Per consentire a queste origini eventi di utilizzare argomenti crittografati, devi eseguire la seguente procedura.

1. Utilizzo di una chiave gestita dal cliente. Per ulteriori informazioni, consulta Creazione di chiavi nella Guida per gli sviluppatori di AWS Key Management Service.

2. Per consentire al servizio AWS di disporre delle autorizzazioni `kms:GenerateDataKey*` e `kms:Decrypt`, aggiungi la seguente dichiarazione alla policy KMS.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Origine eventi	Principale del servizio
Amazon CloudWatch	`cloudwatch.amazonaws.com`
Amazon CloudWatch Events	`events.amazonaws.com`
AWS CodeCommit	`codecommit.amazonaws.com`
AWS CodeStar	`codestar-notifications.amazonaws.com`
AWS Database Migration Service	`dms.amazonaws.com`
AWS Directory Service	`ds.amazonaws.com`
Amazon DynamoDB	`dynamodb.amazonaws.com`
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`

Origine eventi	Principale del servizio
Amazon Simple Email Service	`ses.amazonaws.com`
Amazon Simple Storage Service	`s3.amazonaws.com`
AWS Snowball	`importexport.amazonaws.com`
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager è costituito da due principi di servizio: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Alcune origini di eventi Amazon SNS richiedono un ruolo IAM, invece di un principale del servizio, nella policy delle AWS KMS key:
>
> - Dimensionamento automatico Amazon EC2
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. Aggiungi le chiavi di condizione `aws:SourceAccount` e `aws:SourceArn` per la policy delle risorse KMS per proteggere ulteriormente la chiave KMS da attacchi confused deputy. Fare riferimento all'elenco della documentazione specifica del servizio (sopra) per i dettagli esatti in ciascun caso.

> **Important**
>
> L'aggiunta di un `aws:SourceAccount` e un `aws:SourceArn` a una policy AWS KMS non è supportata per gli argomenti EventBridge-to-encrypted.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "service.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "customer-account-id"
    },
    "ArnLike": {
      "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type:customer-resource-id"
    }
  }
}
```

4. Abilitare SSE per il proprio argomento utilizzando la chiave KMS.
5. Fornire l'ARN dell'argomento crittografato per l'origine eventi.

Errori AWS KMS

Quando utilizzi Amazon SNS e AWS KMS, possono verificarsi degli errori. Di seguito sono descritti gli errori e le possibili opzioni per la risoluzione dei problemi.

KMSAccessDeniedException

Il testo cifrato fa riferimento a una chiave che non esiste o a cui non hai accesso.

Codice di stato HTTP: 400

KMSDisabledException

La richiesta è stata rifiutata perché la chiave KMS specificata non è abilitata.

Codice di stato HTTP: 400

KMSInvalidStateException

La richiesta è stata rifiutata perché lo stato della risorsa specificata non è valido per questa richiesta. Per ulteriori informazioni, consulta Key states of AWS KMS keys (Stati chiave nelle chiavi AWS KMS keys) nella Guida per gli sviluppatori di AWS Key Management Service.

Codice di stato HTTP: 400

KMSNotFoundException

La richiesta è stata rifiutata perché non è possibile trovare l'entità o la risorsa specificata.

Codice di stato HTTP: 400

KMSOptInRequired

L'ID chiave di accesso AWS necessita di una sottoscrizione al servizio.

Codice di stato HTTP: 403

KMSThrottlingException

La richiesta è stata negata a causa del throttling della richiesta. Per ulteriori informazioni sulla limitazione della larghezza di banda della rete consulta Quote nella Guida per gli sviluppatori di AWS Key Management Service.

Codice di stato HTTP: 400

Abilitazione della crittografia lato server (SSE) per un argomento Amazon SNS

La crittografia lato server (SSE) consente di archiviare dati sensibili in argomenti crittografati. SSE protegge il contenuto dei messaggi negli argomenti Amazon SNS utilizzando chiavi gestite in AWS Key Management Service (AWS KMS). Per ulteriori informazioni sulla crittografia lato server con Amazon SNS, consulta Crittografia a riposo. Per ulteriori informazioni sulla creazione di chiavi AWS KMS, consultare Creazione di chiavi nella Guida per gli sviluppatori di AWS Key Management Service.

> **Important**
>
> Tutte le richieste agli argomenti con la funzione SSE abilitata devono utilizzare HTTPS e Signature Version 4.

Abilitare la crittografia lato server (SSE) per un argomento Amazon SNS tramite la AWS Management Console

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti), selezionare un argomento e scegliere Actions (Operazioni), Edit (Modifica).
4. Espandere la sezione Encryption (Crittografia) e procedere come segue:

   a. Scegliere Enable encryption (Abilita crittografia).

   b. Specificare la chiave AWS KMS. Per ulteriori informazioni, consulta Termini chiave.

      Per ciascun tipo di KMS vengono visualizzati i valori Description (Descrizione), Account e KMS ARN (ARN KMS).

      > **Important**
      >
      > Se non sei il proprietario della KMS, oppure se effettui l'accesso con un account che non dispone delle autorizzazioni `kms:ListAliases` e `kms:DescribeKey`, non puoi visualizzare le informazioni sulla KMS sulla console Amazon SNS.
      > Chiedi al proprietario della KMS di concederti queste autorizzazioni. Per esempi e ulteriori informazioni consulta AWS KMS Autorizzazioni API: e Documentazione su operazioni e risorse nella AWS Key Management Service Guida per sviluppatori.

      - La KMS gestita da AWS per Amazon SNS (Default) alias/aws/sns è selezionata per impostazione predefinita.

        > **Note**
        >
        > Ricorda quanto segue:
        >
        > - La prima volta che si utilizza la AWS Management Console per specificare la chiave KMS gestita da AWS per Amazon SNS per un argomento, AWS KMS crea la chiave KMS gestita da AWS per Amazon SNS.

- In alternativa, la prima volta che si utilizza l'operazione `Publish` in un argomento con SSE abilitato, AWS KMS crea la chiave KMS gestita da AWS per Amazon SNS.

- Per utilizzare una chiave KMS personalizzata dal proprio account AWS, selezionare il campo Chiave KMS, quindi scegliere la chiave KMS personalizzata dall'elenco.

> **Note**
>
> Per istruzioni sulla creazione di KMS personalizzate, consulta Creazione di chiavi nella Guida per gli sviluppatori di AWS Key Management Service

- Per utilizzare un ARN KMS personalizzato dal proprio account AWS o da un altro account AWS, inserirlo nel campo Chiave KMS.

5. Seleziona Salva modifiche.

   La crittografia SSE è abilitata per l'argomento e viene visualizzata la pagina ***MioArgomento***.

   Lo status di Crittografia, Account AWS, Chiave principale del cliente (CMK), ARN CMK e Descrizione dell'argomento sono visualizzati nella tabella Crittografia.

Configura un argomento Amazon SNS con crittografia lato server

Quando crei la tua chiave KMS, utilizza la seguente politica per le chiavi KMS:

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "service.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
```

```
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

Abilitazione della crittografia lato server (SSE) per un argomento Amazon SNS con coda Amazon SQS crittografata sottoscritta

Puoi abilitare la crittografia SSE (crittografia lato server) per un argomento per proteggerne i dati. Per consentire ad Amazon SNS di inviare messaggi a code crittografate Amazon SQS, la chiave gestita dal cliente associata alla coda Amazon SQS deve avere una dichiarazione di policy che conceda ad Amazon SNS l'accesso del principale del servizio alle operazioni `GenerateDataKey` e `Decrypt` dell'API AWS KMS. Per ulteriori informazioni sull'utilizzo di SSE, consulta Crittografia a riposo.

Questa pagina mostra come abilitare SSE per un argomento Amazon SNS a cui è iscritta una coda Amazon SQS crittografata utilizzando la AWS Management Console.

Fase 1: creazione di una chiave KMS personalizzata

1. Accedere alla console AWS KMS con un utente che dispone di almeno di una policy `AWSKeyManagementServicePowerUser`.
2. Scegli Create a key (Crea una chiave).
3. Per creare una chiave KMS di crittografia simmetrica, in Key type (Tipo di chiave) scegli Symmetric (Simmetrica).

   Per informazioni su come creare una chiave KMS asimmetrica nella console AWS KMS, consultare Creating asymmetric KMS keys (console) (Creazione di chiavi KMS asimmetriche (console).
4. In Utilizzo della chiave, l'opzione Crittografa e decrittografa è selezionata per impostazione predefinita.

   Per informazioni su come creare le chiavi KMS che generano e verificano i codici MAC, consultare Creating HMAC KMS keys (Creazione di chiavi KMS HMAC).

   Per informazioni sulle Opzioni avanzate, consultare Special-purpose keys (Chiavi per uso speciale).
5. Seleziona Successivo.

6. Digita un alias per la chiave KMS. Un nome di alias non può iniziare con **`aws/`**. Il prefisso **`aws/`** è riservato da Amazon Web Services per rappresentare le Chiavi gestite da AWS nel tuo account.

   > **Note**
   >
   > L'aggiunta, l'eliminazione o l'aggiornamento di un alias può consentire o negare l'autorizzazione alla chiave KMS. Per i dettagli, consultare ABAC for AWS KMS (ABAC per KMS) e Using aliases to control access to KMS keys (Utilizzo degli alias per controllare l'accesso alle chiavi KMS).

   Un alias è un nome visualizzato che può essere utilizzato per identificare la chiave KMS. È consigliabile scegliere un alias che indica il tipo di dati che desideri proteggere o l'applicazione che desideri utilizzare con la chiave KMS.

   Gli alias sono obbligatori quando si crea una chiave KMS nella AWS Management Console. Sono facoltativi quando si utilizza l'operazioneCreateKey.

7. (Facoltativo) Digita una descrizione per la chiave KMS.

   Puoi aggiungere una descrizione ora o aggiornarla in qualsiasi momento, a meno che lo stato della chiave non sia `Pending Deletion` o `Pending Replica Deletion`. Per aggiungere, modificare o eliminare la descrizione di una chiave gestita dal cliente esistente, modifica la descrizione nella AWS Management Console o utilizza l'operazione UpdateKeyDescription.

8. (Facoltativo) Digitare una chiave di tag e un valore di tag facoltativo. Per aggiungere più di un tag alla chiave KMS scegli Add tag (Aggiungi tag).

   > **Note**
   >
   > L'applicazione o l'eliminazione di un tag chiave KMS può consentire o negare l'autorizzazione alla chiave KMS. Per i dettagli, consultare ABAC for AWS KMS (ABAC per KMS) e Using tags to control access to KMS keys (Utilizzo dei tag per controllare l'accesso alle chiavi KMS).

   Quando aggiungi i tag alle risorse AWS, AWS genera un report di allocazione dei costi in cui l'utilizzo e i costi sono aggregati in base ai tag. I tag possono essere utilizzati anche per controllare l'accesso a una chiave KMS. Per informazioni sull'assegnazione di tag alle chiavi

KMS, consultare Tagging keys (Assegnazione di tag alle chiavi) e ABAC for AWS KMS (ABAC per KMS).

9. Seleziona Successivo.
10. Seleziona i ruoli e gli utenti IAM che possono gestire la chiave KMS.

> **Note**
>
> Questa policy delle chiavi fornisce all'Account AWS il controllo completo di questa chiave KMS. Consente agli amministratori dell'account di utilizzare policy IAM per concedere ad altri principali l'autorizzazione per la gestione della chiave KMS. Per informazioni dettagliate, consultare Default key policy (Policy della chiave predefinita).
>
> Le best practice di IAM disincentivano l'uso di utenti IAM con credenziali a lungo termine. Quando possibile, utilizza i ruoli IAM che forniscono credenziali temporanee. Per i dettagli, consultare Best practice per la sicurezza in IAM nella Guida per l'utente IAM.

11. (Facoltativo) Per impedire ai ruoli e agli utenti IAM selezionati di eliminare questa chiave KMS, nella sezione Eliminazione chiave nella parte inferiore della pagina, deseleziona la casella di controllo Consenti agli amministratori delle chiavi di eliminare questa chiave.
12. Seleziona Successivo.
13. Selezionare i ruoli e gli utenti IAM che possono utilizzare la chiave nelle operazioni di crittografia. Seleziona Successivo.
14. Nella pagina Review and edit key policy (Verifica e modifica la policy delle chiavi), aggiungi la seguente istruzione alla policy delle chiavi, quindi scegli Finish (Termina).

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

La nuova chiave personalizzata gestita dal cliente viene visualizzata nell'elenco delle chiavi.

Fase 2: creazione di un argomento Amazon SNS crittografato

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Scegliere Create topic (Crea argomento).
4. Nella pagina Create new topic (Crea nuovo argomento), in Name (Nome), immettere il nome di un argomento (ad esempio `MyEncryptedTopic`), quindi scegliere Create topic (Crea argomento).
5. Espandere la sezione Encryption (Crittografia) e procedere come segue:

   a. Scegli Enable server-side encryption (Abilita crittografia lato server).

   b. Specificare la chiave gestita dal cliente. Per ulteriori informazioni, consulta Termini chiave.

      Per ogni tipo di chiave gestita dal cliente, vengono visualizzati la Descrizione, l'account e l'ARN della chiave gestita dal cliente.

      > **Important**
      >
      > Se non si è il proprietario della chiave gestita dal cliente, oppure se si effettua l'accesso con un account che non dispone delle autorizzazioni `kms:ListAliases` e `kms:DescribeKey`, non è possibile visualizzare le informazioni sulla chiave gestita dal cliente sulla console Amazon SNS.
      > Richiedere al proprietario della chiave gestita dal cliente di concedere queste autorizzazioni. Per esempi e ulteriori informazioni consulta AWS KMS Autorizzazioni API: e Documentazione su operazioni e risorse nella AWS Key Management Service Guida per sviluppatori.

   c. In Chiave gestita dal cliente, scegliere la chiave MyCustomKey creata in precedenza e quindi scegliere Abilita la crittografia lato server.
6. Seleziona Salva modifiche.

   La crittografia SSE è abilitata per l'argomento e viene visualizzata la pagina MioArgomento.

   Lo stato Crittografia, Account AWS, Chiave gestita dal cliente, ARN della chiave gestita dal cliente e Descrizione dell'argomento sono visualizzati nella scheda Crittografia.

Il nuovo argomento crittografato viene visualizzato nell'elenco degli argomenti.

Fase 3: creazione e abbonamento a code Amazon SQS crittografate

1. Accedere alla console Amazon SQS.
2. Scegli Create New Queue (Crea nuova coda).
3. Nella pagina Create New Queue (Crea nuova coda), procedi come indicato di seguito:
   a. Immetti Queue Name (Nome coda) (ad esempio, `MyEncryptedQueue1`).
   b. Scegli Standard Queue (Coda standard), quindi Configure Queue (Configura coda).
   c. Scegli Use SSE (Usa SSE).
   d. Per AWS KMS key, scegliere MyCustomKey creata in precedenza, quindi selezionare Crea coda.
4. Ripeti la procedura per creare una seconda coda (ad esempio, denominata `MyEncryptedQueue2`).

   Le nuove code crittografate vengono visualizzate nell'elenco delle code.
5. Nella console Amazon SQS, seleziona `MyEncryptedQueue1` e `MyEncryptedQueue2`, quindi scegli Operazioni coda, Sottoscrivi code ad argomento SNS.
6. Nella finestra di dialogo Subscribe to a Topic (Sottoscrivi a un argomento), per Choose a Topic (Scegli un argomento) seleziona MyEncryptedTopic, quindi scegli Subscribe (Sottoscrivi).

   Le sottoscrizioni delle code crittografate all'argomento crittografato vengono visualizzate nella finestra di dialogo Topic Subscription Result (Risultato sottoscrizione argomento).
7. Scegli OK.

Fase 4: pubblicazione di un messaggio nell'argomento crittografato

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nell'elenco degli argomenti selezionare MyEncryptedTopic, quindi scegliere Publish message (Pubblica messaggio).
4. Nella pagina Publish a message (Pubblica un messaggio), procedi come indicato di seguito:
   a. (Facoltativo) Nella sezione Message details (Dettagli messaggio), immettere il valore per Subject (Oggetto) (ad esempio, `Testing message publishing`).

b. Nella sezione Message body (Corpo messaggio), immettere il corpo del messaggio (ad esempio `My message body is encrypted at rest.`).

c. Seleziona Publish message (Pubblica messaggio).

Il messaggio viene pubblicato nelle code crittografate sottoscritte.

Fase 5: verifica della consegna del messaggio

1. Accedere alla console Amazon SQS.
2. Dall'elenco delle code, scegli MyEncryptedQueue1, quindi seleziona Send and receive messages (Invia e ricevi messaggi).
3. Nella pagina View/Delete Messages in MyEncryptedQueue1 (Visualizza/Elimina messaggi in MyEncryptedQueue1), scegli Start polling for messages (Avvia polling per i messaggi).

   Viene visualizzato il messaggio inviato in precedenza.
4. Scegli More Details (Altri dettagli) per visualizzare il messaggio.
5. Al termine, scegli Close (Chiudi).
6. Ripeti la procedura per MyEncryptedQueue2.

Riservatezza del traffico Internet

Un endpoint Virtual Private Cloud (VPC) per Amazon SNS è un'entità logica all'interno di un VPC che consente la connettività solo ad Amazon SNS. Il VPC instrada le richieste ad Amazon SNS e le risposte al VPC. Nelle sezioni seguenti vengono fornite informazioni sull'utilizzo degli endpoint VPC e sulla creazione delle policy di endpoint VPC.

Se usi Amazon Virtual Private Cloud (Amazon VPC) per l'hosting delle risorse AWS , puoi stabilire una connessione privata tra il VPC e Amazon SNS. Con questa connessione, puoi pubblicare messaggi nei tuoi argomenti Amazon SNS senza inviarli tramite l'Internet pubblico.

Amazon VPC è un servizio AWS che puoi utilizzare per avviare risorse AWS in una rete virtuale da te definita. Con un VPC;, detieni il controllo delle impostazioni della rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. Per collegare il tuo VPC a Amazon SNS, definisci un endpoint VPC dell'interfaccia. Questo tipo di endpoint consente di collegare il VPC ai servizi AWS. L'endpoint offre una connettività scalabile e affidabile a Amazon SNS senza richiedere un Internet gateway, un'istanza NAT (Network Address Translation) o una connessione VPN. Per ulteriori informazioni, consulta Endpoint VPC di interfaccia nella Guida per l'utente di Amazon VPC.

Le informazioni contenute in questa sezione sono per gli utenti di Amazon VPC. Per ulteriori informazioni e per iniziare a creare un VPC, consulta Nozioni di base su Amazon VPC nella Guida per l'utente di Amazon VPC.

> **Note**
>
> Gli endpoint VPC non consentono di sottoscrivere un argomento Amazon SNS a un indirizzo IP privato.

Argomenti

- Creazione di un endpoint Amazon VPC per Amazon SNS
- Creazione di una policy per endpoint VPC di Amazon per Amazon SNS
- Pubblicazione di un messaggio Amazon SNS da Amazon VPC

Creazione di un endpoint Amazon VPC per Amazon SNS

Per pubblicare messaggi negli argomenti Amazon SNS da Amazon VPC, crea un endpoint VPC di interfaccia. Quindi, puoi pubblicare messaggi agli argomenti mantenendo il traffico all'interno della rete gestita con VPC.

Utilizza le informazioni riportate di seguito per creare l'endpoint e testare la connessione tra VPC e Amazon SNS. In alternativa, per una procedura guidata che consente di iniziare da zero, consulta Pubblicazione di un messaggio Amazon SNS da Amazon VPC.

Creazione dell'endpoint

Puoi creare un endpoint Amazon SNS nel tuo VPC utilizzando un AWS SDK AWS Management Console AWS CLI, l'API Amazon SNS oppure. AWS CloudFormation

Per informazioni sulla creazione e sulla configurazione di un endpoint utilizzando la console Amazon VPC o il AWS CLI, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

> **Important**
>
> Puoi utilizzare Amazon Virtual Private Cloud solo con endpoint Amazon SNS HTTPS.

Quando si crea un endpoint, occorre specificare Amazon SNS come servizio a cui desideri che il tuo VPC si connetta. Nella console Amazon VPC, i nomi dei servizi variano in base alla regione. Ad esempio, se si sceglie Stati Uniti orientali (Virginia settentrionale), il nome del servizio è com.amazonaws.us-east-1.sns.
Quando configuri Amazon SNS per inviare messaggi da Amazon VPC, devi abilitare il DNS privato e specificare gli endpoint nel formato `sns.us-east-2.amazonaws.com`.
Il DNS privato non supporta endpoint precedenti come `queue.amazonaws.com` o `us-east-2.queue.amazonaws.com`.

Per informazioni sulla creazione e configurazione di un endpoint utilizzando AWS CloudFormation, consulta la risorsa nella Guida per l'utente. *AWS::EC2::VPCEndpoint*AWS CloudFormation

Verifica la connessione tra il tuo VPC e Amazon SNS

Dopo aver creato un endpoint per Amazon SNS, puoi pubblicare i messaggi dal VPC agli argomenti Amazon SNS. Per testare questa connessione, procedi come indicato di seguito:

1. Connettiti a un'istanza Amazon EC2 che risiede nel tuo VPC. Per informazioni sulla connessione, consulta Connessione all'istanza Linux o Connessione all'istanza Windows nella documentazione Amazon EC2.

   Ad esempio, per connettersi a un'istanza Linux utilizzando un client SSH, esegui il comando seguente da un terminale:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Dove:

   - ec2-key-pair.pem è il file che contiene la coppia di chiavi che Amazon EC2; ha fornito al momento della creazione dell'istanza.
   - instance-hostname è il nome host pubblico dell'istanza. Per ottenere il nome host nella console Amazon EC2: scegli Istanze, quindi seleziona l'istanza e trova il valore del DNS pubblico (IPv4).
2. Dalla tua istanza, utilizza il comando Amazon SNS `publish` con la AWS CLI. È possibile inviare un messaggio semplice a un argomento con il comando seguente:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

Dove:

- aws-region è AWS la regione in cui si trova l'argomento.
- sns-topic-arnè l'Amazon Resource Name (ARN) dell'argomento. Per ottenere l'ARN dalla console Amazon SNS: scegli Argomenti, quindi trova l'argomento e il valore nella colonna ARN.

Se il messaggio è stato ricevuto da Amazon SNS, il terminale stamperà un ID messaggio, come il seguente:

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Creazione di una policy per endpoint VPC di Amazon per Amazon SNS

È possibile creare una policy per gli endpoint VPC di Amazon per Amazon SNS per specificare quanto segue:

- Il principale che può eseguire operazioni.
- Le operazioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consultare Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

Il seguente esempio di policy di endpoint VPC specifica che l'utente IAM `MyUser` è autorizzato a pubblicare nell'argomento Amazon SNS `MyTopic` .

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
```

```
  }]
}
```

Non si può accedere a quanto segue:

- Altre operazioni API Amazon SNS, come `sns:Subscribe` e `sns:Unsubscribe`.
- Altri utenti IAM e regole che provano a utilizzare questo endpoint VPC.
- `MyUser` che pubblica in un altro argomento Amazon SNS.

> **Note**
>
> L'utente IAM può ancora utilizzare altre operazioni API Amazon SNS dall'esterno del VPC.

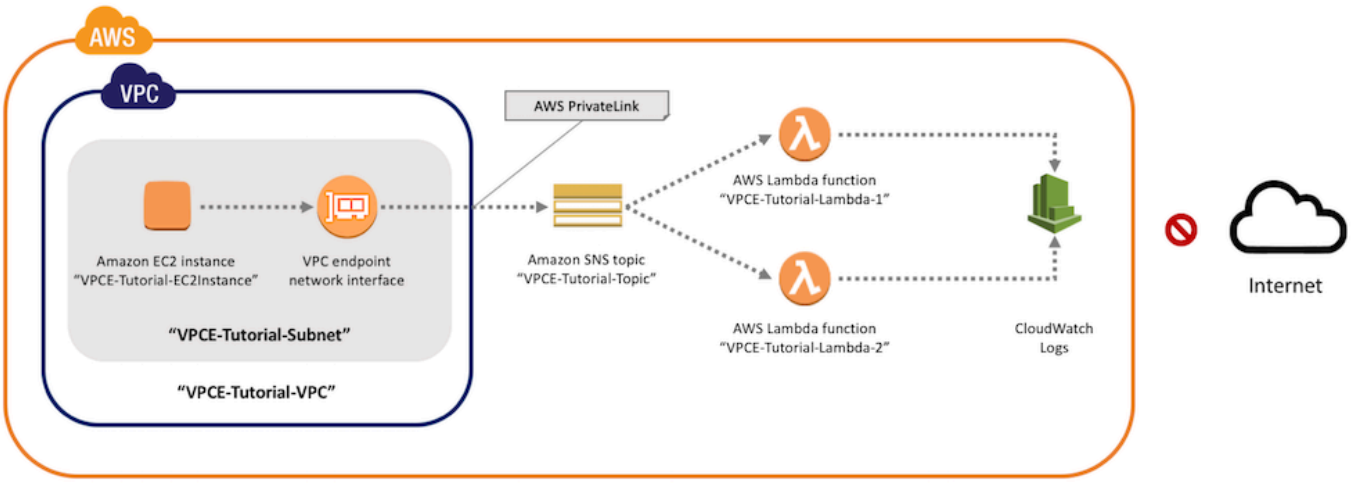
Pubblicazione di un messaggio Amazon SNS da Amazon VPC

In questa sezione viene descritto come pubblicare in un argomento Amazon SNS mantenendo i messaggi protetti in una rete privata. Puoi pubblicare un messaggio da un'istanza Amazon EC2 ospitata in Amazon Virtual Private Cloud (Amazon VPC). Il messaggio rimane sulla rete AWS senza viaggiare sulla rete Internet pubblica. Per pubblicare messaggi privatamente da un VPC, è possibile migliorare la sicurezza del traffico tra le applicazioni e Amazon SNS. Questa sicurezza è importante quando si pubblicano informazioni personali identificabili (PII) sui tuoi clienti oppure quando l'applicazione è soggetta a normative di mercato. Ad esempio, la pubblicazione in privato è utile se si tratta di un sistema sanitario che deve rispettare l'Health Insurance Portability and Accountability Act (HIPAA) o di un sistema finanziario che deve essere conforme al Payment Card Industry Data Security Standard (PCI DSS).

La procedura generale da seguire è riportata di seguito:

- Utilizzare un modello AWS CloudFormation per creare automaticamente una rete privata temporanea nell'account Account AWS.
- Creare un endpoint VPC che collega VPC con Amazon SNS.
- Accedere a un'istanza Amazon EC2 e pubblicare un messaggio in privato a un argomento Amazon SNS.
- Verificare che il messaggio sia stato consegnato correttamente.
- Eliminare le risorse create per durante questa procedura in modo che non rimangano nell' Account AWS.

Il diagramma seguente illustra la rete privata creata nell'account AWS una volta completato questa procedura:



Questa rete è costituita da un VPC che contiene un'istanza Amazon EC2. L'istanza si collega a Amazon SNS; attraverso un endpoint VPC di interfaccia. Questo tipo di endpoint si connette ai servizi supportati da AWS PrivateLink. Con questa connessione stabilita, puoi accedere all'istanza Amazon EC2 e pubblicare messaggi all'argomento Amazon SNS, anche se la rete è disconnessa dalla rete Internet pubblica. L'argomento distribuisce i messaggi che riceve a due funzioni AWS Lambda di sottoscrizione. Queste funzioni registrano i messaggi ricevuti in Amazon CloudWatch Logs.

Per completare questi passaggi occorrono circa 20 minuti.

Argomenti

Prima di iniziare

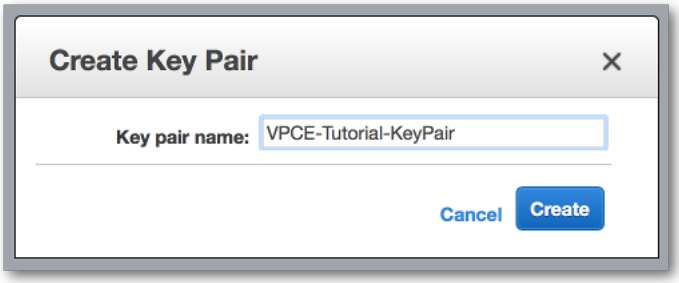
Prima di iniziare è necessario un account Amazon Web Services (AWS). Quando accedi, il tuo account viene automaticamente registrato per tutti i servizi in AWS, compreso Amazon SNS e Amazon VPC. Se non hai già creato un account, vai a https://aws.amazon.com/, quindi scegli Crea un account gratuito.

Fase 1: Creazione di una coppia di chiavi di Amazon EC2

Una coppia di chiavi viene utilizzata per effettuare l'accesso a un'istanza Amazon EC2. È costituita da una chiave pubblica utilizzata per crittografare le informazioni di accesso e una chiave privata utilizzata per decifrarle. Quando crei una coppia di chiavi, viene scaricata una copia della chiave privata. Successivamente, una coppia di chiavi viene utilizzata per effettuare l'accesso a un'istanza Amazon EC2. Per effettuare l'accesso, è necessario specificare il nome della coppia di chiavi e fornire la chiave privata.

Per creare la coppia di chiavi

1. Accedere a AWS Management Console e aprire la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel menu di navigazione a sinistra, individua la sezione Network & Security (Rete e sicurezza). Quindi scegli Key Pairs (Coppie di chiavi).
3. Scegli Crea coppia di chiavi.
4. Nella finestra Create Key Pair (Crea coppia di chiavi), per Key pair name (Nome coppia di chiavi), digita **`VPCE-Tutorial-KeyPair`**. Quindi scegli Create (Crea).



5. Il file della chiave privata viene automaticamente scaricato dal browser. Salvalo in un posto sicuro. Amazon EC2 assegna al file un'estensione `.pem`.
6. (Facoltativo) Se usi un client SSH su un computer Mac o Linux per connetterti alla tua istanza, usa il comando `chmod` per impostare le autorizzazioni del file della chiave privata in modo da essere l'unico a poterlo leggere:

a. Apri un terminale e vai alla directory che contiene la chiave privata:

```
$ cd /filepath_to_private_key/
```

b. Imposta le autorizzazioni utilizzando il comando seguente:

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

Fase 2: creazione delle risorse AWS

Per configurare l'infrastruttura è possibile utilizzare un modello AWS CloudFormation. Un modello è un file che funge come schema per la creazione di risorse AWS, come istanze Amazon EC2 e argomenti Amazon SNS. È possibile scaricare il modello per questa procedura su GitHub.

Puoi fornire il modello a AWS CloudFormation e AWS CloudFormation effettua il provisioning delle risorse di cui hai bisogno come uno stack nel proprio Account AWS. Uno stack è una raccolta di risorse che puoi gestire come un'unità singola. Una volta terminato la procedura, è possibile utilizzare AWS CloudFormation per eliminare contemporaneamente tutte le risorse nello stack. Queste risorse non rimangono nell' Account AWS a meno che non lo si desideri.

Lo stack per questa procedura include le seguenti risorse:

- Un VPC e le risorse di rete associate, tra cui una sottorete, un gruppo di sicurezza, un gateway Internet e una tabella di routing.
- Un'istanza Amazon EC2; avviata nella sottorete nel VPC.
- Argomento Amazon SNS
- Due funzioni AWS Lambda. Queste funzioni ricevono messaggi che vengono pubblicati all'argomento Amazon SNS e registrano gli eventi in CloudWatch Logs.
- Parametri e registri di Amazon CloudWatch
- Un ruolo IAM; che consente all'istanza Amazon EC2 di utilizzare Amazon SNS e un ruolo IAM che consente alle funzioni Lambda; di scrivere nei CloudWatch logs.

Per creare le risorse AWS

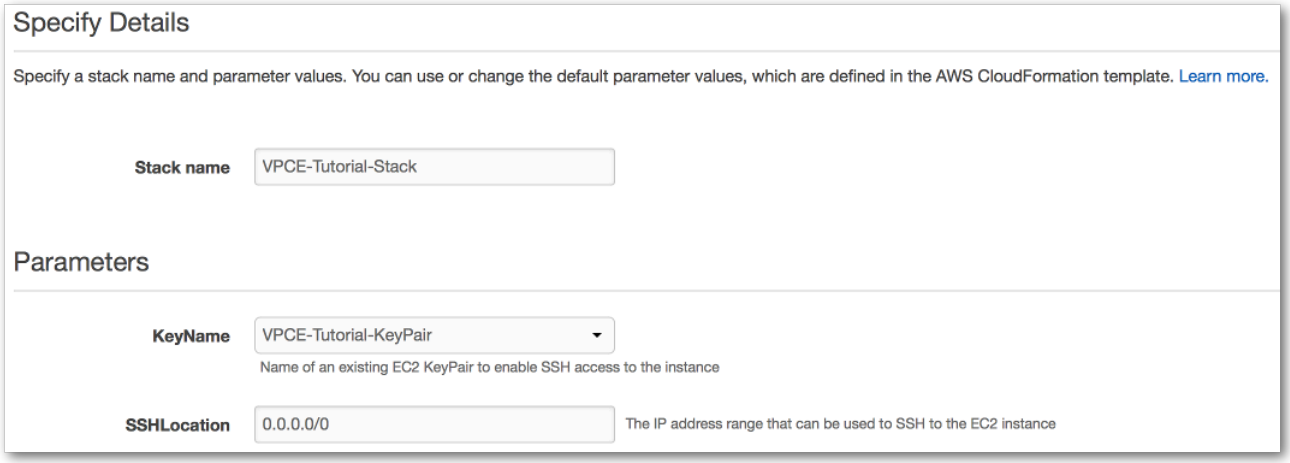
1. Scarica il file di modello dal sito Web di GitHub.
2. Accedi alla console AWS CloudFormation.

3. Scegli Create Stack (Crea stack).
4. Nella pagina Select Template (Seleziona modello) scegli Upload a template to Amazon S3 (Carica un modello in Amazon S3), seleziona il file, quindi scegli Next (Avanti).
5. Nella pagina Specify Details (Specifica dettagli), specifica i nomi dello stack e della chiave:

   a. Per Stack name (Nome stack), digitare **`VPCE-Tutorial-Stack`**.

   b. Per KeyName (Nome chiave), scegli VPCE-Tutorial-KeyPair.

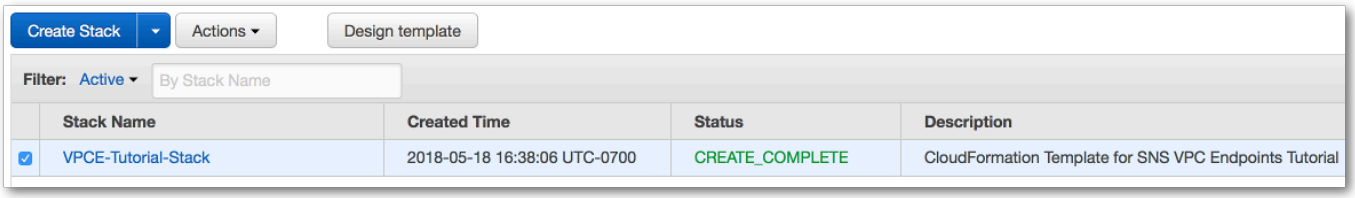
   c. Per SSHLocation, lascia il valore predefinito di **`0.0.0.0/0`**.



   d. Seleziona Next (Successivo).

6. Nella pagina Options (Opzioni), lascia tutti i valori predefiniti e scegli Next (Avanti).
7. Nella pagina Review (Rivedi), verifica i dettagli dello stack.
8. In Funzionalità, ricordare che AWS CloudFormation può creare risorse IAM con nomi personalizzati.
9. Scegliere Create (Crea).

   La console AWS CloudFormation apre la pagina Stacks (Stack). Il VPCE-Tutorial-Stack ha lo stato CREATE_IN_PROGRESS. Dopo pochi minuti, quando il processo di creazione è stato completato, lo stato diventa CREATE_COMPLETE.



> **Tip**
>
> Scegli il pulsante Refresh (Aggiorna) per visualizzare l'ultimo stato dello stack.

Fase 3: Conferma che l'istanza Amazon EC2; non dispone di accesso Internet

L'istanza Amazon EC2 che è stata avviata nel VPC nella fase precedente non dispone di accesso a Internet. Non consente il traffico in uscita e non è in grado di pubblicare messaggi a Amazon SNS. Verifica accedendo all'istanza. Quindi, prova a connetterti a un endpoint pubblico e ad inviare un messaggio a Amazon SNS.

A questo punto della procedura il tentativo di pubblicazione ha esito negativo. In un secondo momento, dopo aver creato un endpoint VPC per Amazon SNS il tentativo di pubblicazione va a buon fine.

Eseguire la connessione all'istanza di Amazon EC2.

1. Apri la console Amazon EC2 all'indirizzo https://console.aws.amazon.com/ec2/.
2. Nel menu di navigazione a sinistra, individua la sezione Instances (Istanze). Quindi scegli Instances (Istanze).
3. Nell'elenco delle istanze, seleziona VPCE-Tutorial-EC2Instance.
4. Copia il nome host fornito nella colonna Public DNS (IPv4).

5. Aprire un terminale. Dalla directory che contiene la coppia di chiavi, connettiti all'istanza utilizzando il comando seguente, dove *instance-hostname* è il nome host copiato dalla console Amazon EC2:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

Per verificare che l'istanza non dispone di una connessione a Internet

- Nel terminale, tenta di connetterti a un endpoint pubblico, ad esempio amazon.com:

```
$ ping amazon.com
```

Poiché il tentativo di connessione ha esito negativo, è possibile annullare in qualsiasi momento (Ctrl+C in Windows o Comando+C su macOS).

Per verificare che l'istanza non dispone di una connessione a Amazon SNS

1. Accedi alla console Amazon SNS.
2. Nel menu di navigazione a sinistra, scegli Topics (Argomenti).
3. Nella pagina Topics (Argomenti), copia l'Amazon Resource Name (ARN) per l'argomento VPCE-Tutorial-Topic.
4. Nel terminale, tenta di pubblicare un messaggio all'argomento:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Poiché il tentativo di pubblicazione ha esito negativo, è possibile annullare in qualsiasi momento.

Fase 4: Creazione di un endpoint Amazon VPC per Amazon SNS

Per collegare il tuo VPC a Amazon SNS, definisci un endpoint VPC dell'interfaccia. Dopo aver aggiunto l'endpoint, è possibile accedere all'istanza Amazon EC2 nel proprio VPC, quindi è possibile utilizzare l'API di Amazon SNS. È possibile pubblicare messaggi all'argomento che vengono pubblicati privatamente. Rimangono nella rete AWS e non viaggiano sulla rete Internet pubblica.

> **Note**
>
> L'istanza non dispone ancora dell'accesso ad altri servizi AWS ed endpoint su Internet.

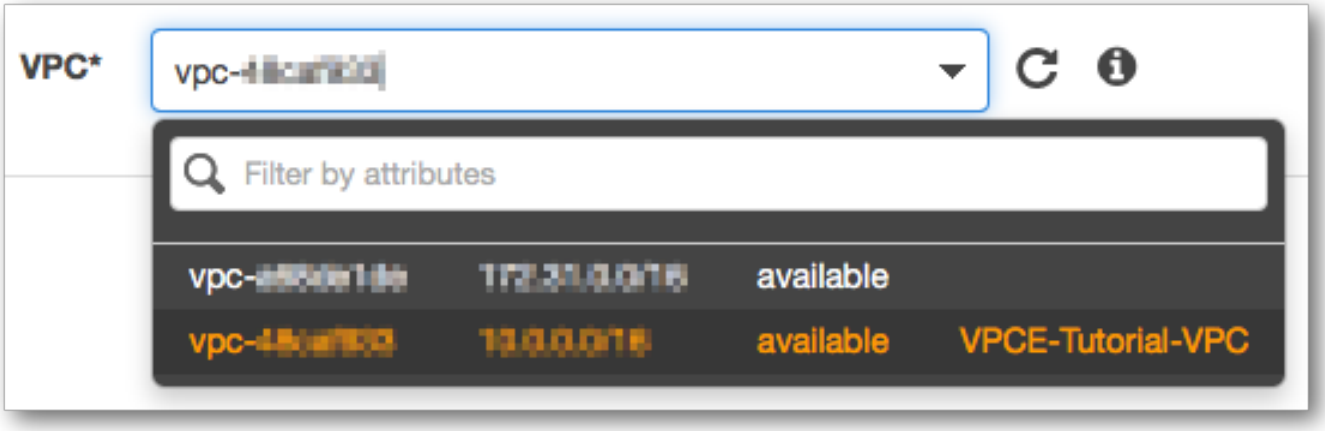
Per creare l'endpoint

1. Accedere alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel menu di navigazione a sinistra, scegli Endpoints.
3. Scegliere Create Endpoint (Crea endpoint).
4. Nella pagina Crea endpoint, per Categoria di servizio lascia la scelta predefinita di AWS Servizi.

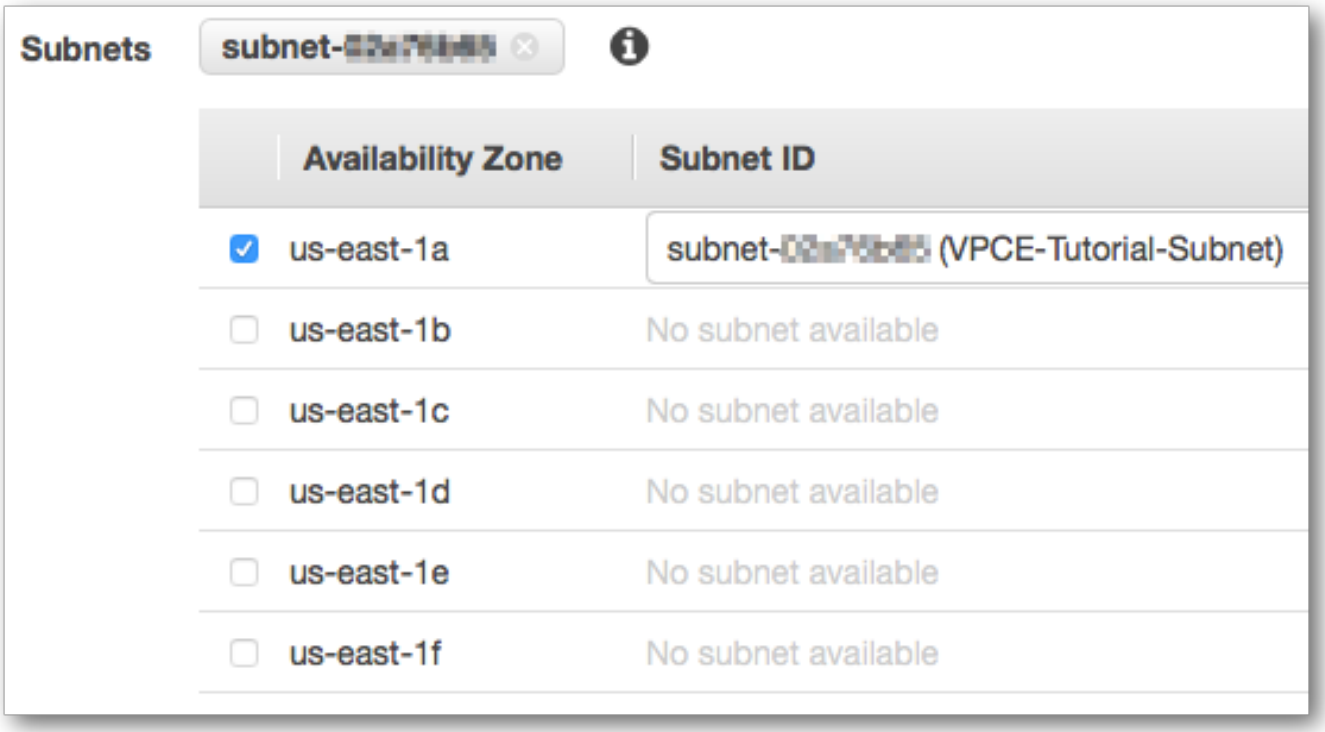
5. Per Nome servizio, scegli il nome del servizio per Amazon SNS;.

   I nomi dei servizi variano in base alla regione scelta. Ad esempio, se si sceglie Stati Uniti orientali (Virginia settentrionale), il nome del servizio è com.amazonaws.***us-east-1***.sns.

6. Per VPC, scegli il VPC che ha il nome VPCE-Tutorial-VPC.

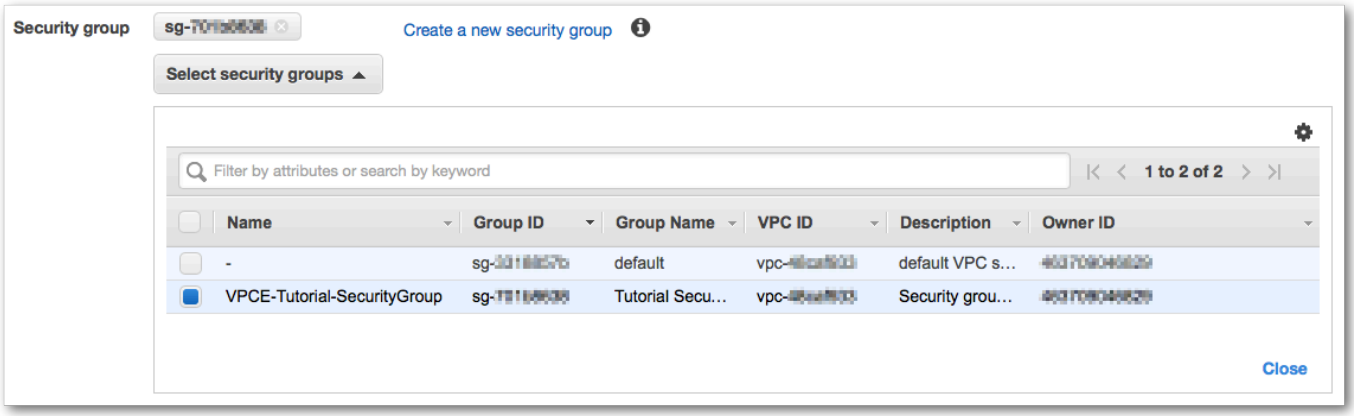


7. Per Subnets (Sottoreti), scegli la sottorete che ha VPCE-Tutorial-Subnet nell'ID della sottorete.

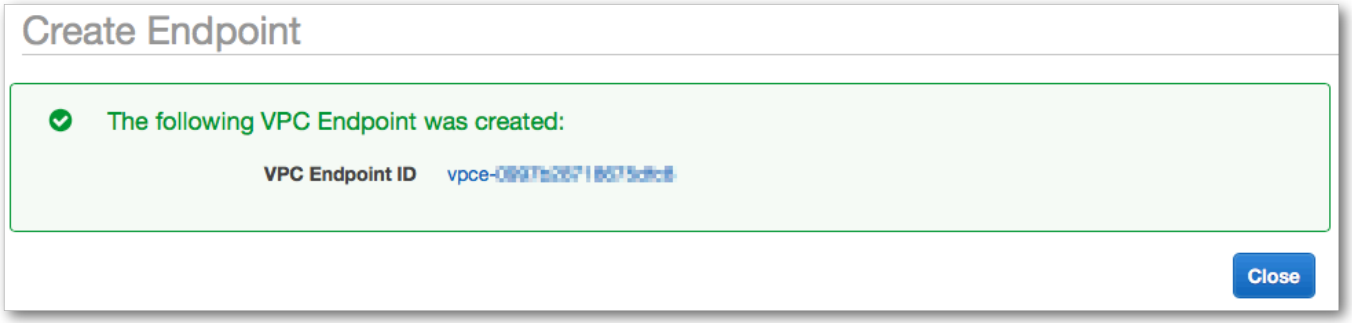


8. Per Enable Private DNS Name (Abilita nome DNS privato), seleziona Enable for this endpoint (Abilita per questo endpoint).

9. Per Security group (Gruppo di sicurezza), scegli Select security group (Seleziona gruppo di sicurezza) e seleziona VPCE-Tutorial-SecurityGroup.

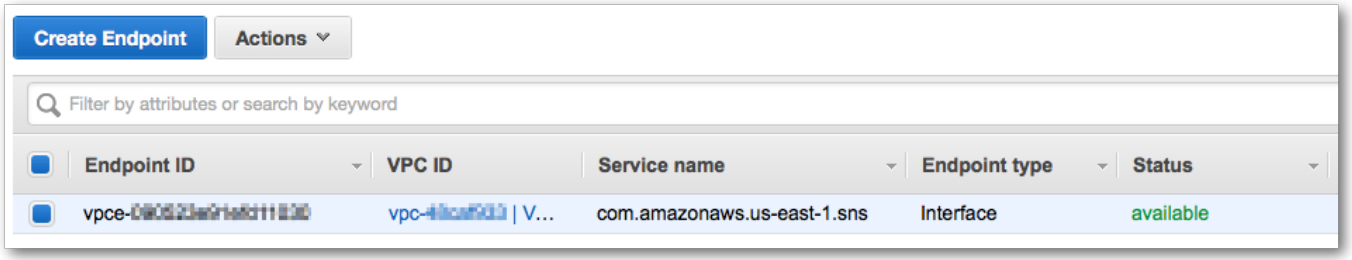


10. Selezionare Create endpoint (Crea endpoint). La console Amazon VPC conferma che è stato creato un endpoint VPC.



11. Scegli Close (Chiudi).

    La console Amazon VPC apre la pagina Endpoints. Il nuovo endpoint ha lo stato pending (in sospeso). Dopo pochi minuti, quando il processo di creazione è stato completato, lo stato diventa available (disponibile).



Fase 5: Pubblicazione di un messaggio nel proprio argomento Amazon SNS

Ora che il VPC include un endpoint per Amazon SNS, può possibile accedere all'istanza Amazon EC2 e pubblicare messaggi nell'argomento.

Per pubblicare un messaggio

1. Se il terminale non è più connesso all'istanza Amazon EC2, effettuare nuovamente il collegamento:

   ```
   $ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
   ```

2. Eseguire lo stesso comando eseguito precedentemente per pubblicare un messaggio nel proprio argomento Amazon SNS. Questa volta, il tentativo di pubblicazione va a buon fine e Amazon SNS; restituisce un ID messaggio:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

   {
       "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
   }
   ```

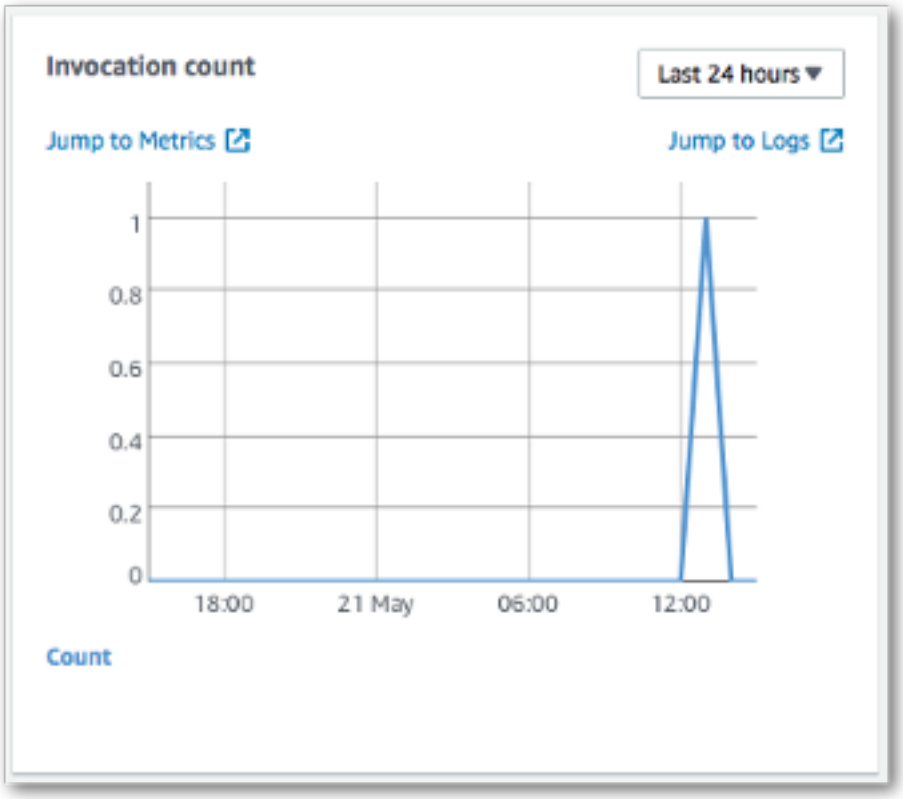
Fase 6: verifica delle consegne del messaggio

Quando l'argomento Amazon SNS riceve un messaggio, distribuisce il messaggio inviandolo alle due funzioni Lambda di sottoscrizione. Quando queste funzioni ricevono il messaggio, registrano l'evento nei CloudWatch logs. Per verificare che la consegna del messaggio sia riuscita, controllare che le funzioni siano state richiamate e che i CloudWatch logs siano stati aggiornati.

Per verificare che le funzioni Lambda; siano state richiamate

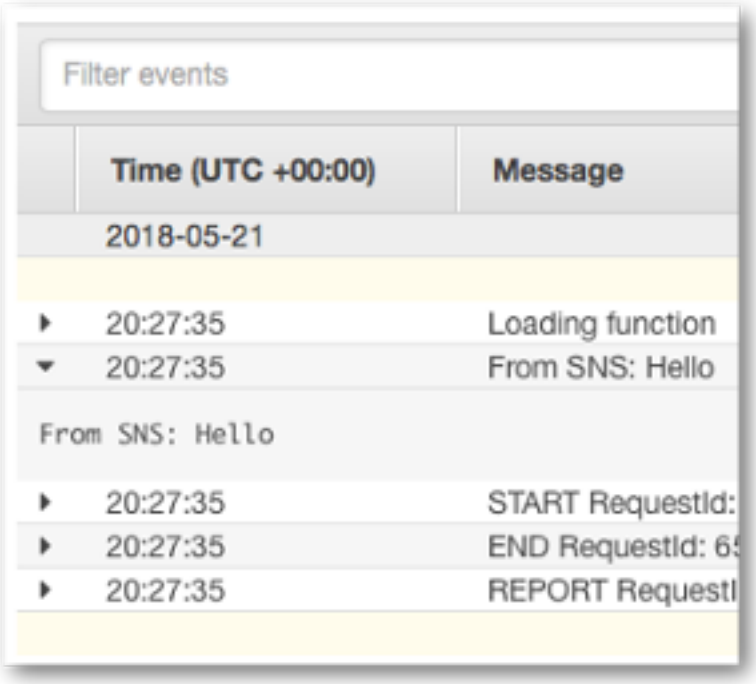
1. Apri la console AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Nella pagina Functions (Funzioni), scegli VPCE-Tutorial-Lambda-1.
3. Selezionare Monitoring (Monitoraggio).
4. Controlla il grafico Invocation count (Conteggio invocazioni). Questo grafico mostra il numero di volte che la funzione Lambda; è stata eseguita.

   Il conteggio di invocazioni corrisponde al numero di volte in cui è stato pubblicato un messaggio all'argomento.



Per verificare che i CloudWatch logs siano stati aggiornati

1. Aprire la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Nel menu di navigazione a sinistra, scegli Logs.
3. Controlla i log scritti dalle funzioni Lambda:
    a. Scegli il gruppo di log /aws/lambda/VPCE-Tutorial-Lambda-1/.
    b. Scegli il flusso di log.
    c. Verifica che il log includa la voce `From SNS: Hello`.



d. Scegli Log Groups (Gruppi di log) nella parte superiore della console per tornare alla pagina Log Groups (Gruppi di log). Quindi ripeti le fasi precedenti per il gruppo di log /aws/lambda/VPCE-Tutorial-Lambda-2/.

Complimenti! Aggiungendo un endpoint per Amazon SNS a un VPC, è possibile pubblicare un messaggio in un argomento all'interno della rete gestita dal VPC. Il messaggio è stato pubblicato privatamente senza essere esposto sulla rete Internet pubblica.

Passaggio 7: pulizia

Se non si desidera conservare le risorse create, è possibile eliminarle ora. Eliminando le risorse AWS che non si utilizzano più, è possibile evitare addebiti superflui sul proprio account Account AWS.

In primo luogo, elimina l'endpoint VPC utilizzando la console Amazon VPC. Quindi elimina le altre risorse create eliminando lo stack nella console AWS CloudFormation. Quando si elimina uno stack, AWS CloudFormation rimuove le risorse dello stack dal proprio Account AWS.

Per eliminare l'endpoint VPC

1. Accedere alla console Amazon VPC all'indirizzo https://console.aws.amazon.com/vpc/.
2. Nel menu di navigazione a sinistra, scegli Endpoints.
3. Seleziona l'endpoint creato.

4. Scegli Actions (Operazioni), quindi Delete Endpoint (Elimina endpoint).
5. Nella finestra Delete Endpoint (Elimina endpoint), scegli Yes, Delete (Sì, elimina).

   Lo stato dell'endpoint diventa deleting (in eliminazione). Quando l'eliminazione viene completata, l'endpoint viene rimosso dalla pagina.

Per eliminare lo stack AWS CloudFormation

1. aprire la console di AWS CloudFormation all'indirizzo https://console.aws.amazon.com/cloudformation.
2. Selezionare lo stack VPCE-Tutorial-Stack.
3. Scegliere Actions (Operazioni), quindi Delete Stack (Elimina stack).
4. Nella finestra Delete Stack (Elimina stack), scegli Yes, Delete (Sì, elimina).

   Lo stato dello stack diventa DELETE_IN_PROGRESS. Quando l'eliminazione viene completata, lo stack viene rimosso dalla pagina.

Risorse correlate

Per ulteriori informazioni, consulta le risorse seguenti.

- Blog sulla sicurezza di AWS: protezione dei messaggi pubblicati su Amazon SNS con PrivateLink di AWS
- Che cos'è Amazon VPC?
- Endpoint VPC
- Che cos'è Amazon EC2?
- AWS CloudFormation Concetti correlati a

Sicurezza della protezione dei dati dei messaggi

- Rispetto a quanto avviene per i dati a riposo, la protezione dei dati dei messaggi è una caratteristica di Amazon SNS utilizzata per definire regole e policy personalizzate per l'audit e il controllo del contenuto dei dati in transito.
- La protezione dei dati dei messaggi fornisce servizi di governance, conformità e audit per le applicazioni aziendali basate su messaggi, in modo che l'ingresso e l'uscita dei dati possano

essere controllati dal proprietario dell'argomento di Amazon SNS e i flussi di contenuti possano essere tracciati e registrati.

- Puoi scrivere regole di governance basate sul payload per impedire l'ingresso dei contenuti non autorizzati del payload nei flussi di messaggi.
- Puoi concedere diverse autorizzazioni di accesso ai contenuti ai singoli iscritti e controllare l'intero processo del flusso dei contenuti.

Identity and Access Management in Amazon SNS

L'accesso a Amazon SNS richiede credenziali che AWS può utilizzare per autenticare le richieste. Queste credenziali devono disporre delle autorizzazioni per accedere alle risorse AWS, come argomenti e messaggi Amazon SNS. Le sezioni che seguono forniscono informazioni su come utilizzare AWS Identity and Access Management (IAM) e Amazon SNS per proteggere le proprie risorse attraverso il controllo degli accessi.

AWS Identity and Access Management (IAM) è un Servizio AWSche consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. Gli amministratori IAM controllano chi può essere autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) per utilizzare le risorse Amazon SNS. IAM è un Servizio AWS il cui uso non comporta costi aggiuntivi.

Destinatari

Le modalità di utilizzo di AWS Identity and Access Management (IAM) cambiano in base alle operazioni eseguite in Amazon SNS.

Utente del servizio: se si utilizza il servizio Amazon SNS per svolgere il proprio lavoro, l'amministratore fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di caratteristiche Amazon SNS utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità in Amazon SNS, consulta Risoluzione dei problemi di identità e accesso ad Amazon Simple Notification Service.

Amministratore del servizio: se si è responsabile delle risorse Amazon SNS presso la propria azienda, probabilmente si dispone dell'accesso completo a Amazon SNS. Il compito dell'utente è determinare le caratteristiche e le risorse di Amazon SNS a cui gli utenti del servizio devono accedere. 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 ulteriori informazioni su come la tua azienda può utilizzare IAM con Amazon SNS, consulta Come funziona Amazon Simple Notification Service.

Amministratore IAM: se si è amministratore IAM, potrebbe essere interessante ottenere informazioni su come scrivere policy per gestire l'accesso ad Amazon SNS. Per visualizzare policy basate su identità Amazon SNS di esempio che possono essere utilizzate in IAM, consulta Esempi di policy basate su identità per Amazon Simple Notification Service.

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWS con le credenziali di identità. È necessario essere autenticato (connesso a AWS) come utente root Account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite tramite un'origine di identità. AWS IAM Identity Center (Centro identità IAM), l'autenticazione Single Sign-On (SSO) dell'azienda e le 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. Se accedi ad AWS tramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere alla AWS Management Console o al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione Come accedere al tuo Account AWS nella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consulta Firma delle richieste AWS nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Utente root di un Account AWS

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWS e le risorse nell'account. Tale identità è detta utente root Account AWS ed è

possibile accedervi 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 utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Identità federata

Come best practice, richiedi agli utenti umani, compresi quelli che richiedono l'accesso di amministratore, di utilizzare la federazione con un provider di identità per accedere a Servizi AWSutilizzando credenziali temporanee.

Un'identità federata è un utente della directory degli utenti aziendali, un provider di identità Web, AWS Directory Service, la directory Identity Center o qualsiasi utente che accede ai Servizi AWS utilizzando le credenziali fornite tramite un'origine di identità. Quando le identità federate accedono agli Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. È possibile creare utenti e gruppi in IAM Identity Center oppure connettersi e sincronizzarsi con un gruppo di utenti e gruppi nell'origine di identità per utilizzarli in tutte le applicazioni e gli Account AWS. Per ulteriori informazioni sul Centro identità IAM, consulta Cos'è Centro identità IAM? nella Guida per l'utente di AWS IAM Identity Center.

Utenti e gruppi IAM

Un utente IAM è una identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità all'interno di un Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Console mediante lo scambio di ruoli. È possibile assumere un ruolo chiamando un'azione AWS CLI o API AWS oppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center.
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.
- Accesso multi-servizio: alcuni Servizi AWS utilizzano funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione

utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.

- Inoltro delle sessioni di accesso (FAS): quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, tale utente o ruolo viene considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.
- Ruolo di servizio: un ruolo di servizio è un ruolo IAM assunto da un servizio per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'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: è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 che eseguono richieste di AWS CLIo dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWS a un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWS che, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste policy quando un principale IAM (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene archiviata in AWSsotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWSJSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLI o l'API AWS.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWS e le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy gestite da AWS da IAM in una policy basata su risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano le ACL. Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizationsè un servizio per il raggruppamento e la gestione centralizzata degli Account

AWSmultipli di proprietà dell'azienda. Se si abilitano tutte le caratteristiche in un'organizzazione, è possibile applicare le policy di controllo dei servizi (SCP) a uno o tutti i propri account. Una SCP limita le autorizzazioni per le entità negli account membri, compreso ogni utente root Account AWS. Per ulteriori informazioni su Organizations e le policy SCP, consulta Utilizzo delle SCP nella Guida per l'utente di AWS Organizations.

- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella Guida per l'utente di IAM.

Controllo accessi

Amazon SNS dispone del proprio sistema di autorizzazioni basate sulla risorsa che utilizza le policy scritte nello stesso linguaggio utilizzato per le policy (IAM) AWS Identity and Access Management. Questo significa che puoi ottenere risultati simili con le policy Amazon SNS e IAM.

> **Note**
>
> È importante capire che tutti gli Account AWS possono delegare le autorizzazioni agli utenti sotto i loro account. L'accesso tra account consente di condividere l'accesso a risorse AWS senza dover gestire utenti aggiuntivi. Per ulteriori informazioni sull'utilizzo dell'accesso multi-account, consultare la sezione relativa all'abilitazione dell'accesso multiaccount nella Guida dell'utente IAM.

Panoramica sulla gestione degli accessi in Amazon SNS

In questa sezione vengono descritti i concetti di base necessari per l'utilizzo del linguaggio della policy di accesso per scrivere le policy. Viene inoltre descritto il processo generale per la modalità

di funzionamento del controllo accessi con il linguaggio della policy di accesso e le modalità di valutazione delle policy.

Argomenti

Quando usare il controllo accessi

Hai a disposizione una grande flessibilità nel modo in cui concedere o rifiutare l'accesso a una risorsa. Tuttavia, i casi d'uso tipici sono abbastanza semplici:

- Permettere a un altro Account AWS un particolare tipo di operazione di argomento (ad esempio, Pubblica). Per ulteriori informazioni, consulta Permettere a Account AWS di accedere a un argomento.
- Limitare le sottoscrizioni all’argomento solo per il protocollo HTTPS. Per ulteriori informazioni, consulta Limitazione delle sottoscrizioni a HTTPS.
- Permettere ad Amazon SNS di pubblicare i messaggi nella coda Amazon SQS. Per ulteriori informazioni, consulta Pubblicare messaggi in una coda Amazon SQS..

Concetti chiave

Nelle seguenti sezioni vengono descritti i concetti necessari per l’utilizzo del linguaggio della policy di accesso. Sono presentati secondo un ordinamento logico, con i primi termini che devi conoscere in cima all'elenco.

Argomenti

- Principale
- Azione
- Risorsa
- Condizioni e chiavi
- Richiedente
- Valutazione
- Effetto
- Rifiuto per default
- Consenso
- Rifiuto esplicito

Autorizzazione

L'autorizzazione è il concetto che permette o rifiuta un tipo di accesso a una particolare risorsa. Le autorizzazioni seguono essenzialmente questo modulo: "A ha/non ha l'autorizzazione di eseguire B in C se si applica D". Ad esempio, Jane (A) ha l'autorizzazione a pubblicare (B) in TopicA (C) se usa il protocollo HTTP (D). Ogni volta che Jane pubblica in TopicA, il servizio verifica se ha l'autorizzazione e se la richiesta soddisfa le condizioni stabilite nell'autorizzazione.

Dichiarazione

Una dichiarazione è la descrizione formale di una singola autorizzazione, scritta nel linguaggio della policy di accesso. Scrivi sempre una dichiarazione nell'ambito di un documento container più ampio denominato policy (vedi il prossimo concetto).

Policy

Una policy è un documento (scritto nella sintassi della/e policy di accesso) che funge da container per una o più dichiarazioni. Ad esempio, una policy potrebbe avere due dichiarazioni: una che afferma che Jane può iscriversi utilizzando il protocollo e-mail e un'altra che afferma che Bob non può pubblicare in Topic A. Come mostrato nella figura seguente, uno scenario equivalente potrebbe avere due policy, una che afferma che Jane può iscriversi utilizzando il protocollo e-mail e un altro che afferma che Bob non può pubblicare su Topic A.

Solo i caratteri ASCII sono ammessi nei documenti delle policy. È possibile utilizzare `aws:SourceAccount` e `aws:SourceOwner` per aggirare lo scenario in cui è necessario collegare altri ARN dei servizi AWS che contengono caratteri non ASCII. vedi la differenza tra `aws:SourceAccount` rispetto a `aws:SourceOwner`.

Emittente

L'emittente è la persona che scrive una policy per concedere le autorizzazioni per una risorsa. L'approvatore, per definizione, è sempre il proprietario della risorsa. AWS non consente AWS agli utenti di creare policy per risorse che non sono di loro proprietà. Se John è il proprietario della risorsa, AWS autentica l'identità di John quando invia la policy che ha scritto per concedere le autorizzazioni a tale risorsa.

Principale

Il principale è la persona o le persone che ricevono l'autorizzazione nella policy. Il principale è A nella dichiarazione "A ha l'autorizzazione di eseguire B in C se si applica D". In una policy, puoi impostare il principale su "chiunque" (ovvero, puoi specificare un carattere jolly per rappresentare tutte le persone). Potresti farlo, ad esempio, se non vuoi limitare l'accesso in base all'identità effettiva del richiedente, ma in base ad altre caratteristiche identificative come l'indirizzo IP del richiedente.

Azione

L'operazione è l'attività che il principale è autorizzato a eseguire. L'operazione è B nella dichiarazione "A dispone dell'autorizzazione di eseguire B in C se si applica D". In genere, l'operazione è solo l'operazione nella richiesta a AWS. Ad esempio, Jane invia una richiesta ad Amazon SNS; con `Action=Subscribe`. Puoi specificare una o più operazioni in una policy.

Risorsa

La risorsa è l'oggetto al quale il principale richiede l'accesso. La risorsa è C nella dichiarazione "A ha l'autorizzazione di eseguire B in C se si applica D".

Condizioni e chiavi

Le condizioni sono restrizioni o dettagli relativi all'autorizzazione. La condizione è D nella dichiarazione "A ha l'autorizzazione di eseguire B in C se si applica D". La parte della policy che specifica le condizioni può essere la più dettagliata e complessa di tutte le parti. Le condizioni tipiche sono correlate a:

- Data e ora (ad esempio, la richiesta deve arrivare prima di un giorno specifico)
- Indirizzo IP (ad esempio, l'indirizzo IP del richiedente deve far parte di un particolare intervallo CIDR)

La chiave è la caratteristica specifica che costituisce la base per la limitazione degli accessi. Ad esempio, la data e l'ora della richiesta.

Usa insieme condizioni e chiavi per esprimere la limitazione. Il modo più semplice per capire come si implementa effettivamente una limitazione è con un esempio: se vuoi limitare l'accesso a prima del 30 maggio 2010, utilizzi la condizione chiamata `DateLessThan`. Utilizzare la chiave chiamata `aws:CurrentTime` e impostarla sul valore `2010-05-30T00:00:00Z`. AWS definisce le condizioni e le chiavi che è possibile utilizzare. Anche il servizio AWS stesso (ad esempio, Amazon SQS o Amazon SNS) può definire le chiavi specifiche del servizio. Per ulteriori informazioni, consulta Autorizzazioni API Amazon SNS: riferimento a operazioni e risorse.

Richiedente

Il richiedente è la persona che invia una richiesta a un servizio AWS e richiede l'accesso a una particolare risorsa. Il richiedente invia una richiesta a AWS che dice essenzialmente: "Mi concedi l'autorizzazione per eseguire B in C se si applica D?"

Valutazione

La valutazione è il processo utilizzato dal servizio AWS per determinare se una richiesta in arrivo debba essere rifiutata o consentita in base alle policy applicabili. Per ulteriori informazioni sulla logica della valutazione, consulta Logica della valutazione.

Effetto

L'effetto è il risultato che una dichiarazione della policy vuoi che restituisca al momento della valutazione. Specifica questo valore quando scrivi le dichiarazioni in una policy e i valori possibili sono diniego e permettere.

Ad esempio, potresti scrivere una policy che ha una dichiarazione che rifiuta tutte le richieste provenienti dall'Antartide (effect=deny, presupponendo che la richiesta utilizzi un indirizzo IP assegnato all'Antartide). In alternativa, potresti scrivere una policy che ha una dichiarazione che consente tutte le richieste che non sono provenienti dall'Antartide (effect=allow, presupponendo che la richiesta non provenga dall'Antartide). Sebbene le due dichiarazioni sembrano facciano la stessa cosa, nella logica del linguaggio della policy di accesso sono diverse. Per ulteriori informazioni, consulta Logica della valutazione.

Sebbene ci siano solo due possibili valori che puoi specificare per l'effetto (allow o deny), ci possono essere tre diversi risultati al momento della valutazione della policy: rifiuto per default, consenso o rifiuto esplicito. Per ulteriori informazioni, vedi i seguenti concetti e Logica della valutazione.

Rifiuto per default

Rifiuto per default è il risultato predefinito di una policy in assenza di un consenso o di un rifiuto esplicito.

Consenso

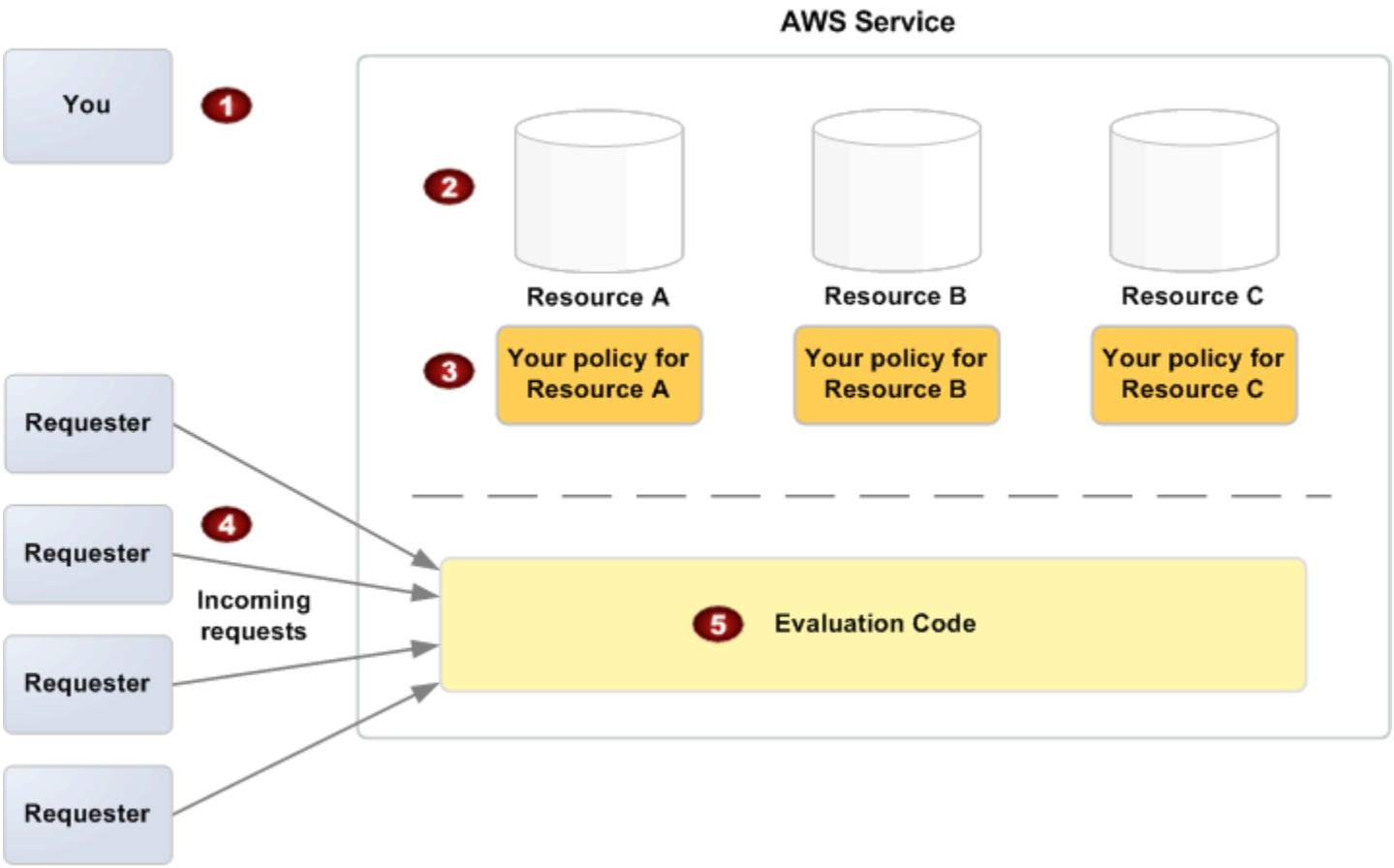
Un consenso viene restituito da una dichiarazione che ha effect=allow, presupponendo che tutte le condizioni dichiarate siano soddisfatte. Esempio: consenti le richieste se vengono ricevute prima delle 13:00 del 30 aprile 2010. Un consenso sovrascrive tutti i rifiuti per default, ma mai un rifiuto esplicito.

Rifiuto esplicito

Il rifiuto esplicito viene restituito da una dichiarazione che ha effect=deny, presupponendo che tutte le condizioni dichiarate siano soddisfatte. Esempio: rifiuta tutte le richieste se provengono dall'Antartide. Qualsiasi richiesta proveniente dall'Antartide sarà sempre rifiutata, a prescindere da eventuali altre policy.

Panoramica dell'architettura

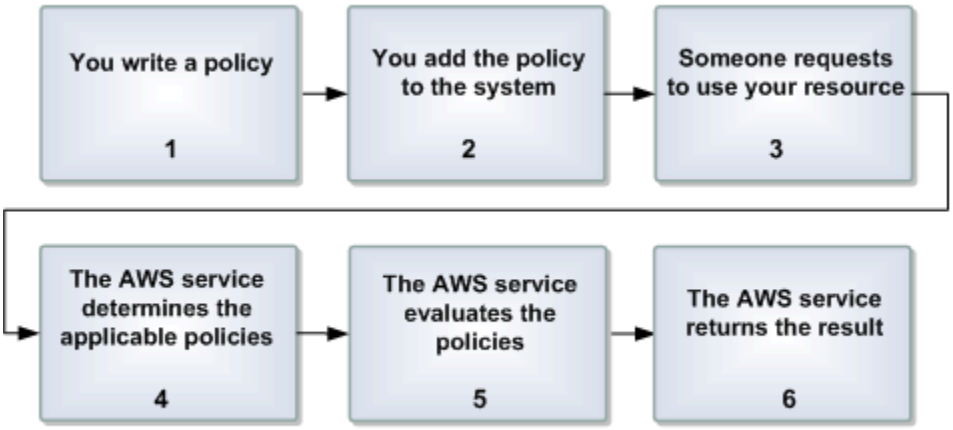
Nella seguente figura e nella tabella vengono descritti i componenti principali che interagiscono per fornire il controllo accessi alle risorse.



1	Tu, il proprietario delle risorse.
2	Le tue risorse (contenute nel servizio AWS; ad esempio le code Amazon SQS).
3	Le tue policy. In genere hai una policy per risorsa, ma possono essere di più. Il servizio AWS stesso fornisce un'API da utilizzare per caricare e gestire le proprie policy.
4	I richiedenti e le loro richieste in arrivo per il servizio AWS.
5	Codice di valutazione del linguaggio della policy di accesso. Si tratta del set di codici interni al servizio AWS che valuta le richieste in arrivo rispetto alle policy applicabili e determina se il richiedente può accedere alla risorsa. Per informazioni su come il servizio prende le decisioni, vedi Logica della valutazione.

Utilizzo del linguaggio della policy di accesso

Le seguenti figura e tabella descrivono il processo generale riguardante il funzionamento del controllo accessi con il linguaggio della policy di accesso.



Processo per l'utilizzo del controllo accessi con il linguaggio della policy di accesso

1	Scrivi una policy per la tua risorsa. Ad esempio, scrivi una policy per specificare le autorizzazioni per i tuoi argomenti Amazon SNS.
2	Caricare la policy in AWS. Il servizio AWS stesso fornisce un'API da utilizzare per caricare le proprie policy. Ad esempio, usare l'operazione di Amazon SNS `SetTopicAttributes` per caricare una policy per un particolare argomento Amazon SNS.
3	Qualcuno invia una richiesta per usare la tua risorsa. Ad esempio, un utente invia una richiesta ad Amazon SNS; per usare uno dei tuoi argomenti.
4	Il servizio AWS determina quali policy sono applicabili alla richiesta. Ad esempio, Amazon SNS guarda tutte le policy Amazon SNS disponibili e determina quali sono applicabili (in base a cosa è la risorsa, chi è il richiedente e così via).
5	Il servizio AWS valuta le policy.

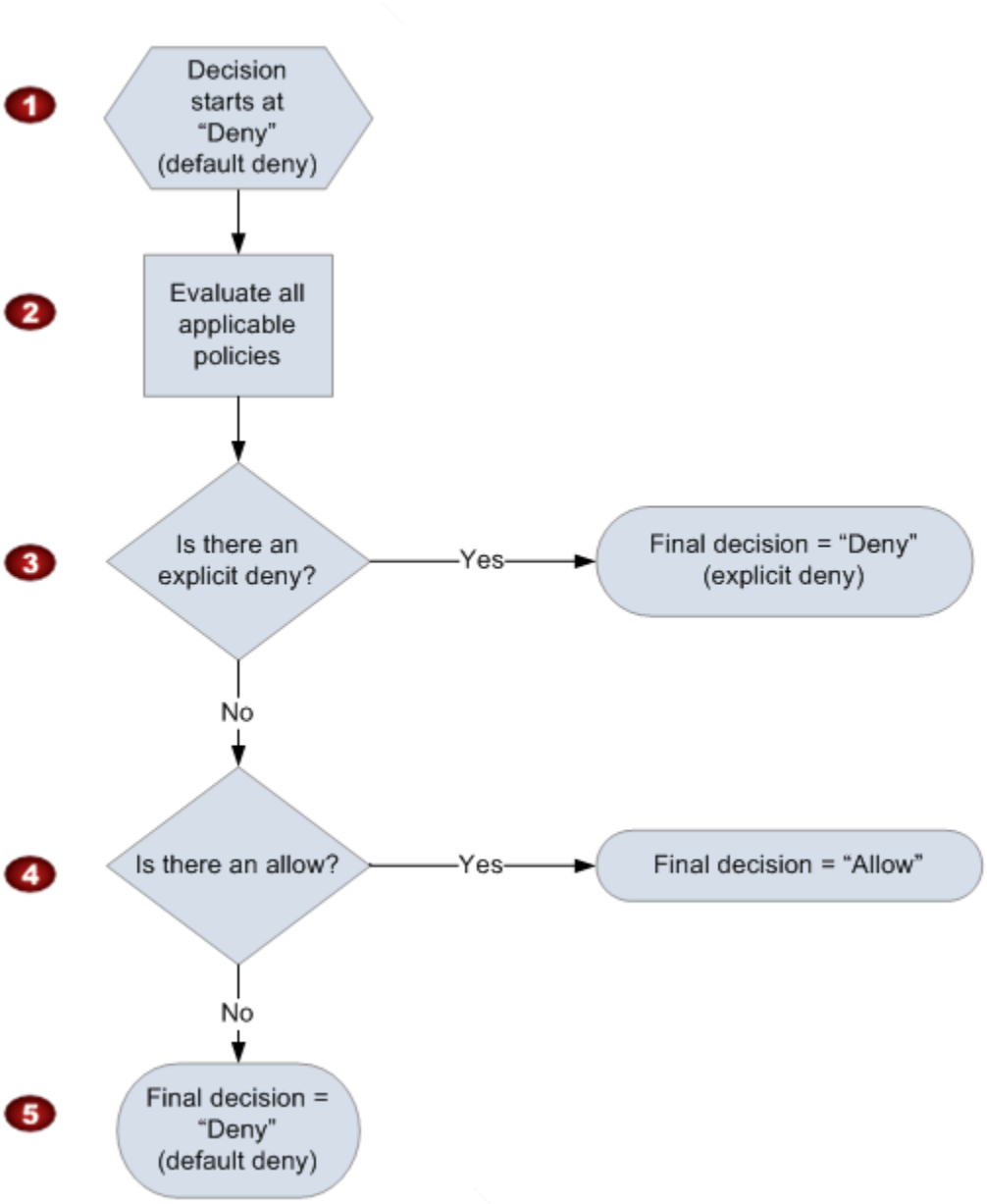
	Ad esempio, Amazon SNS valuta le policy e determina se il richiedente è autorizzato o meno ad utilizzare il tuo argomento. Per ulteriori informazioni sulla logica della decisione, consulta Logica della valutazione.
6	Il servizio AWS rifiuta la richiesta o continua ad elaborarla. Ad esempio, in base al risultato della valutazione della policy, il servizio restituisce un errore di "Accesso rifiutato" al richiedente o continua a elaborare la richiesta.

Logica della valutazione

L'obiettivo al momento della valutazione consiste nel decidere se una determinata richiesta deve essere autorizzata o rifiutata. La logica della valutazione segue diverse regole di base:

- Di default, tutte le richieste per utilizzare la tua risorsa che provengono da chiunque eccetto te, vengono rifiutate
- Un consenso sovrascrive tutti i rifiuti per default
- Un rifiuto esplicito sovrascrive tutti i consensi
- L'ordine in cui vengono valutate le policy non è importante

Il seguente diagramma di flusso e la discussione descrivono in modo più dettagliato come viene presa la decisione.



1	La decisione inizia con un rifiuto per default.
2	Il codice di applicazione quindi valuta tutte le policy applicabili alla richiesta (in base alla risorsa, al principale, all'operazione e alle condizioni). L'ordine in cui il codice di attuazione valuta le policy non è importante.
3	In tutte queste policy, il codice di applicazione cerca un'istruzione di rifiuto esplicito che applicherebbe alla richiesta.

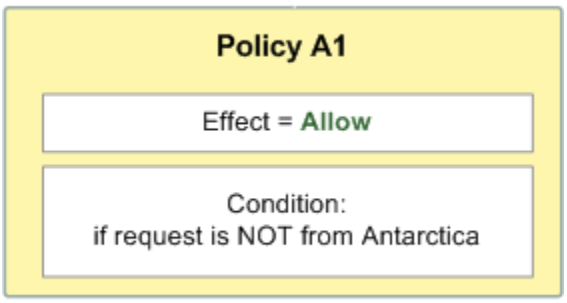
	Se ne trova anche uno, il codice di applicazione restituisce una decisione di "rifiuto" e il processo è finito (questo è un rifiuto esplicito; per ulteriori informazioni, vedi Rifiuto esplicito).
4	Se non viene trovato un rifiuto esplicito, il codice di applicazione cerca un'istruzione di "consenso" da applicare alla richiesta. Se ne trova anche uno, il codice di applicazione restituisce una decisione di "consenso" e il processo è completato (il servizio continua a elaborare la richiesta).
5	Se non viene trovato alcun consenso, la decisione finale è il "rifiuto" e dal momento che non è stato trovato un rifiuto esplicito o un consenso, questo viene considerato un rifiuto per default (per ulteriori informazioni, vedi Rifiuto per default).

L'interazione tra rifiuti espliciti e per default

Una policy restituisce un rifiuto per default se non si applica direttamente alla richiesta. Ad esempio, se un utente richiede di utilizzare Amazon SNS, ma la policy sull'argomento non fa alcun riferimento all'utente Account AWS, tale policy genera un rifiuto per default.

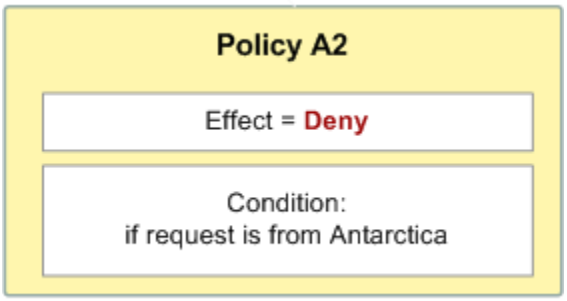
Una policy restituisce un rifiuto per default anche quando una condizione in una dichiarazione non viene soddisfatta. Se tutte le condizioni nella dichiarazione sono soddisfatte, la policy restituisce un consenso o un rifiuto esplicito, a secondo del valore dell'elemento Effetto nella policy. Le policy non specificano cosa fare se una condizione non viene soddisfatta e quindi il risultato di default in quel caso è un rifiuto per default.

Ad esempio, supponiamo che tu voglia impedire le richieste provenienti dall'Antartide. Scrivi una policy (chiamata Policy A1) che consente una richiesta solo se non proviene dall'Antartide. Il diagramma seguente illustra la policy.



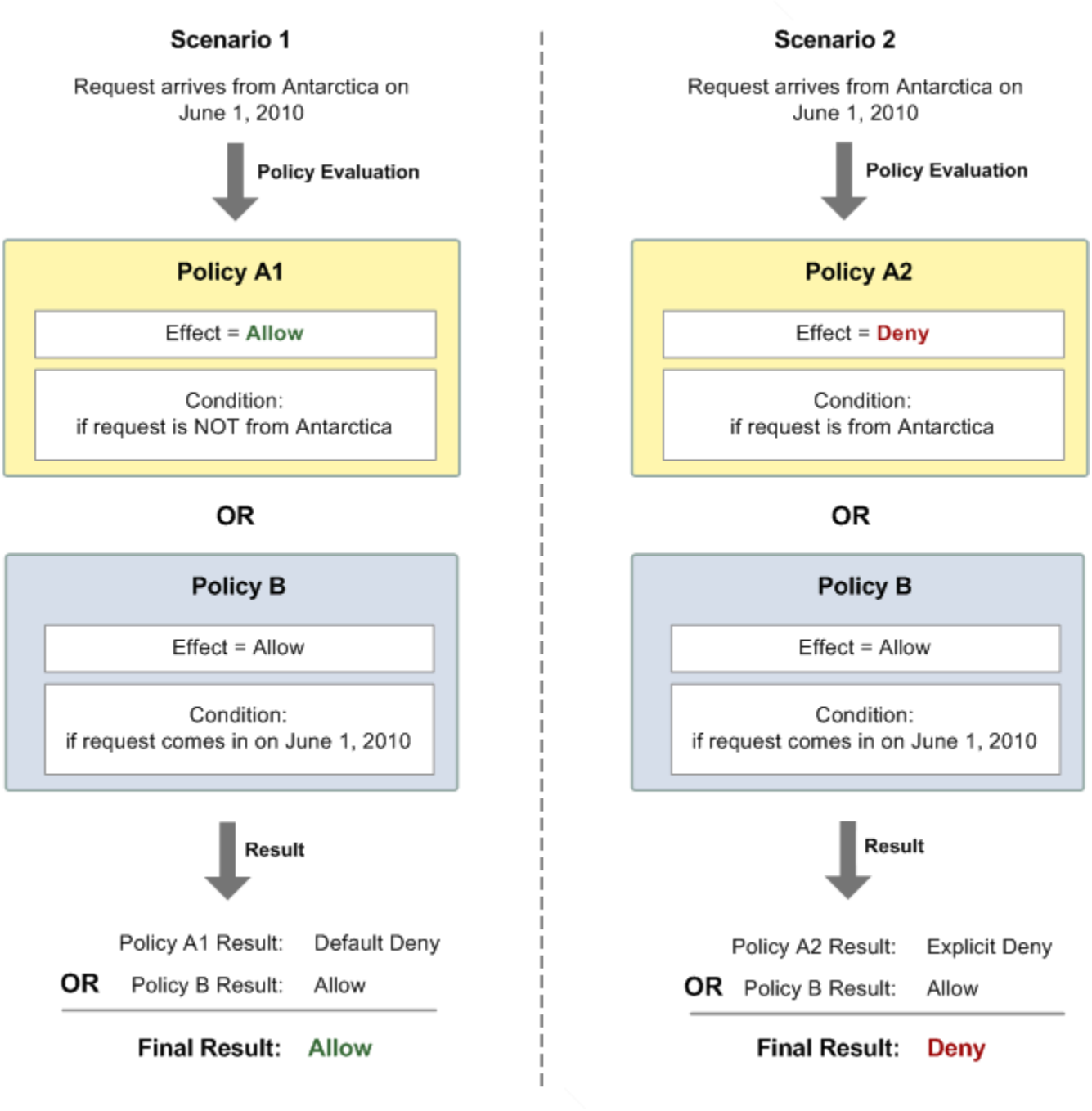
Se qualcuno invia una richiesta dagli Stati Uniti, la condizione è soddisfatta (la richiesta non proviene dall'Antartide). Pertanto, la richiesta è consentita. Tuttavia, se qualcuno invia una richiesta dall'Antartide, la condizione non viene soddisfatta e il risultato della policy è quindi un rifiuto per default.

Puoi trasformare il risultato in un rifiuto esplicito riscrivendo la policy (denominata Policy A2) come nel diagramma seguente. In questo caso, la policy rifiuta esplicitamente una richiesta se proviene dall'Antartide.



Se qualcuno invia una richiesta dall'Antartide, la condizione viene soddisfatta e il risultato della policy è quindi un rifiuto esplicito.

La differenza tra un rifiuto per default e un rifiuto esplicito è importante perché un rifiuto per default può essere sovrascritto da un consenso, mentre un rifiuto esplicito no. Ad esempio, supponiamo che ci sia un'altra policy che consente le richieste se arrivano il 1° giugno 2010. In che modo questa policy influisce sul risultato complessivo se abbinata alla policy che limita l'accesso dall'Antartide? Confronteremo il risultato complessivo quando abbiniamo la policy basata sulla data (che chiameremo policy B) con le precedenti policy A1 e A2. Lo scenario 1 abbina la policy A1 con la policy B e lo scenario 2 abbina la policy A2 con la policy B. La seguente figura e la discussione mostrano i risultati quando arriva una richiesta dall'Antartide il 1 giugno 2010.



Nello scenario 1, la policy A1 restituisce un rifiuto per default, come descritto in precedenza in questa sezione. La policy B restituisce un consenso perché la policy (per definizione) consente le richieste che arrivano il 1 giugno 2010. Il consenso dalla policy B sovrascrive il rifiuto per default dalla policy A1 e pertanto la richiesta è consentita.

Nello scenario 2, la policy A2 restituisce un rifiuto esplicito, come descritto in precedenza in questa sezione. Anche in questo caso, la policy B restituisce un consenso. Il rifiuto esplicito dalla policy A2 sovrascrive il consenso dalla policy B e pertanto la richiesta viene rifiutata.

Esempi di casi per il controllo degli accessi Amazon SNS

Questa sezione include alcuni esempi di casi d'uso tipici per il controllo accessi.

Argomenti

Permettere a Account AWS di accedere a un argomento

Supponiamo che tu abbia un argomento nel sistema Amazon SNS. Nel caso più semplice, vuoi consentire a uno o più Account AWS di accedere a un'azione specifica dell'argomento (ad esempio, Pubblica).

Puoi farlo usando l'operazione API di Amazon SNS `AddPermission` Questa richiede un argomento, un elenco di ID Account AWS, un elenco di operazioni e un'etichetta e crea automaticamente una nuova dichiarazione nella policy di controllo accessi dell'argomento. In questo caso, non sei tu a scrivere una policy perché è Amazon SNS che genera automaticamente la nuova dichiarazione di policy per te. Puoi rimuovere la dichiarazione di policy in seguito chiamando `RemovePermission` con la propria etichetta.

Ad esempio, se `AddPermission` chiami l'argomento arn:aws:sns:us-east-2:444455556666:MyTopic, con Account AWS ID 1111-2222-3333, l'`Publish`azione e l'etichetta, Amazon SNS genererà e inserirà la seguente dichiarazione sulla politica di controllo degli `grant-1234-publish` accessi:

```
{
```

```
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Una volta aggiunta questa dichiarazione, l'utente con Account AWS 1111-2222-3333 può pubblicare messaggi nell'argomento.

Limitazione delle sottoscrizioni a HTTPS

Nel seguente esempio, il protocollo di distribuzione delle notifiche viene limitato a HTTPS.

Hai bisogno di sapere come scrivere la tua policy per l'argomento perché l'operazione di Amazon SNS `AddPermission` non consente di specificare una restrizione del protocollo quando si concede a qualcuno l'accesso al proprio argomento. In questo caso, scrivi la tua policy e poi usi l'operazione `SetTopicAttributes` per impostare l'attributo `Policy` dell'argomento sulla tua nuova policy.

L'esempio seguente di una policy completa fornisce all'ID Account AWS 1111-2222-3333 la possibilità di sottoscriversi alle notifiche da un argomento.

```
{
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
      }
    }
  }]
}
```

Pubblicare messaggi in una coda Amazon SQS.

In questo caso d'uso, è possibile pubblicare i messaggi dal tuo argomento nella tua coda Amazon SQS. Come Amazon SNS, Amazon SQS; utilizza il linguaggio della policy di controllo accessi di Amazon. Per consentire a Amazon SNS; di inviare messaggi, dovrai usare l'operazione Amazon SQS; `SetQueueAttributes` per impostare una policy sulla coda.

Ancora una volta, avrai bisogno di sapere come scrivere la tua policy perché l'operazione `AddPermission` di Amazon SQS non crea dichiarazioni di policy con condizioni.

> **Note**
>
> L'esempio presentato di seguito è una policy Amazon SQS che controlla l'accesso alla tua coda e non una policy Amazon SNS che controlla l'accesso al tuo argomento. Queste sono quindi operazioni Amazon SQS e la risorsa è l'Amazon Resource Name (ARN) della coda. Puoi determinare l'ARN della coda recuperando l'attributo `QueueArn` della coda con l'operazione `GetQueueAttributes`.

```
{
  "Statement": [{
    "Sid": "Allow-SNS-SendMessage",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": ["sqs:SendMessage"],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:MyQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:444455556666:MyTopic"
      }
    }
  }]
}
```

Questa policy usa la condizione `aws:SourceArn` per limitare l'accesso alla coda in base all'origine del messaggio inviato alla coda. Puoi usare questo tipo di policy per consentire a Amazon SNS di inviare messaggi alla tua coda solo se i messaggi provengono da uno dei tuoi argomenti. In

questo caso, specifichi uno dei tuoi argomenti in particolare, il cui ARN è arn:aws:sns:us-east-2:444455556666:. MyTopic

La precedente policy è un esempio di policy Amazon SQS che è possibile scrivere e aggiungere a una coda specifica. Ciò consentirebbe l'accesso ad Amazon SNS e ad altri servizi AWS. Amazon SNS; fornisce una policy predefinita a tutti gli argomenti appena creati. La policy predefinita consente l'accesso all'argomento a tutti gli altri servizi AWS. Questa policy predefinita usa una condizione `aws:SourceArn` per garantire che i servizi AWS accedano al tuo argomento solo per conto delle risorse AWS in tuo possesso.

Consentire la pubblicazione delle notifiche degli eventi Amazon S3 su un argomento

In questo caso, bisogna configurare la policy di un argomento in modo che il bucket Amazon S3 di un altro Account AWS possa pubblicare nel proprio argomento. Per ulteriori informazioni sulla pubblicazione di notifiche da Amazon S3, vai a Impostazione delle notifiche degli eventi bucket.

In questo esempio si presume che tu scriva la tua policy e poi usi l'operazione `SetTopicAttributes` per impostare l'attributo `Policy` dell'argomento sulla tua nuova policy.

Nell'istruzione di esempio seguente viene utilizzata la condizione `SourceAccount` per garantire che solo il proprietario dell'account Amazon S3 possa accedere all'argomento. In questo esempio, il proprietario dell'argomento è 111122223333 e il proprietario Amazon S3 è 444455556666. L'esempio indica che qualsiasi bucket Amazon S3 di proprietà di 444455556666 può pubblicare su. MyTopic

```
{
  "Statement": [{
    "Effect": "Allow",
     "Principal": {
      "Service": "s3.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      }
    }
  }]
}
```

Quando si pubblicano eventi su Amazon SNS, i seguenti servizi supportano`aws:SourceAccount`:

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon ElastiCache
- Amazon GameLift
- Servizio di SMS e messaggi vocali Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

Consenti ad Amazon SES di pubblicare in un argomento di proprietà di un altro account

È possibile consentire a un altro servizio AWS di pubblicare su un argomento di proprietà di un altro Account AWS. Supponiamo di aver effettuato l'accesso dall'account 111122223333, aperto Amazon SES e creato un'e-mail. Per pubblicare notifiche relative a questa e-mail in un argomento Amazon SNS proprietario dell'account 444455556666, bisogna creare una policy come la seguente. A tale scopo, è necessario fornire informazioni sull'entità principale (l'altro servizio) e sulla proprietà di ciascuna risorsa. La dichiarazione `Resource` fornisce l'argomento ARN, che include l'account ID del proprietario dell'argomento, 444455556666. La dichiarazione `"aws:SourceOwner": "111122223333"` specifica che il tuo account è proprietario dell'e-mail.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
  ]
}
```

Quando si pubblicano eventi su Amazon SNS, i seguenti servizi supportano`aws:SourceOwner`:

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon ElastiCache
- Amazon GameLift
- Servizio di SMS e messaggi vocali Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` rispetto a **`aws:SourceOwner`**

> **Important**
>
> `aws:SourceOwner` è obsoleto e i nuovi servizi possono integrarsi con Amazon SNS solo tramite e `aws:SourceArn` e `aws:SourceAccount`. Amazon SNS mantiene ancora la

compatibilità con le versioni precedenti per i servizi esistenti che supportano attualmente `aws:SourceOwner`.

Le chiavi di condizione `aws:SourceAccount` e `aws:SourceOwner` sono impostate ciascuna da alcuni Servizi AWS quando pubblicano in un argomento Amazon SNS. Se supportato, il valore dell'account ID AWS sarà di 12 cifre per conto del quale il servizio sta pubblicando i dati. Alcuni servizi supportano uno e altri supportano l'altro.

- Consulta Consentire la pubblicazione delle notifiche degli eventi Amazon S3 su un argomento per sapere come le notifiche di Amazon S3 utilizzano `aws:SourceAccount` ed un elenco di servizi AWS che supportano tale condizione.
- Consulta Consenti ad Amazon SES di pubblicare in un argomento di proprietà di un altro account per sapere come le notifiche di Amazon SES utilizzano `aws:SourceOwner` ed un elenco di servizi AWS che supportano tale condizione.

Consenti agli account di un'organizzazione AWS Organizations di pubblicare in un argomento in un account diverso

Il servizio AWS Organizations ti aiuta a gestire centralmente la fatturazione, controllare l'accesso e la sicurezza e condividere le risorse tra i tuoi Account AWS.

Puoi trovare l'ID organizzazione nella console Organizations. Per ulteriori informazioni, consulta Visualizzazione dei dettagli di un'organizzazione dall'account master.

In questo esempio, qualsiasi Account AWS nell'organizzazione `myOrgId` può pubblicare in un argomento `MyTopic` Amazon SNS nell'account `444455556666`. La policy controlla il valore dell'ID organizzazione utilizzando la chiave di condizione globale `aws:PrincipalOrgID`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

Consenti la pubblicazione di qualsiasi CloudWatch avviso su un argomento in un altro account

In questo caso, tutti gli CloudWatch allarmi 111122223333 presenti nell'account possono essere pubblicati su un argomento di Amazon SNS dell'account. 444455556666

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

Limitare la pubblicazione a un argomento Amazon SNS solo da un endpoint VPC specifico

In questo caso, l'argomento nell'account 444455556666 può pubblicare solo dall'endpoint VPC con l'ID vpce-1ab2c34d.

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "SNS:Publish",
```

```
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

Come funziona Amazon Simple Notification Service

Prima di utilizzare IAM per gestire l'accesso ad Amazon SNS, scopri quali funzioni IAM sono disponibili per l'uso con Amazon SNS.

Funzionalità di IAM utilizzabili con Amazon Simple Notification Service

Funzionalità IAM	Supporto per Amazon SNS
Policy basate su identità	Sì
Policy basate su risorse	Sì
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
Liste di controllo degli accessi (ACL)	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per ottenere un quadro generale del funzionamento di Amazon SNS e altri servizi AWS con la maggior parte delle caratteristiche di IAM, consulta Servizi AWS supportati da IAM nella Guida per l'utente di IAM.

Operazioni delle policy per Amazon SNS

Supporta le azioni di policy	Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a una policy. Le azioni di policy hanno spesso lo stesso nome dell'operazione API AWS. Ci sono alcune eccezioni, ad esempio le azioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di operazioni di Amazon SNS, consulta Risorse definite da Amazon SNS nella documentazione di riferimento all'autorizzazione del servizio.

Le operazioni delle policy in Amazon SNS utilizzano il seguente prefisso prima dell'operazione:

```
sns
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta Esempi di policy basate su identità per Amazon Simple Notification Service.

Risorse delle policy per Amazon SNS

Supporta le risorse di policy	Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'azione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco di tipi di risorse di Amazon SNS, consulta Operazioni definite da Amazon SNS nella documentazione di riferimento all'autorizzazione del servizio. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ciascuna risorsa, consulta Risorse definite da Amazon SNS.

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta Esempi di policy basate su identità per Amazon Simple Notification Service.

Chiavi di condizione delle policy per Amazon SNS

Supporta le chiavi di condizione delle policy specifiche del servizio	Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. Puoi compilare espressioni

condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione `OR` logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche per il servizio. Per visualizzare tutte le chiavi di condizione globali di AWS, consulta Chiavi di contesto delle condizioni globali di AWS nella Guida per l'utente di IAM.

Per un elenco completo delle chiavi di condizione di Amazon SNS, consulta Chiavi di condizione per Amazon SNS nella documentazione di riferimento all'autorizzazione di servizio. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consulta Risorse definite da Amazon SNS.

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta Esempi di policy basate su identità per Amazon Simple Notification Service.

ACL in Amazon SNS

Supporta le ACL	No

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Amazon SNS

Supporta ABAC (tag nelle policy)	Parziale

Il controllo dell'accesso basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, tali attributi sono denominati tag. È possibile collegare dei tag alle entità IAM (utenti o ruoli) e a numerose risorse AWS. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni su ABAC, consulta Che cos'è ABAC? nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Amazon SNS

Supporta le credenziali temporanee	Sì

Alcuni Servizi AWS non funzionano quando si accede utilizzando credenziali temporanee. Per ulteriori informazioni, inclusi i Servizi AWS che funzionano con le credenziali temporanee, consulta Servizi AWS supportati da IAM nella Guida per l'utente IAM.

Le credenziali temporanee sono utilizzate se si accede alla AWS Management Console utilizzando qualsiasi metodo che non sia la combinazione di nome utente e password. Ad esempio, quando accedi ad AWS utilizzando il collegamento Single Sign-On (SSO) della tua azienda, tale processo crea in automatico credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta Cambio di un ruolo (console) nella Guida per l'utente di IAM.

È possibile creare manualmente credenziali temporanee utilizzando la AWS CLIo l'API AWS. È quindi possibile utilizzare tali credenziali temporanee per accedere ad AWS. AWSconsiglia di generare le

credenziali temporanee dinamicamente anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza provvisorie in IAM.

Autorizzazioni dei principali tra servizi per Amazon SNS

Supporta sessioni di accesso diretto (FAS)	Sì

Quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, si viene considerati un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'azione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

Ruoli di servizio per Amazon SNS

Supporta i ruoli di servizio	Sì

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 Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe compromettere la funzionalità di Amazon SNS. Modifica i ruoli del servizio solo quando Amazon SNS fornisce le indicazioni per farlo.

Ruoli collegati ai servizi per Amazon SNS

Supporta i ruoli collegati ai servizi	No

Un ruolo collegato ai servizi è un tipo di ruolo di servizio che è collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'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.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate su identità per Amazon Simple Notification Service

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse di Amazon SNS. Inoltre, non sono in grado di eseguire attività utilizzando la AWS Management Console, l'AWS Command Line Interface (AWS CLI) o l'API AWS. Per concedere agli utenti l'autorizzazione per 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.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Per informazioni dettagliate sulle operazioni e sui tipi di risorse definiti da Amazon SNS, incluso il formato degli ARN per ogni tipo di risorsa, consulta Operazioni, risorse e chiavi di condizione per Amazon SNS nella documentazione di riferimento all'autorizzazione di servizio.

Argomenti

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon SNS nell'account. Queste operazioni possono comportare costi aggiuntivi per l'Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Nozioni di base sulle policy gestite da AWSe passaggio alle autorizzazioni con privilegio minimo: per le informazioni di base su come concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy gestite da AWSche concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo Account AWS. Ti consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente di AWSspecifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente IAM.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi inoltre utilizzare le condizioni per concedere l'accesso alle operazioni di servizio, ma solo se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente di IAM.
- Richiesta dell'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o utenti root nel tuo Account AWS, attiva MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Utilizzo della console Amazon SNS

Per accedere alla console Amazon SNS, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentire di elencare e visualizzare i dettagli relativi alle risorse Amazon SNS nel tuo Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non è necessario concedere le autorizzazioni minime della console agli utenti che effettuano chiamate solo alla AWS CLI o all'API AWS. Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che gli utenti e i ruoli possano continuare a utilizzare la console Amazon SNS, collega anche la policy `ConsoleAccess` o `ReadOnly` gestita da AWS di Amazon SNS alle entità. Per ulteriori informazioni, consulta Aggiunta di autorizzazioni a un utente nella Guida per l'utente IAM.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizationsè un servizio per il raggruppamento e la gestione centralizzata degli Account AWSmultipli di proprietà dell'azienda. Se si abilitano tutte le caratteristiche in un'organizzazione, è possibile applicare le policy di controllo dei servizi (SCP) a uno o tutti i propri account. Una SCP limita le autorizzazioni per le entità negli account membri, compreso ogni utente root Account AWS. Per ulteriori informazioni su Organizations e le policy SCP, consulta Utilizzo delle SCP nella Guida per l'utente di AWS Organizations.

- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella Guida per l'utente di IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono allegate alla relativa identità utente. La policy include le autorizzazioni per completare questa azione sulla console o a livello di programmazione utilizzando la AWS CLIo l'API AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
```

```
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy basate su identità per Amazon SNS

Supporta le policy basate su identità	Sì

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 azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Esempi di policy basate su identità per Amazon SNS

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta Esempi di policy basate su identità per Amazon Simple Notification Service.

Policy basate su risorse all'interno di Amazon SNS

Supporta le policy basate su risorse	Sì

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando l'entità principale e la risorsa si trovano in diversi Account AWS, un amministratore IAM nell'account attendibile deve concedere all'entità principale (utente o ruolo) anche l'autorizzazione per accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Utilizzo di policy basate su identità con Amazon SNS

Argomenti

- Utilizzo congiunto di policy IAM e Amazon SNS;
- Formato ARN della risorsa Amazon SNS
- Operazioni dell’API Amazon SNS
- Chiavi di policy Amazon SNS
- Esempi di policy per Amazon SNS

Amazon Simple Notification Service si integra con (IAM) AWS Identity and Access Management di modo che sia possibile specificare quali operazioni Amazon SNS un utente nel tuoAccount AWS può eseguire con le risorse Amazon SNS. Puoi specificare un particolare argomento nella policy. Ad esempio, puoi utilizzare delle variabili durante la creazione di una policy IAM che autorizza certi utenti della tua organizzazione a utilizzare l'operazione `Publish` con specifici argomenti nel tuo Account AWS. Per ulteriori informazioni, consulta la pagina sulle Variabili delle policy nella guida Uso di IAM.

> **Important**
>
> L'utilizzo di Amazon SNS con IAM; non comporta modifiche nell'utilizzo di Amazon SNS. Le operazioni di Amazon SNS esistenti rimangono immutate e non vengono aggiunte nuove operazioni di Amazon SNS relative agli utenti e al controllo accessi.

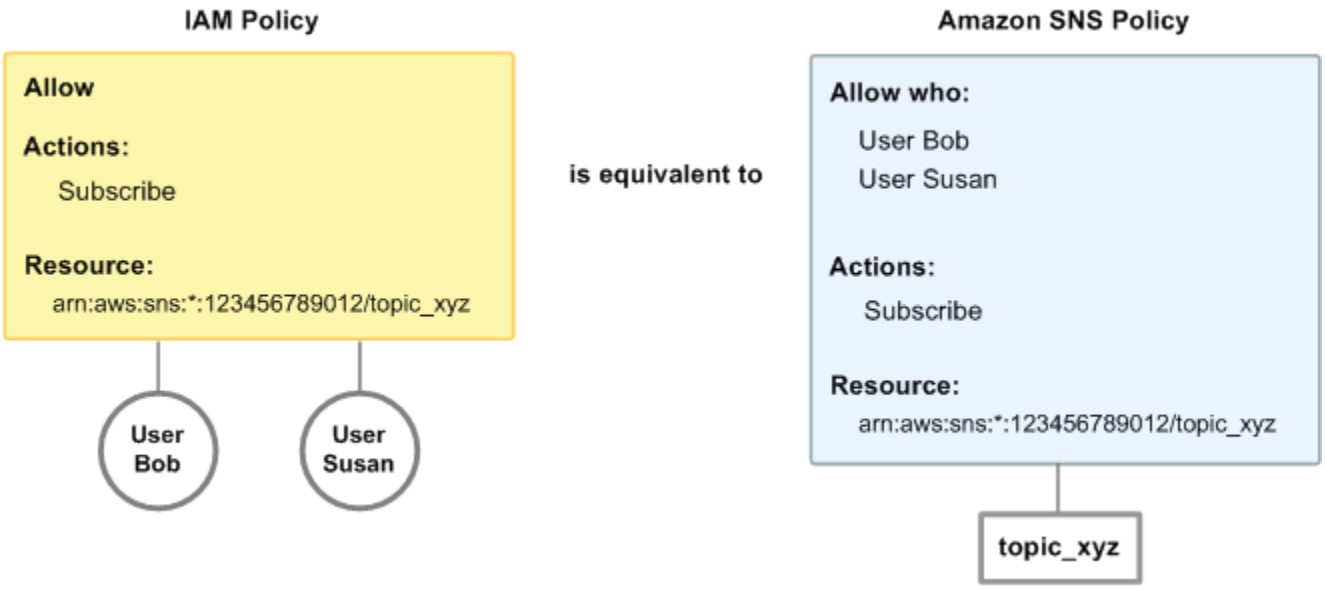
Per esempi di policy che coprono le operazioni e le risorse di Amazon SNS, consulta Esempi di policy per Amazon SNS.

Utilizzo congiunto di policy IAM e Amazon SNS;

Una policy IAM; consente di limitare l'accesso degli utenti ad operazioni e argomenti di Amazon SNS. Una policy IAM può limitare l'accesso solo agli utenti del tuo account AWS e non a quelli di altri Account AWS.

Puoi utilizzare una policy Amazon SNS con un particolare argomento per stabilire quali utenti sono autorizzati a utilizzare quell'argomento (ad esempio, chi può pubblicare messaggi nell'argomento, chi può eseguire una sottoscrizione, ecc.). Le policy di Amazon SNS possono concedere l'accesso ad altri Account AWS, o agli utenti all'interno del proprio Account AWS.

Per autorizzare gli utenti ad accedere ai tuoi argomenti Amazon SNS, puoi utilizzare le policy IAM, le policy Amazon SNS o entrambe. Nella maggior parte dei casi, puoi ottenere gli stessi risultati con le une o le altre. Ad esempio, il diagramma seguente mostra l'equivalenza tra una policy IAM e una policy Amazon SNS. La policy IAM autorizza l'operazione `Subscribe` di Amazon SNS per l'argomento denominato topic_xyz nel tuo account AWS. La policy IAM è associata agli utenti Bob e Susan (il che significa che Bob e Susan dispongono delle autorizzazioni definite nella policy). Anche la policy Amazon SNS fornisce a Bob e Susan l'autorizzazione per accedere all'operazione `Subscribe` per l'argomento topic_xyz.



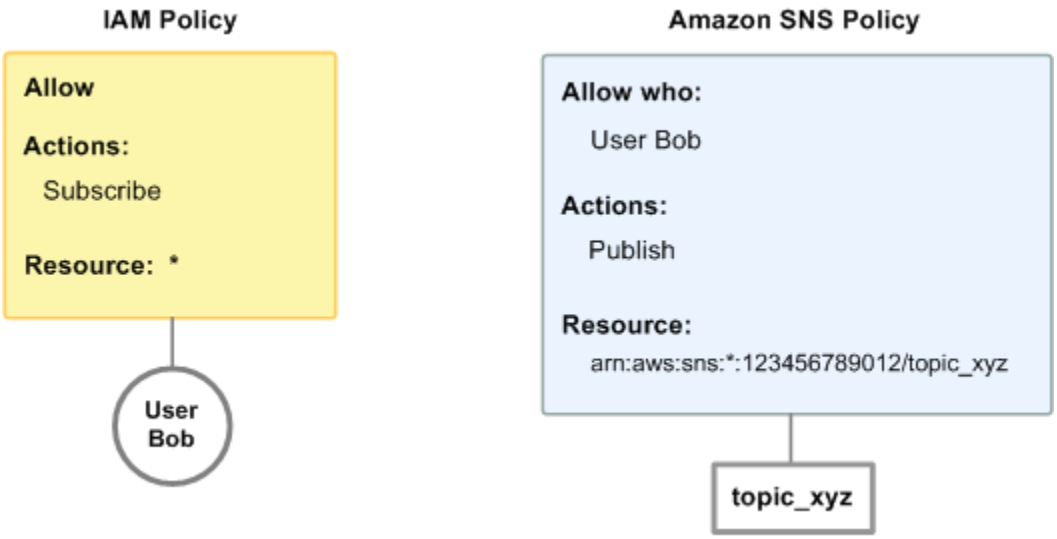
> **Note**
>
> L'esempio precedente mostra policy semplici senza condizioni. Puoi specificare una particolare condizione in una qualsiasi delle policy e ottenere lo stesso risultato.

Esiste tuttavia una differenza tra le policy Amazon SNS e IAM di AWS: il sistema di policy Amazon SNS, contrariamente alle policy IAM, consente di concedere l'autorizzazione ad altri Account AWS.

Sta a te decidere se utilizzare insieme i sistemi per gestire le autorizzazioni, in base alle tue esigenze. Gli esempi seguenti mostrano il modo in cui i due sistemi di policy interagiscono.

Example 1

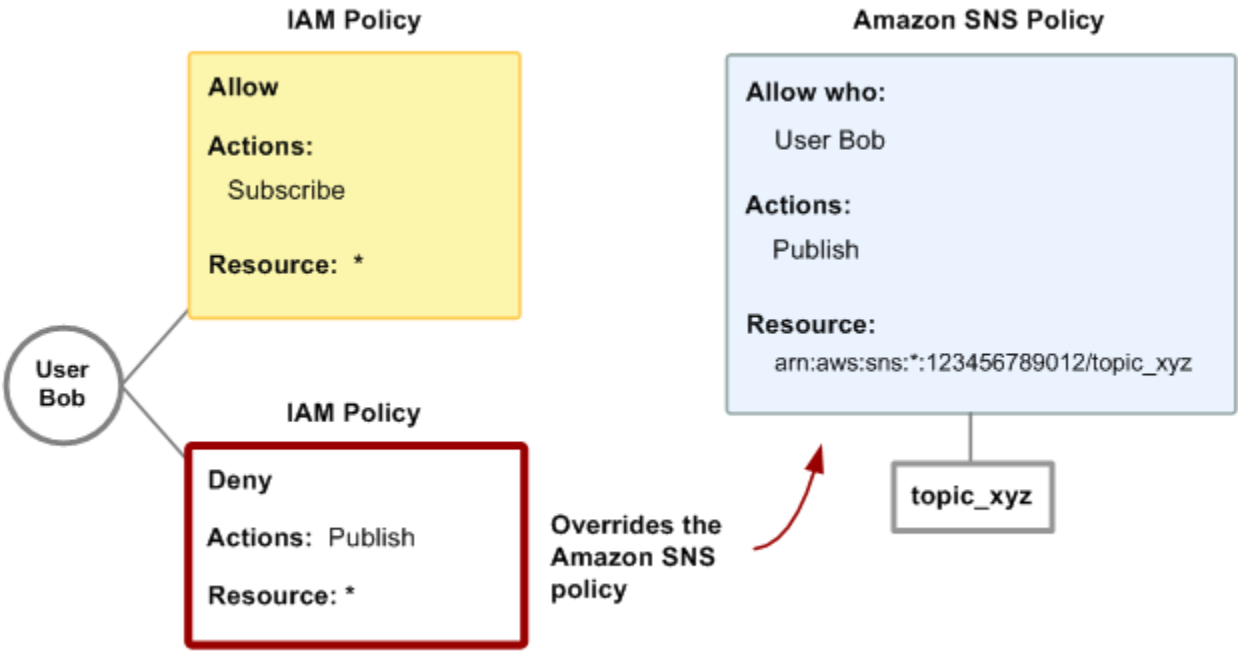
In questo esempio, una policy IAM e una policy Amazon SNS si applicano entrambe a Bob. La policy IAM lo autorizza a utilizzare `Subscribe` per qualsiasi argomento Account AWS, mentre la policy Amazon SNS gli concede l'autorizzazione a utilizzare `Publish` per un argomento specifico (topic_xyz). Il diagramma seguente illustra questo concetto.



Se Bob dovesse inviare una richiesta di sottoscrizione a qualsiasi argomento nell'account AWS, la policy IAM autorizzerebbe l'operazione. Se Bob dovesse inviare una richiesta di pubblicazione di un messaggio in topic_xyz, la policy Amazon SNS autorizzerebbe l'operazione.

Example 2

In questo esempio, facciamo riferimento all'esempio 1 (dove due policy si applicano a Bob). Supponiamo che Bob pubblichi dei messaggi in topic_xyz quando non dovrebbe farlo e che di conseguenza tu intenda negargli completamente la possibilità di pubblicare negli argomenti. La cosa più semplice da fare è aggiungere una policy IAM che gli nega l'accesso all'operazione `Publish` per tutti gli argomenti. Questa terza policy sostituisce la policy Amazon SNS che lo autorizzava inizialmente a pubblicare nell'argomento topic_xyz, in quanto un rifiuto esplicito sovrascrive sempre un'autorizzazione (per ulteriori informazioni sulla logica di valutazione delle policy, consulta Logica della valutazione). Il diagramma seguente illustra questo concetto.



Per esempi di policy che coprono le operazioni e le risorse di Amazon SNS, consulta Esempi di policy per Amazon SNS. Per ulteriori informazioni sulla scrittura di policy Amazon SNS, consulta la documentazione tecnica per Amazon SNS;.

Formato ARN della risorsa Amazon SNS

Per Amazon SNS, gli argomenti sono il solo tipo di risorsa che è possibile specificare in una policy. Di seguito viene riportato il formato Amazon Resource Name (ARN) per gli argomenti.

```
arn:aws:sns:region:account_ID:topic_name
```

Per ulteriori informazioni sugli ARN, consulta la pagina ARN nella Guida Utente IAM.

Example

Quello che segue è un ARN per un argomento denominato MyTopic nella regione us-east-2, appartenente a 123456789012. Account AWS

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Example

Se disponevi di un argomento denominato MyTopic in ciascuna delle diverse regioni supportate da Amazon SNS, puoi specificare l'argomento con il seguente ARN.

```
arn:aws:sns:*:123456789012:MyTopic
```

Puoi utilizzare i caratteri jolly * e ? nel nome dell'argomento. Ad esempio, quanto segue potrebbe fare riferimento a tutti gli argomenti creati da Bob ai quali ha aggiunto il prefisso bob_.

```
arn:aws:sns:*:123456789012:bob_*
```

Per convenienza, quando crei un argomento, Amazon SNS restituisce l'ARN dell'argomento nella risposta.

Operazioni dell'API Amazon SNS

In una policy IAM, si può specificare qualsiasi operazione che Amazon SNS offre. Tuttavia, le operazioni `ConfirmSubscription` e `Unsubscribe` non richiedono l'autenticazione, il che significa che anche se specifichi quelle operazioni in una policy, IAM; non limiterà l'accesso degli utenti alle stesse.

Ogni operazione che specifichi in una policy deve essere preceduta dalla stringa in minuscolo `sns:`. Per esempio, per specificare tutte le operazioni di Amazon SNS, bisogna utilizzare `sns:*`. Per un elenco delle operazioni, consulta laDocumentazione di riferimento API di Amazon Simple Notification Service.

Chiavi di policy Amazon SNS

Amazon SNS implementa le seguenti chiavi di policy AWS, più alcune chiavi specifiche dei servizi.

Per un elenco di chiavi di condizione supportato da ogni servizio Servizio AWS, vedere Operazioni, risorse e chiavi di condizione per servizi Servizi AWS nella Guida per l'utente IAM. Per un elenco di chiavi di condizione che possono essere utilizzate in più servizi Servizi AWS, vedere Chiavi di contesto delle condizioni globali AWS nella Guida per l'utente IAM.

Amazon SNS utilizza le chiavi specifiche dei servizi descritte di seguito. Utilizza queste chiavi nelle policy che limitano l'accesso alle richieste `Subscribe`.

- sns:endpoint–URL, indirizzo e-mail o ARN di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza questa chiave con le condizioni di stringa (vedi Esempi di policy per Amazon SNS) per limitare l'accesso a specifici endpoint (ad esempio, *@yourcompany.com).
- sns:protocol– il valore `protocol` di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza questa chiave con le condizioni di stringa (vedi Esempi di policy per Amazon SNS) per limitare la pubblicazione a specifici protocolli di consegna (ad esempio, https).

Esempi di policy per Amazon SNS

In questa sezione vengono illustrati varie semplici policy per il controllo dell'accesso degli utenti a Amazon SNS.

> **Note**
>
> In futuro, è possibile che Amazon SNS aggiunga nuove operazioni che dovrebbero essere logicamente incluse in una delle seguenti policy, a seconda degli obiettivi definiti della policy.

Example Esempio 1: autorizzare un gruppo a creare e gestire argomenti

In questo esempio, creiamo una policy che autorizza l'accesso a `CreateTopic`, `ListTopics`, `SetTopicAttributes` e `DeleteTopic`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example Esempio 2: autorizzare il reparto IT a pubblicare messaggi in un particolare argomento

In questo esempio, creiamo un gruppo per il reparto IT e assegniamo una policy che autorizza l'accesso a `Publish` per l'argomento desiderato.

```
{
  "Statement": [{
```

```
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example Esempio 3: autorizzare gli utenti Account AWS a effettuare la sottoscrizione a degli argomenti

In questo esempio, creiamo una policy che consente di accedere all'operazione `Subscribe`, con condizioni di corrispondenza di stringa per le chiavi di policy `sns:Protocol` e `sns:Endpoint`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example Esempio 4: autorizzare un partner a pubblicare messaggi in un particolare argomento

Puoi utilizzare una policy Amazon SNS; o IAM; per consentire a un partner di pubblicare in uno specifico argomento. Se il partner dispone di un Account AWS, potrebbe risultare più facile utilizzare una policy Amazon SNS. Tuttavia, qualsiasi persona nell'azienda del partner che dispone delle credenziali di sicurezza AWS potrebbe pubblicare messaggi nell'argomento. Questo esempio presuppone che tu voglia limitare l'accesso a una determinata persona (o applicazione). A questo proposito, devi considerare il partner come un utente nella tua azienda e utilizzare una policy IAM; anziché una policy Amazon SNS;.

Per questo esempio, creiamo un gruppo chiamato WidgetCo che rappresenta l'azienda partner; creiamo un utente per la persona (o applicazione) specifica presso l'azienda partner che ha bisogno di accedere; quindi inseriamo l'utente nel gruppo.

Quindi alleghiamo una politica che concede al gruppo l'`Publish`accesso all'argomento specifico denominato WidgetPartnerTopic.

Desideriamo inoltre impedire al WidgetCo gruppo di fare altro con gli argomenti, quindi aggiungiamo una dichiarazione che nega l'autorizzazione a qualsiasi azione di Amazon SNS `Publish` diversa da quella su argomenti diversi. WidgetPartnerTopic Ciò è necessario solo se è presente una policy generica altrove nel sistema che fornisce agli utenti ampio accesso a Amazon SNS.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Utilizzo delle credenziali di sicurezza temporanee con Amazon SNS

Oltre a creare utenti IAM con credenziali di sicurezza personali, IAM consente inoltre di concedere credenziali di sicurezza temporanee a qualsiasi utente, autorizzandone quindi l'accesso ai tuoi servizi e risorse AWS. Puoi gestire utenti che hanno Account AWS, denominati utenti IAM, nonché gestire utenti per il tuo sistema che non hanno Account AWS, denominati utenti federati. Inoltre, gli "utenti" possono anche essere applicazioni che hai creato per accedere alle risorse AWS.

È possibile utilizzare queste credenziali di sicurezza temporanee quando si effettuano richieste ad Amazon SNS. Le librerie API calcolano il valore di firma necessario utilizzando tali credenziali per autenticare la richiesta dell'utente. In caso di invio di richieste tramite l'utilizzo di credenziali scadute, Amazon SNS rifiuta la richiesta.

Per ulteriori informazioni sul supporto di IAM per le credenziali di sicurezza temporanee, vai a Concessione dell'accesso temporaneo alle risorse AWS in Utilizzo di IAM.

Example Utilizzo delle credenziali di sicurezza temporanee per autenticare una richiesta Amazon SNS

Nell'esempio seguente viene mostrato come ottenere credenziali di sicurezza temporanee per autenticare una richiesta Amazon SNS.

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Autorizzazioni API Amazon SNS: riferimento a operazioni e risorse

Il seguente elenco fornisce informazioni specifiche all'implementazione del controllo accessi di Amazon SNS:

- Ogni policy deve coprire un solo argomento (durante la scrittura di una policy, non includere dichiarazioni che coprono differenti argomenti)
- Ogni policy deve avere una policy `Id` univoca
- Ogni dichiarazione in una policy deve avere una dichiarazione `sid` univoca

Quote delle policy

La tabella seguente elenca le quote massime per una dichiarazione della policy.

Nome	Quota massima
Byte	30 KB
Dichiarazioni	100
Principali	Da 1 a 200 (0 non è valido).

Nome	Quota massima
Risorsa	1 (0 è nullo. Il valore deve corrispondere all'ARN dell'argomento della policy).

Operazioni di policy Amazon SNS valide

Amazon SNS supporta le operazioni elencate nella tabella seguente.

Azione	Descrizione
sns: AddPermission	Autorizza l'aggiunta di autorizzazioni alla policy dell'argomento.
sms: DeleteTopic	Autorizza l'eliminazione di un argomento.
sms: GetDataProtectionPolicy	Concede l'autorizzazione per recuperare la policy di protezione dei dati dell'argomento
sms: GetTopicAttributes	Autorizza la ricezione di tutti gli attributi di argomento.
sms: ListSubscriptionsByTopic	Autorizza il recupero di tutte le sottoscrizioni a uno specifico argomento.
sms: ListTagsForResource	Concede l'autorizzazione per elencare tutti i tag aggiunti a un argomento specifico.
sns:Publish	Autorizza sia la pubblicazione che la pubblicazione batch in un argomento o in un endpoint. Per ulteriori informazioni, consulta Publish and PublishBatchin Amazon Simple Notification Service API Reference.
sns: PutDataProtectionPolicy	Concede l'autorizzazione per impostare la policy di protezione dei dati dell'argomento
sms: RemovePermission	Autorizza la rimozione di tutte le autorizzazioni nella policy dell'argomento.
sms: SetTopicAttributes	Autorizza l'impostazione degli attributi di un argomento.

Azione	Descrizione
sns:Subscribe	Autorizza la sottoscrizione a un argomento.

Chiavi specifiche del servizio

Amazon SNS utilizza le chiavi specifiche ai servizi descritte di seguito. Puoi utilizzare queste chiavi nelle policy che limitano l'accesso alle richieste `Subscribe`.

- sns:endpoint–URL, indirizzo e-mail o ARN di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza con le condizioni di stringa (vedi Esempi di policy per Amazon SNS) per limitare l'accesso a specifici endpoint (ad esempio, *@example.com).
- sns:protocol– il valore `protocol` di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza con le condizioni di stringa (vedi Esempi di policy per Amazon SNS) per limitare la pubblicazione a specifici protocolli di consegna (ad esempio, https).

> **Important**
>
> Quando utilizzi una policy per il controllo accessi mediante sns:Endpoint, tieni presente che i problemi DNS potrebbero alterare la risoluzione dei nomi dell'endpoint in futuro.

Risoluzione dei problemi di identità e accesso ad Amazon Simple Notification Service

Utilizza le informazioni seguenti per eseguire la diagnosi e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Amazon SNS.

Argomenti

- Non dispongo dell'autorizzazione per eseguire un'operazione in Amazon SNS
- Non sono autorizzato a eseguire iam: PassRole
- Desidero consentire alle persone esterne al mio Account AWS di accedere alle mie risorse Amazon SNS

Non dispongo dell'autorizzazione per eseguire un'operazione in Amazon SNS

Se ricevi un errore che indica che non disponi dell'autorizzazione per eseguire un'operazione, le tue policy devono essere aggiornate in modo che ti sei consentito eseguire tale operazione.

Il seguente esempio di errore si verifica quando l'utente `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia, ma non dispone di autorizzazioni `sns:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

In questo caso, la policy deve essere aggiornata in modo che Mateo possa accedere alla risorsa *`my-example-widget`* mediante l'operazione `sns:`*`GetWidget`*.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. 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 disponi dell'autorizzazione per eseguire l'operazione `iam:PassRole`, per poter passare un ruolo ad Amazon SNS dovrai aggiornare le policy.

Alcuni Servizi AWSconsentono di trasmettere un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente esempio di errore si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon SNS. Tuttavia, l'azione 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`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Desidero consentire alle persone esterne al mio Account AWS di accedere alle mie risorse Amazon SNS

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon SNS supporta queste funzionalità, consulta Come funziona Amazon Simple Notification Service.
- Per informazioni su come garantire l'accesso alle risorse negli Account AWSche possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS in tuo possesso nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso alle risorse ad Account AWS di terze parti, consulta Fornire l'accesso agli Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Registrazione e monitoraggio in Amazon SNS

Questa sezione fornisce informazioni relative agli argomenti di registrazione e monitoraggio di Amazon SNS.

Argomenti

- Registrazione delle chiamate API Amazon SNS utilizzando CloudTrail
- Monitoraggio di argomenti Amazon SNS tramite CloudWatch

Registrazione delle chiamate API Amazon SNS utilizzando CloudTrail

Amazon SNS è integrato con AWS CloudTrail, un servizio che offre un record delle operazioni eseguite da un utente, un ruolo o un servizio AWS in Amazon SNS. CloudTrail acquisisce le chiamate API per Amazon SNS come eventi. Le chiamate acquisite includono le chiamate dalla console Amazon SNS e le chiamate di codice alle operazioni API di Amazon SNS. Se crei un trail, puoi abilitare la distribuzione continua di eventi CloudTrail in un bucket Amazon S3, inclusi gli eventi per Amazon SNS. Se non si configura un trail, è comunque possibile visualizzare gli eventi più recenti nella console di CloudTrail in Event history (Cronologia eventi). Le informazioni raccolte da CloudTrail consentono di determinare la richiesta effettuata ad Amazon SNS, l'indirizzo IP da cui è partita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli.

Per ulteriori informazioni su CloudTrail, incluso come configurarlo e abilitarlo, consultare la Guida per l'utente di AWS CloudTrail.

Informazioni su Amazon SNS in CloudTrail

CloudTrail è abilitato sul tuo Account AWS al momento della sua creazione. Quando si verifica un'attività supportata in Amazon SNS, questa viene registrata in un evento CloudTrail insieme ad altri eventi di servizio AWS in Cronologia eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nell'Account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi mediante la cronologia eventi di CloudTrail.

Per una registrazione continua degli eventi nell'account Account AWS, inclusi gli eventi per Amazon SNS, crea un trail. Un trail consente a CloudTrail di distribuire i file di log in un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di log nel bucket Amazon S3 specificato. Inoltre, è possibile configurare altri servizi AWS per analizzare con maggiore dettaglio e usare i dati evento raccolti nei log CloudTrail. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un percorso
- Servizi e integrazioni CloudTrail supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di log CloudTrail da più regioni e Ricezione di file di log CloudTrail da più account

Eventi del piano di controllo in CloudTrail

Amazon SNS supporta la registrazione delle operazioni seguenti come eventi nei file di log CloudTrail:

- AddPermission
- CheckIfPhoneNumberIsOptedOut
- ConfirmSubscription
- CreatePlatformApplication
- CreatePlatformEndpoint
- CreateSMSSandboxPhoneNumber
- CreateTopic
- DeleteEndpoint
- DeletePlatformApplication
- DeleteSMSSandboxPhoneNumber
- DeleteTopic
- GetDataProtectionPolicy
- GetEndpointAttributes
- GetPlatformApplicationAttributes
- GetSMSAttributes
- GetSMSSandboxAccountStatus
- GetSubscriptionAttributes
- GetTopicAttributes
- ListEndpointsByPlatformApplication
- ListOriginationNumbers
- ListPhoneNumbersOptedOut
- ListPlatformApplications
- ListSMSSandboxPhoneNumbers
- ListSubscriptions
- ListSubscriptionsByTopic
- ListTagsForResource

- ListTopics
- OptInPhoneNumber
- PutDataProtectionPolicy
- RemovePermission
- SetEndpointAttributes
- SetPlatformApplicationAttributes
- SetSMSAttributes
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource
- VerifySMSSandboxPhoneNumber

> **Note**
>
> Quando non hai effettuato l'accesso a Amazon Web Services (modalità non autenticata) e viene chiamata l'operazione ConfirmSubscription o Unsubscribe, l'accesso a CloudTrail non viene eseguito. Ad esempio, quando scegli il collegamento fornito in una notifica e-mail per confermare una sottoscrizione in sospeso a un argomento, l'operazione `ConfirmSubscription` viene richiamata in modalità non autenticata. In questo esempio, l'operazione `ConfirmSubscription` non esegue l'accesso a CloudTrail.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, consulta Elemento userIdentity di CloudTrail.

Eventi del piano dati in CloudTrail

Per abilitare la registrazione delle seguenti operazioni API nei file CloudTrail, è necessario abilitare la registrazione dell'attività API del piano dati in CloudTrail. Per ulteriori informazioni, consultare Registrazione di eventi di dati nella Guida per l'utente di AWS CloudTrail.

Gli eventi del piano dati possono anche essere filtrati in base al tipo di risorsa per un controllo granulare sulle chiamate API Amazon SNS che desideri registrare e pagare in modo selettivo in CloudTrail. Ad esempio, specificando `AWS::SNS::Topic` come un tipo di risorsa, puoi registrare le chiamate nelle azioni API `Publish` e `PublishBatch` per gli argomenti. Allo stesso modo, specificando `AWS::SNS::PlatformEndpoint` come un tipo di risorsa, puoi registrare le chiamate nell'azione API Publish per gli endpoint della piattaforma. Per ulteriori informazioni, consulta `AdvancedEventSelector` nella documentazione di riferimento dell'API AWS CloudTrail.

> **Note**
>
> Il tipo di risorsa Amazon SNS `AWS::SNS::PhoneNumber` non viene registrato da CloudTrail.

API del piano dati di Amazon SNS

- `Publish`
- `PublishBatch`

Esempio: voci del file di log di Amazon SNS

Un percorso è una configurazione che consente la distribuzione di eventi come i file di log in un bucket Amazon S3 specificato. I file di log di CloudTrail possono contenere una o più voci di log. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, sulla data e sull'ora dell'operazione, sui parametri richiesti e così via. I file di log CloudTrail non sono una traccia di pila ordinata delle chiamate API pubbliche e di conseguenza non devono apparire in base a un ordine specifico.

L'esempio seguente mostra una voce di log di CloudTrail che illustra l'operazione `ListTopics`, `CreateTopic` o `DeleteTopic`.

```
{
```

```
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "ListTopics",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "nextToken": "ABCDEF1234567890EXAMPLE=="
      },
      "responseElements": null,
      "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
      "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "CreateTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "name": "hello"
```

```
      },
      "responseElements": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "requestID": "example7-5cd3-5323-8a00-f1889011fee9",
      "eventID": "examplec-4f2f-4625-8378-130ac89660b1",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "DeleteTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "responseElements": null,
      "requestID": "example5-4faa-51d5-aab2-803a8294388d",
      "eventID": "example8-6443-4b4d-abfd-1b867280d964",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
  ]
}
```

Gli esempi seguenti mostrano voci di log di CloudTrail che illustrano le azioni `Publish` e `PublishBatch`.

Pubblicare

```
{
```

```
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T16:44:05Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T16:48:37Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "Publish",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "message": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "subject": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageStructure": "json",
  "messageAttributes": "HIDDEN_DUE_TO_SECURITY_REASONS"
 },
 "responseElements": {
  "messageId": "0787cd1e-d92b-521c-a8b4-90434e8ef840"
 },
 "requestID": "0a8ab208-11bf-5e01-bd2d-ef55861b545d",
 "eventID": "bb3496d4-5252-4660-9c28-3c6aebdb21c0",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
```

```
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

PublishBatch

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T19:20:49Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T19:22:01Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "PublishBatch",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
```

```
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "publishBatchRequestEntries": [{
    "id": "1",
    "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
   },
   {
    "id": "2",
    "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
   }
  ]
 },
 "responseElements": {
  "successful": [{
    "id": "1",
    "messageId": "30d68101-a64a-5573-9e10-dc5c1dd3af2f"
   },
   {
    "id": "2",
    "messageId": "c0aa0c5c-561d-5455-b6c4-5101ed84de09"
   }
  ],
  "failed": []
 },
 "requestID": "e2cdf7f3-1b35-58ad-ac9e-aaaea0ace2f1",
 "eventID": "10da9a14-0154-4ab6-b3a5-1825b229a7ed",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
```

```
}
```

Monitoraggio di argomenti Amazon SNS tramite CloudWatch

Amazon SNS e Amazon CloudWatch sono integrati in modo da poter raccogliere, visualizzare e analizzare i parametri per ogni notifica Amazon SNS attiva. Dopo aver configurato CloudWatch per Amazon SNS, avrai una migliore idea delle prestazioni relative a argomenti Amazon SNS, notifiche push e consegne di SMS. Ad esempio, puoi impostare un allarme per ricevere una notifica tramite e-mail se una soglia specificata viene raggiunta per un parametro Amazon SNS come `NumberOfNotificationsFailed`. Per un elenco di tutti i parametri che Amazon SNS invia a CloudWatch, consulta Parametri di Amazon SNS. Per ulteriori informazioni sulle notifiche push Amazon SNS, consulta Notifiche push per dispositivi mobili.

> **Note**
>
> I parametri configurati con CloudWatch per gli argomenti Amazon SNS sono automaticamente raccolti e inviati a CloudWatch ad intervalli di 1 minuto. Questi parametri sono raccolti su tutti gli argomenti considerati come attivi da CloudWatch. Un argomento è considerato attivo da CloudWatch fino a sei ore dall'ultima attività (ovvero, qualsiasi chiamata API) sull'argomento.
>
> Non sono previsti addebiti per i parametri Amazon SNS riportati in CloudWatch in quanto sono forniti come parte del servizio Amazon SNS.

Visualizzazione dei parametri CloudWatch per Amazon SNS

Puoi monitorare i parametri per Amazon SNS utilizzando la console CloudWatch, l'interfaccia a riga di comando (CLI) di CloudWatch oppure a livello di codice utilizzando l'API CloudWatch. Le procedure seguenti mostrano come accedere ai parametri mediante la AWS Management Console.

Per visualizzare i parametri utilizzando la console CloudWatch

1. Accedi alla Console CloudWatch.
2. Nel pannello di navigazione, scegli Metrics (Parametri).
3. Nella scheda All metrics (Tutti i parametri) scegliere SNS, quindi scegliere una delle seguenti dimensioni:

   - Country, SMS Type (Paese, tipo SMS)

- PhoneNumber
- Topic Metrics (Parametri argomento)
- Metrics with no dimensions (Parametri senza dimensioni)

4. Per visualizzare ulteriori dettagli, scegli un elemento specifico. Ad esempio, se si sceglie Topic Metrics (Parametri argomento) e quindi si sceglie NumberOfMessagesPublished, viene visualizzato il numero medio di messaggi Amazon SNS pubblicati per un periodo di cinque minuti durante l'intervallo di tempo di 6 ore.
5. Per visualizzare i parametri di utilizzo di Amazon SNS, nella scheda All metrics (Tutti i parametri), scegli Usage (Utilizzo) e seleziona il parametro di utilizzo di Amazon SNS target (ad esempio, `NumberOfMessagesPublishedPerAccount`).

Impostare gli allarmi CloudWatch per le metriche Amazon SNS

CloudWatch consente inoltre di impostare allarmi quando viene raggiunta la soglia definita per un parametro. Ad esempio, puoi impostare un allarme per il parametro NumberOfNotificationsFailed in modo da ricevere un'e-mail di notifica quando il valore di soglia definito viene raggiunto durante il periodo di campionamento.

Impostazione di allarmi mediante la console CloudWatch

1. Accedi alla AWS Management Console e apri la console CloudWatch all'indirizzo https://console.aws.amazon.com/cloudwatch/.
2. Seleziona Alarms (Allarmi), quindi scegli il pulsante Create Alarm (Crea allarme). Viene avviata la procedura guidata per la creazione di allarmi.
3. Scorri i parametri Amazon SNS per individuare quello per cui vuoi impostare un allarme. Seleziona il parametro per il quale vuoi creare un allarme e scegli Continue (Continua).
4. Indica i valori Name (Nome), Description (Descrizione), Threshold (Soglia) e Time (Ora) per il parametro e scegli Continue (Continua).
5. Scegli Alarm (Allarme) come stato dell'allarme. Se vuoi ricevere un'e-mail da CloudWatch quando viene raggiunto lo stato dell'allarme, seleziona un argomento Amazon SNS esistente o scegli Create New Email Topic (Crea nuovo argomento e-mail). Se scegli Create New Email Topic (Crea nuovo argomento e-mail), puoi impostare il nome e gli indirizzi e-mail per un nuovo argomento. Questo elenco viene salvato ed è visualizzato nella casella di riepilogo a discesa per gli allarmi futuri. Scegli Continue (Continua).

> **Note**
>
> Se utilizzi Create New Email Topic (Crea nuovo argomento e-mail) per creare un nuovo argomento Amazon SNS, gli indirizzi e-mail devono essere verificati prima di ricevere le notifiche. Le e-mail sono inviate solo quando l'allarme passa allo stato definito. Se lo stato cambia prima della verifica degli indirizzi e-mail, questi non riceveranno una notifica.

6. A questo punto, la procedura guidata per la creazione di allarmi ti consente di esaminare l'allarme che stai per creare. Se devi apportare delle modifiche, puoi utilizzare i collegamenti Edit (Modifica) a destra. Al termine, scegli Create Alarm (Crea allarme).

Per ulteriori informazioni sull'utilizzo di e degli allarmi, consulta la documentazione di Amazon CloudWatch.

Parametri di Amazon SNS

Amazon SNS invia i parametri seguenti a CloudWatch.

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfMessagesPublished`	Il numero di messaggi pubblicati ai tuoi argomenti Amazon SNS. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum
AWS/SNS	`NumberOfNotificationsDelivered`	Il numero di messaggi consegnati con successo dai tuoi argomenti Amazon SNS agli endpoint sottoscritti.

Spazio dei nomi	Parametro	Descrizione
		Affinché un tentativo di invio abbia successo, la sottoscrizione dell'endpoint deve accettare il messaggio. Una sottoscrizione accetta un messaggio se a.) manca di un criterio di filtro o b.) i criteri di filtro includono attributi che corrispondono a quelli assegnati al messaggio. Se la sottoscrizione rifiuta il messaggio, il tentativo di invio non viene conteggiato per questo parametro. Unità: numero Dimensioni valide: Applicati on, PhoneNumber, Platform e TopicName Statistiche valide: Sum

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfNotificationsFailed`	Il numero di messaggi non consegnati da Amazon SNS. Per Amazon SQS, e-mail, SMS o endpoint push mobili, il parametro aumenta di 1 quando Amazon SNS interrompe i tentativi di consegna dei messaggi. Per gli endpoint HTTP o HTTPS, il parametro include ogni tentativo di invio non riuscito, inclusi i nuovi tentativi che seguono il tentativo iniziale. Per tutti gli altri endpoint, il numero aumenta di 1 quando il messaggio non viene distribuito (indipendentemente dal numero di tentativi). Questo parametro non include i messaggi rifiutati da policy di filtro di sottoscrizione. Puoi controllare il numero di nuovi tentativi per gli endpoint HTTP. Per ulteriori informazioni, consulta Tentativi di consegna dei messaggi di Amazon SNS. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfNotificationsFilteredOut`	Il numero di messaggi rifiutati da policy di filtro di sottoscrizione. Una policy di filtro rifiuta un messaggio quando gli attributi del messaggio non corrispondono agli attributi della policy. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Il numero di messaggi rifiutati dalle policy di filtro delle sottoscrizioni per il filtro basato su attributi. Unità: CountValid Dimensioni: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Il numero di messaggi rifiutati dalle policy di filtro delle sottoscrizioni per il filtro basato sul payload. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Il numero di messaggi rifiutati dalle policy di filtro di sottoscrizione perché gli attributi dei messaggi non sono validi - ad esempio perché l'attributo JSON non è formattato correttamente. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	Il numero di messaggi che sono stati rifiutati da policy di filtro di sottoscrizione perché i messaggi non hanno attributi. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Il numero di messaggi rifiutati dalle policy di filtro delle sottoscrizioni perché il corpo del messaggio non è valido per il filtro (ad esempio, corpo del messaggio JSON non valido). Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Numero di messaggi che sono stati spostati in una coda dead-letter. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Numero di messaggi che non possono essere spostati in una coda dead-letter. Unità: numero Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Sum, Average

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`PublishSize`	Dimensione dei messaggi pubblicati. Unità: byte Dimensioni valide: Application, PhoneNumber, Platform e TopicName Statistiche valide: Minimum, Maximum, Average, Count

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`SMSMonthToDateSpentUSD`	I costi che hai cumulato dall'inizio del mese di calendario corrente per l'invio di SMS. Puoi impostare un allarme per questo parametro in modo da essere avvisato quando l'ammontare dei costi mensili è prossimo al limite di spesa SMS mensile per il tuo account. Quando Amazon SNS determina che l'invio di un SMS comporterebbe un costo superiore a tale limite, interrompe la pubblicazione di SMS nel giro di alcuni minuti. Per informazioni sull'impostazione del limite di spesa SMS mensile o sulla richiesta di aumento del limite di spesa con AWS, consulta Impostazione delle preferenze di messaggistica SMS. Unità: USD Dimensioni valide: PhoneNumber Statistiche valide: massimo
AWS/SNS	`SMSSuccessRate`	Il tasso di consegne SMS riuscite. Unità: numero Dimensioni valide: PhoneNumber Statistiche valide: somma, media, esempi di dati

Dimensioni per i parametri Amazon SNS

Amazon Simple Notification Service invia le dimensioni seguenti a CloudWatch.

Dimensione	Descrizione
`Application`	Filtra in base agli oggetti applicazione, che rappresentano un'app e un dispositivo registrati in uno dei servizi di notifica push supportati, ad esempio APN e FCM.
`Application,Platform`	Filtra in base agli oggetti applicazione e piattaforma, dove gli oggetti piattaforma sono destinati ai servizi di notifica push supportati, ad esempio APN e FCM
`Country`	Filtra in base al paese o alla regione di destinazione di un SMS. Il paese o la regione è rappresentato dal relativo codice ISO 3166-1 alpha-2.
`PhoneNumber`	Filtra il numero di telefono quando pubblichi SMS direttamente su un numero di telefono (senza argomento).
`Platform`	Filtra in base agli oggetti piattaforma per i servizi di notifica push, ad esempio APN e FCM.
`TopicName`	Filtra in base ai nomi di argomenti Amazon SNS.
`SMSType`	Filtra in base al tipo di SMS. Può essere promozionale o transazionale.

Parametri di utilizzo di Amazon SNS

Amazon Simple Notification Service invia i seguenti parametri di utilizzo a CloudWatch.

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
AWS/Usage	SNS	`ResourceCount`	`NumberOfMessagesPu`	Risorsa	• Il numero di

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
			`blishedPerAccount`		messaggi pubblicati sui tuoi argomenti Amazon SNS nel tuo account AWS. • Unità: nessuna • Statistiche valide: Sum
AWS/Usage	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Risorsa	• Il numero approssimativo di argomenti nel tuo account AWS. • Unità: nessuna • Statistiche valide: Average (Media), Minimum (Minimo), Maximum (Massimo), Sum (Somma)

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
AWS/Usage	SNS	`ResourceCount`	`ApproximateNumberOfFilterPolicies`	Risorsa	• Il numero approssimativo di policy di filtro nel tuo account AWS. • Unità: nessuna • Statistiche valide: Average (Media), Minimum (Minimo), Maximum (Massimo), Sum (Somma)

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
AWS/Usage	SNS	`ResourceCount`	`ApproximateNumberOfPendingSubscriptions`	Risorsa	• Il numero approssimativo di abbonamenti in sospeso nel tuo account AWS. • Unità: nessuna • Statistiche valide: Average (Media), Minimum (Minimo), Maximum (Massimo), Sum (Somma)

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
AWS/Usage	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber`	API	• Il numero di chiamate API per l'API Amazon SNS selezionata nel tuo account AWS. • Unità: nessuna • Statistiche valide: Sum

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
			• DeleteTopic • GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut		

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
			• ListPlatformApplications • ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes		

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
			• SetSubscriptionAttributes • SetTopicAttributes • Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Convalida della conformità per Amazon SNS

I revisori di terze parti valutano la sicurezza e la conformità di Amazon SNS nell'ambito di più programmi di conformità AWS, incluso il programma Health Insurance Portability and Accountability Act (HIPAA).

Per un elenco dei servizi AWS che rientrano nell'ambito di programmi di conformità specifici, consulta Servizi AWS che rientrano nell'ambito del programma di conformità . Per informazioni generali, consulta Programmi per la conformità di AWS.

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download di report in AWS Artifact.

La responsabilità di conformità durante l'utilizzo di Amazon SNS è determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità AWS mette a disposizione le seguenti risorse:

- Guide Quick Start Sicurezza e compliance Guide Quick Start – Queste guide alla distribuzione illustrano considerazioni relative all'architettura e forniscono procedure per la distribuzione di ambienti di base incentrati sulla sicurezza e sulla conformità su AWS.
- Whitepaper sulla progettazione per la sicurezza HIPAA e la conformità: questo whitepaper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni conformi ai requisiti HIPAA.
- Risorse per la conformitàAWS - Una raccolta di cartelle di lavoro e guide suddivise per settore e area geografica.
- Valutazione delle risorse con le regole nella Guida per lo sviluppatore di AWS Config: il servizio AWS Config valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti.
- AWS Security Hub - Questo servizio AWS fornisce una visione completa dello stato di sicurezza all'interno di AWS che consente di verificare la conformità con gli standard e le best practice di sicurezza del settore.

Resilienza in Amazon SNS

La resilienza in Amazon SNS è garantita dall'utilizzo AWS dell'infrastruttura globale, che ruota attorno Regioni AWS alle zone di disponibilità. Regioni AWS offrono zone di disponibilità fisicamente separate e isolate collegate tramite reti a bassa latenza, ad alto throughput e altamente ridondanti. Questa architettura consente un failover senza interruzioni tra le zone di disponibilità, rendendo le applicazioni e i database intrinsecamente più tolleranti ai guasti e scalabili rispetto alle tradizionali infrastrutture di data center. Utilizzando le zone di disponibilità, gli abbonati Amazon SNS beneficiano di una maggiore disponibilità e affidabilità, garantendo la consegna dei messaggi nonostante potenziali interruzioni. Per ulteriori informazioni sulle zone di disponibilità, Regioni AWS consulta Global Infrastructure.AWS

Inoltre, gli abbonamenti agli argomenti di Amazon SNS possono essere configurati con nuovi tentativi di consegna e code di lettera morta, abilitando la gestione automatica degli errori transitori e garantendo che i messaggi raggiungano in modo affidabile le destinazioni previste.

Amazon SNS supporta anche il filtraggio e gli attributi dei messaggi, che consentono di personalizzare le strategie di resilienza in base ai casi d'uso specifici, migliorando la robustezza complessiva delle applicazioni.

Sicurezza dell'infrastruttura in Amazon SNS

In quanto servizio gestito, Amazon SNS è protetto dalle procedure di sicurezza di rete AWS globali riportate nella documentazione sulle migliori pratiche per la sicurezza, l'identità e la conformità.

Utilizza le azioni AWS API per accedere ad Amazon SNS attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.2 o versioni successive. I client devono, inoltre, supportare le suite di cifratura con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE).

Le richieste devono essere firmate utilizzando sia un ID chiave di accesso che una chiave di accesso segreta associate a un'entità principale IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare credenziali di sicurezza temporanee per le richieste di firma.

Puoi richiamare queste operazioni API da qualsiasi posizione di rete, ma Amazon SNS supporta le policy di accesso basate sulle risorse, che possono includere limitazioni in base all'indirizzo IP di origine. È inoltre possibile utilizzare le policy di Amazon SNS per controllare l'accesso da endpoint Amazon VPC o VPC specifici. Ciò isola efficacemente l'accesso alla rete a un determinato argomento di Amazon SNS solo dal VPC specifico all'interno della rete. AWS Per ulteriori informazioni, consulta Limitare la pubblicazione a un argomento Amazon SNS solo da un endpoint VPC specifico.

Best practice di sicurezza per Amazon SNS

AWS fornisce molte caratteristiche di sicurezza per Amazon SNS. Esaminare queste caratteristiche di sicurezza nel contesto delle policy di sicurezza personalizzate.

> **Note**
>
> Le indicazioni per queste caratteristiche di sicurezza si applicano ai casi d'utilizzo comuni e alle implementazioni. Si consiglia di esaminare le best practice nel contesto del caso d'uso specifico, dell'architettura e del modello di minaccia.

Best practice di prevenzione

Di seguito sono riportate le best practice di prevenzione per la sicurezza per Amazon SNS.

Argomenti

- Assicurarsi che gli argomenti non siano accessibili pubblicamente

- Implementazione dell'accesso con privilegi minimi
- Utilizzare ruoli IAM per le applicazioni e i servizi AWS che richiedono l'accesso Amazon SNS
- Implementare la crittografia lato server
- Applica la crittografia dei dati in transito
- Prendere in considerazione l'utilizzo di endpoint VPC per accedere a Amazon SNS
- Assicurarsi che le sottoscrizioni non siano configurate per il recapito agli endpoint http non elaborati

Assicurarsi che gli argomenti non siano accessibili pubblicamente

A meno che non sia esplicitamente necessario che chiunque su Internet debba essere in grado di leggere o scrivere nell'argomento Amazon SNS, dovresti assicurarti che la l'argomento non sia accessibile pubblicamente (accessibile da tutti nel mondo o da qualsiasi utente AWS autenticato).

- Evitare di creare policy con `Principal` impostato su `""`.
- Evitare di utilizzare un carattere jolly (`*`). Assegnare invece un nome a uno o più utenti specifici.

Implementazione dell'accesso con privilegi minimi

Quando si concedono autorizzazioni, si decide chi le riceve, per quali argomenti vengono fornite le autorizzazioni e le operazioni API specifiche che si desidera consentire per tali argomenti. Implementare il principio del privilegio minimo è importante per ridurre i rischi per la sicurezza. Aiuta anche a ridurre l'effetto negativo di errori o intenti dannosi.

Seguire i consigli di sicurezza standard relativi alla concessione dei privilegi minimi. Cioè, concedere solo le autorizzazioni necessarie per eseguire un'attività specifica. È possibile implementare i privilegi minimi utilizzando una combinazione di policy di sicurezza relative all'accesso degli utenti.

Amazon SNS utilizza il modello entità di pubblicazione-sottoscrittore, che richiede tre tipi di accesso dell'account utente:

- Amministratori - Accesso alla creazione, alla modifica e all'eliminazione di argomenti. Gli amministratori controllano anche le policy degli argomenti.
- Editori - Accesso all'invio di messaggi negli argomenti.
- Subscribers - Accesso alla sottoscrizione agli argomenti.

Per ulteriori informazioni, consulta le sezioni seguenti:

- Identity and Access Management in Amazon SNS
- Autorizzazioni API Amazon SNS: riferimento a operazioni e risorse

Utilizzare ruoli IAM per le applicazioni e i servizi AWS che richiedono l'accesso Amazon SNS

Per applicazioni o servizi AWS, ad esempio Amazon EC2, per l'accesso agli argomenti Amazon SNS, è necessario utilizzare credenziali AWS valide nelle richieste API AWS. Poiché queste credenziali non vengono ruotate automaticamente, non è necessario archiviare le credenziali AWS direttamente nell'applicazione o nell'istanza EC2.

È preferibile usare un ruolo IAM per gestire credenziali provvisorie per le applicazioni o i servizi che devono accedere ad Amazon SNS. Quando utilizzi un ruolo, non è necessario distribuire credenziali a lungo termine (ad esempio, nome utente, password e chiavi di accesso) a un'istanza EC2 o a un servizio AWS, ad esempio AWS Lambda. Al contrario, il ruolo fornisce autorizzazioni provvisorie che possono essere utilizzate dalle applicazioni durante le chiamate ad altre risorse AWS.

Per ulteriori informazioni, consultaRuoli IAM e Scenari comuni per ruoli: utenti, applicazioni e servizi nella Guida per l'utente IAM.

Implementare la crittografia lato server

Per attenuare i problemi di perdita di dati, utilizzare la crittografia dei dati inattivi per crittografare i messaggi utilizzando una chiave memorizzata in un percorso diverso da quello in cui vengono archiviati i messaggi. La crittografia lato server (SSE) fornisce la crittografia dei dati inattivi. Amazon SNS esegue la crittografia dei dati a livello di messaggio quando li memorizza e decrittografa i messaggi per l'utente quando vi accede. SSE utilizza le chiavi gestite in AWS Key Management Service. Quando si autentica la richiesta e si dispone delle autorizzazioni di accesso, non vi è alcuna differenza tra l'accesso agli argomenti crittografati e non crittografati.

Per ulteriori informazioni, consulta Crittografia a riposo e Gestione delle chiavi.

Applica la crittografia dei dati in transito

È possibile, ma non consigliato, pubblicare messaggi che non sono crittografati durante il transito utilizzando HTTP. Tuttavia, non è possibile utilizzare HTTP quando si pubblica su un argomento SNS crittografato.

AWS consiglia di utilizzare HTTPS invece di HTTP. Quando si utilizza HTTPS, i messaggi vengono crittografati automaticamente durante il transito, anche se l'argomento SNS non è crittografato. Senza HTTPS, un utente malintenzionato basato sulla rete può spiare il traffico di rete o manipolarlo, utilizzando un attacco di tipo man-in-the-middle.

Per applicare solo le connessioni crittografate su HTTPS, aggiungere il `aws:SecureTransport` nel criterio IAM allegato agli argomenti SNS non crittografati. In questo modo gli editori di messaggi devono utilizzare HTTPS anziché HTTP. È possibile utilizzare il seguente criterio di esempio come guida:

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:1234567890:test-topic"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

Prendere in considerazione l'utilizzo di endpoint VPC per accedere a Amazon SNS

Se si dispone di argomenti con cui è necessario essere in grado di interagire ma che non devono assolutamente essere esposti a Internet, utilizzare gli endpoint VPC per limitare l'accesso agli argomenti solo agli host all'interno di un determinato VPC. È possibile utilizzare le policy degli argomenti per controllare l'accesso agli argomenti da endpoint Amazon VPC specifici o da VPC specifici.

Gli endpoint VPC di Amazon SNS offrono due modi per controllare l'accesso ai messaggi:

- È possibile controllare le richieste, gli utenti o i gruppi autorizzati tramite un endpoint VPC specifico.
- È possibile controllare quali VPC o endpoint VPC hanno accesso all'argomento utilizzando una policy dell'argomento.

Per ulteriori informazioni, consulta Creazione dell'endpoint e Creazione di una policy per endpoint VPC di Amazon per Amazon SNS.

Assicurarsi che le sottoscrizioni non siano configurate per il recapito agli endpoint http non elaborati

Evitare di configurare le sottoscrizioni per il recapito a endpoint http non elaborati. Disponete sempre di sottoscrizioni che consegnano a un nome di dominio endpoint. Ad esempio, una sottoscrizione configurata per la consegna a un endpoint, `http://1.2.3.4/my-path`, dovrebbe essere cambiata in `http://my.domain.name/my-path`.

Risoluzione dei problemi argomenti di Amazon SNS

In questa sezione vengono fornite informazioni sulla risoluzione dei problemi degli argomenti Amazon SNS.

Argomenti Amazon SNS tramite AWS X-Ray

AWS X-Ray raccoglie i dati sulle richieste relative alla tua applicazione e ti consente di visualizzare e filtrare i dati per identificare i potenziali problemi e le opportunità di ottimizzazione. Per qualsiasi richiesta tracciata che raggiunge la tua applicazione, puoi visualizzare informazioni dettagliate non solo sulla richiesta ma anche sulle chiamate che la tua applicazione esegue verso le risorse AWS, i microservizi, i database e le API web HTTP a valle.

È possibile utilizzare X-Ray con Amazon SNS per tracciare e analizzare i messaggi che passano attraverso la tua applicazione. È possibile utilizzare il AWS Management Console per visualizzare la mappa delle connessioni tra Amazon SNS e altri servizi impiegati dall'applicazione. È inoltre possibile utilizzare la console per visualizzare i parametri come la latenza media e le percentuali di errore. Per ulteriori informazioni, consultare Amazon SNS e AWS X-Ray nella AWS X-Ray Guida per sviluppatori.

Tracciamento attivo in Amazon SNS

Puoi utilizzarlo AWS X-Ray per tracciare e analizzare le richieste degli utenti mentre passano dagli argomenti di Amazon SNS agli abbonamenti agli endpoint Amazon Data Firehose, Amazon AWS LambdaSQS e HTTP/S. Poiché X-Ray ti offre una end-to-end visualizzazione dell'intera richiesta, puoi visualizzare ciò che chiama il tuo argomento Amazon SNS e ciò che è a valle degli abbonamenti del tuo argomento. È possibile analizzare le latenze nei tuoi messaggi e nei relativi servizi di backend, ad esempio, è possibile vedere quanto tempo trascorre una richiesta in un argomento e quanto tempo impiega per recapitare il messaggio a ciascuna delle sottoscrizioni dell'argomento.

> **Important**
>
> Gli argomenti di Amazon SNS con numerose sottoscrizioni possono raggiungere il limite di dimensioni consentito e non essere tracciati del tutto. Per informazioni sui limiti di dimensioni per il tracciamento dei documenti, consulta le quote di servizio di X-Ray nella pagina relativa ai riferimenti generali AWS.

Se chiami un'API Amazon SNS da un servizio che viene già tracciato, Amazon SNS esegue il pass-through del tracciamento, anche se sull'API non è abilitato il tracciamento di X-Ray.

Amazon SNS supporta il tracciamento di X-Ray per gli argomenti standard e FIFO. È possibile abilitare X-Ray per un argomento Amazon SNS utilizzando la console Amazon SNS, l'API `SetTopicAttributes` Amazon SNS, la documentazione di riferimento della CLI del Servizio di notifica semplice Amazon o AWS CloudFormation.

Per ulteriori informazioni sull'utilizzo di Amazon SNS con X-Ray, consulta Amazon SNS e AWS X-Ray nella Guida per gli sviluppatori di AWS X-Ray.

Argomenti

Autorizzazioni per il tracciamento attivo

Quando si utilizza la console Amazon SNS, Amazon SNS tenta di creare le autorizzazioni necessarie per consentire all'argomento Amazon SNS di chiamare X-Ray. Il tentativo può essere rifiutato se si non dispone di autorizzazioni sufficienti per utilizzare la console Amazon SNS. Per ulteriori informazioni, consultare Identity and Access Management in Amazon SNS e Esempi di casi per il controllo degli accessi Amazon SNS.

Quando si utilizza la CLI, è necessario configurare manualmente le autorizzazioni. Tali autorizzazioni vengono configurate utilizzando le policy delle risorse. Per ulteriori informazioni sull'utilizzo delle autorizzazioni richieste in X-Ray, consulta Amazon SNS e AWS X-Ray.

Abilitazione del tracciamento attivo su un argomento Amazon SNS (console)

Quando il tracciamento attivo è abilitato su un argomento Amazon SNS, legge l'ID di tracciamento, lo utilizza per inviare i dati al cliente e lo propaga ai servizi a valle.

1. Accedi alla console Amazon SNS.
2. Scegli un argomento o creane uno nuovo. Per ulteriori dettagli sulla creazione di argomenti, consulta Creare un argomento Amazon SNS.
3. Nella pagina Crea argomento, nella sezione Dettagli scegli un tipo di argomento: FIFO o Standard.

   a. Immetti un nome per l'argomento.

   b. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.
4. Espandi Active tracing (Monitoraggio attivo) e scegli Use active tracing (Usa tracciamento attivo).

Dopo aver abilitato X-Ray per il tuo argomento Amazon SNS, puoi utilizzare la mappa dei servizi X-Ray per visualizzare le tracce e le mappe dei servizi per end-to-end l'argomento.

Abilitazione del tracciamento attivo su un argomento Amazon SNS (AWS SDK)

L'esempio di codice seguente mostra come abilitare il tracciamento attivo su un argomento Amazon SNS utilizzando AWS SDK per Java.

```
public static void enableActiveTracing(SnsClient snsClient, String topicArn) {

        try {

            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName("TracingConfig")
                .attributeValue("Active")
                .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());
        }
    }
```

Abilitazione del tracciamento attivo su un argomento Amazon SNS (AWS CLI)

L'esempio di codice seguente mostra come abilitare il tracciamento attivo su un argomento Amazon SNS utilizzando AWS CLI.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Abilitazione del tracciamento attivo su un argomento Amazon SNS (AWS CloudFormation)

Il seguente stack AWS CloudFormation mostra come abilitare il tracciamento attivo su un argomento Amazon SNS.

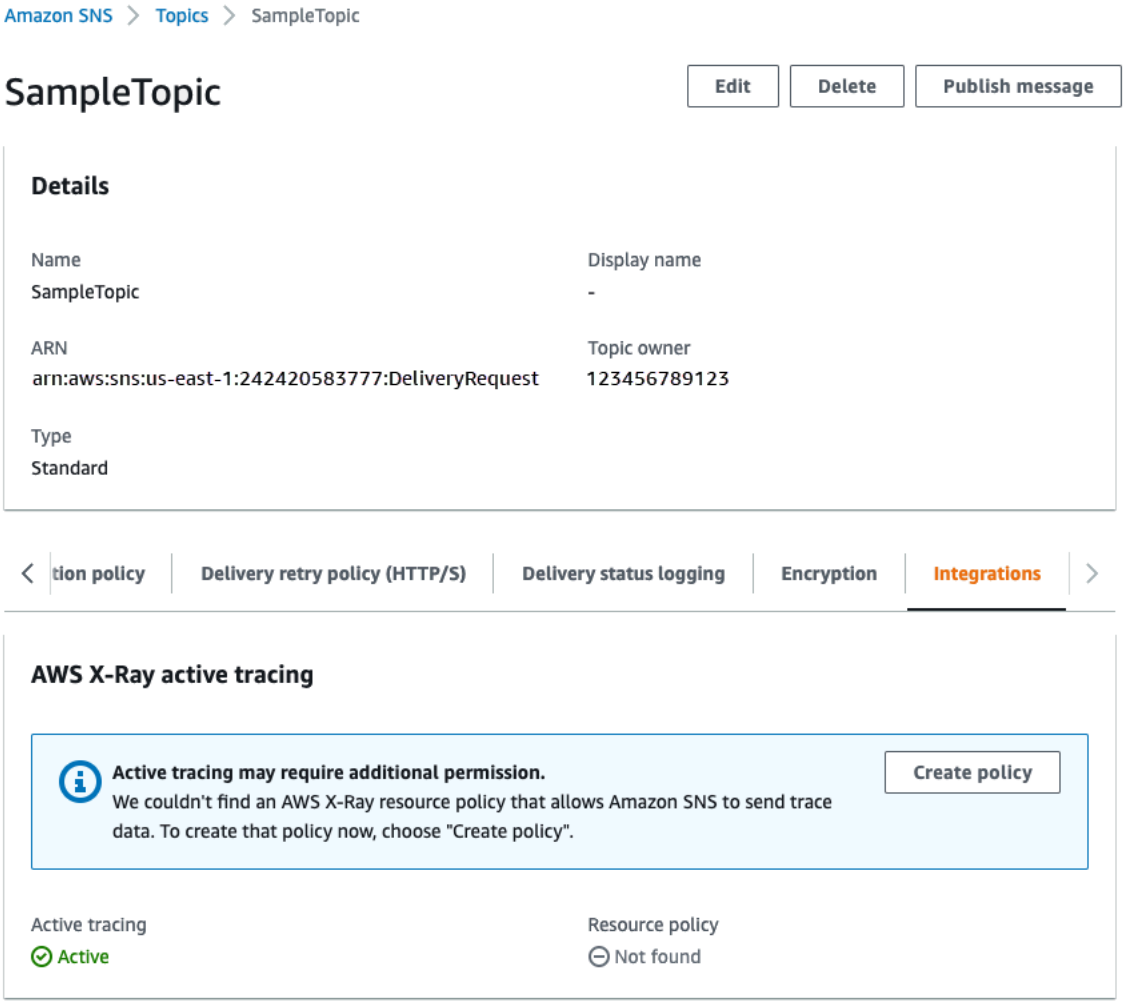
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Verifica dell'abilitazione del tracciamento attivo per l'argomento

Puoi utilizzare la console Amazon SNS per verificare se il tracciamento attivo è abilitato per il tuo argomento o quando la policy delle risorse non è stata aggiunta.

1. Accedi alla console Amazon SNS.
2. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).
3. Nella pagina Topics (Argomenti), scegli un argomento.
4. Seleziona la scheda Integrations (Integrazioni).

   Quando il tracciamento attivo è abilitato, viene mostrata l'icona verde Active (Attivo).
5. Se hai abilitato il tracciamento attivo e non vedi che la policy delle risorse è stata aggiunta, scegli Create policy (Crea policy) per aggiungere le ulteriori autorizzazioni richieste.



Test del tracciamento attivo

1. Accedi alla console Amazon SNS.
2. Creazione di un argomento Amazon SNS. Per informazioni dettagliate su come eseguire questa operazione, consulta Per creare un argomento utilizzando il AWS Management Console.
3. Espandi Active tracing (Monitoraggio attivo) e scegli Use active tracing (Usa tracciamento attivo).
4. Pubblica un messaggio nel tuo argomento Amazon SNS. Per informazioni dettagliate su come eseguire questa operazione, consulta Per pubblicare messaggi su argomenti Amazon SNS utilizzando il AWS Management Console.
5. Utilizza la mappa dei servizi X-Ray per visualizzare le end-to-end tracce e le mappe dei servizi per l'argomento.


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

Cronologia della documentazione

Nella tabella seguente sono descritte le modifiche recenti apportate a Amazon Simple Notification Service Guida per sviluppatori.

Le funzionalità del servizio a volte vengono implementate in modo incrementale nelle AWS regioni in cui il servizio è disponibile. Aggiorniamo questa documentazione solo per la prima versione. Non forniamo informazioni sulla disponibilità delle regioni e non annunciamo implementazioni successive delle regioni. Per informazioni sulla disponibilità regionale delle funzionalità del servizio e per iscriverti alle notifiche sugli aggiornamenti, vedi Cosa c'è di AWS nuovo? .

Modifica	Descrizione	Data
Supporto di Canada West (Calgary) per gli argomenti FIFO	Amazon SNS supporta l'argomento FIFO in Canada occidentale (Calgary).	28 marzo 2024
Supporto SMS di Amazon SNS in cinque nuove regioni	Amazon SNS ha aggiunto il supporto SMS alle seguenti regioni: Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Medio Oriente (Emirati Arabi Uniti), Europa (Zurigo) ed Europa (Spagna).	8 febbraio 2024
Supporto HTTP v1 per Firebase Cloud Messaging (FCM)	Amazon SNS supporta le credenziali FCM v1.	18 gennaio 2024
Amazon SNS supporta gli SMS in Asia Pacifico (Giacarta)	Amazon SNS supporta la messaggistica SMS in Asia Pacifico (Giacarta)	14 dicembre 2023
AWS CloudFormation supporto per la configurazione `DeliveryStatusLogging` per argomenti di Amazon SNS	AWS CloudFormation è disponibile il supporto per la configurazione `DeliveryStatusLogging` durante la creazione o l'aggiornamento	7 dicembre 2023

	degli argomenti di Amazon SNS.	
Nuovi operatori di filtro messaggi aggiunti	Ora puoi utilizzare gli operatori suffix matching, equals-ignore case e OR per filtrare i messaggi Amazon SNS.	16 novembre 2023
Supporto aggiunto per l'archiviazione e riproduzione dei messaggi	I proprietari degli argomenti possono archiviare i messaggi relativi a un argomento per un massimo di 365 giorni. Gli abbonati agli argomenti possono riprodurre i messaggi archiviati su un endpoint sottoscritto per recuperare i messaggi causati da un errore in un'applicazione downstream o per replicare lo stato di un'applicazione esistente.	26 ottobre 2023
Support aggiunto per la sottoscrizione di una coda standard a un argomento FIFO	È possibile effettuare la sottoscrizione di una coda FIFO o di una coda standard di Amazon SQS a un argomento FIFO di Amazon SNS SNS. Solo le code FIFO di Amazon SQS garantiscono che i messaggi vengano ricevuti in ordine e senza duplicati.	14 settembre 2023
Aggiunto il supporto SMS per Israele (Tel Aviv)	SMS di Amazon SNS è ora supportato nella regione Israele (Tel Aviv).	28 agosto 2023

Supporto per il tracciamento attivo di X-Ray aggiunto agli argomenti FIFO	In precedenza supportata solo con argomenti standard di Amazon SNS, AWS X-Ray ora traccia e analizza le richieste degli utenti mentre passano dagli argomenti FIFO agli abbonamenti Amazon Data Firehose, AWS Lambda Amazon SQS e HTTP/S endpoint.	31 maggio 2023
Supporto intestazione Content-Type avanzato	Puoi impostare l'intestazione Content-Type nella policy di richiesta per specificare il tipo di supporto della notifica.	23 marzo 2023
Aggiunto il supporto per il tracciamento attivo	AWS X-Ray traccia e analizza le richieste degli utenti mentre viaggiano attraverso gli argomenti standard di Amazon SNS verso gli abbonamenti agli endpoint Amazon Data Firehose, AWS Lambda Amazon SQS e HTTP/S.	8 febbraio 2023
Registrazione dell'ID mittente di Singapore	Sono state aggiunte le istruzioni per la registrazione degli ID mittente di Singapore.	10 gennaio 2023
Filtro dei messaggi in base al payload	Il filtro basato sul payload consente di filtrare i messaggi in base al payload ed evitare i costi associati all'elaborazione di dati indesiderati.	22 novembre 2022

Aggiunta dell'algoritmo hash SHA256 per la firma dei messaggi di Amazon SNS	Aggiunta del supporto per l'algoritmo hash SHA256 in caso di utilizzo della firma dei messaggi di Amazon SNS.	15 settembre 2022
Aggiunta di ulteriori aree alla messaggistica SMS	Amazon SNS supporta la messaggistica SMS nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Osaka), Europa (Milano) e AWS GovCloud (Stati Uniti orientali).	9 settembre 2022
Aggiunta del supporto per la protezione dei dati dei messaggi	La protezione dei dati dei messaggi salvaguarda i dati pubblicati negli argomenti di Amazon SNS utilizzando policy di protezione dei dati per controllare e bloccare le informazioni sensibili che si spostano tra applicazioni AWS o servizi.	8 settembre 2022
Nuova procedura di registrazione per i numeri verdi	È stato aggiunto il supporto per l'invio di messaggi Amazon SNS utilizzando numeri verdi (TFN) a destinatari negli Stati Uniti.	1 agosto 2022

Supporto per il controllo degli accessi basato sugli attributi (ABAC)	Aggiunto il supporto per il controllo degli accessi basato su attributi (ABAC) per le azioni API, tra cui `Publish` e `PublishBatch` . ABAC è una strategia di autorizzazione che definisce le autorizzazioni di accesso in base a tag che possono essere allegati a risorse IAM, come utenti e ruoli IAM, e a AWS risorse, come gli argomenti di Amazon SNS, per semplificare la gestione delle autorizzazioni.	10 gennaio 2022
Supporto per l'autenticazione basata su token Apple per le notifiche push	Puoi autorizzare Amazon SNS a inviare notifiche push all'app iOS o macOS fornendo informazioni che identificano l'utente come sviluppatore dell'app.	28 ottobre 2021
I nuovi mittenti di messaggi SMS vengono inseriti nella sandbox (ambiente di sperimentazione) SMS	La sandbox SMS è disponibile per prevenire frodi e abusi e per aiutare a proteggere la tua reputazione come mittente. Mentre il tuo AWS account è nella sandbox SMS, puoi inviare messaggi SMS solo a numeri di telefono di destinazione verificati.	1 giugno 2021

I nuovi mittenti di messaggi SMS vengono inseriti nella sandbox (ambiente di sperimentazione) SMS	La sandbox SMS è disponibile per prevenire frodi e abusi e per aiutare a proteggere la tua reputazione come mittente. Mentre il tuo AWS account è nella sandbox SMS, puoi inviare messaggi SMS solo a numeri di telefono di destinazione verificati.	1 giugno 2021
Nuovi attributi per l'invio di messaggi SMS ai destinatari in India	Due nuovi attributi, ID entità e ID modello, sono ora necessari per l'invio di messaggi SMS ai destinatari in India.	22 Aprile 2021
Aggiornamenti agli operatori di filtro messaggi	Un nuovo operatore, `cidr`, è disponibile per gli indirizzi IP e le subnet dell'origine dei messaggi corrispondenti. Ora è anche possibile verificare l'assenza di una chiave di attributo e utilizzare un prefisso con l' `anything-but` operatore per la corrispondenza della stringa di attributo.	7 Aprile 2021

Termine del supporto per i codici lunghi P2P per le destinazioni degli Stati Uniti	A partire dal 1° giugno 2021, i provider di telecomunicazioni statunitensi non supportan o più l'uso di codici lunghi person-to-person (P2P) per le comunicazioni application-to-person (A2P) verso destinazi oni negli Stati Uniti. Invece, è possibile utilizzare codici brevi, numeri verdi o un nuovo tipo di numero di origine chiamato 10DLC.	16 febbraio 2021
Support per i parametri Amazon CloudWatch di 1 minuto	Il CloudWatch parametro di 1 minuto per Amazon SNS è ora disponibile in tutte le regioni. AWS	28 gennaio 2021
Supporto per gli endpoint Amazon Data Firehose	È possibile abbonare i flussi di distribuzione di Firehose agli argomenti SNS. Ciò consente di inviare notifiche a endpoint di archiviazione e analisi come bucket Amazon Simple Storage Service (Amazon S3), tabelle Amazon Redshift, Amazon Service (Service) e altro ancora OpenSearch . OpenSearch	12 gennaio 2021
I numeri di origine sono disponibili	È possibile utilizzare i numeri di origine quando si inviano messaggi di testo (SMS).	23 ottobre 2020

Supporto per gli argomenti FIFO di Amazon SNS	Per integrare applicazioni distribuite che richiedono coerenza dei dati in tempo quasi reale, puoi utilizzare gli argomenti FIFO (First-in, First-Out) di Amazon SNS con le code FIFO di Amazon SQS.	22 ottobre 2020
La libreria client ampia Amazon SNS per Java è disponibile	Puoi utilizzare questa libreria per pubblicare messaggi Amazon SNS di grandi dimensioni.	25 agosto 2020
SSE è disponibile nelle regioni della Cina	La crittografia lato server (SSE) per Amazon SNS è disponibile nelle regioni della Cina.	20 gennaio 2020
Supporto per l'utilizzo di DLQ per acquisire messaggi non recapitabili	È possibile utilizzare una coda Amazon SQS dead-letter con una sottoscrizione Amazon SNS per acquisire messaggi non recapitabili.	14 novembre 2019
Supporto per la specifica di valori di intestazione APN personalizzati	È possibile specificare un valore di intestazione APN personalizzato.	18 ottobre 2019
Supporto per il Campo di intestazione per APN "`apns-push-type `"	Puoi utilizzare il campo di intestazione `apns-push-type` per le notifiche per dispositivi mobili inviate tramite APN.	10 settembre 2019
Support per la risoluzione dei problemi tramite AWS X-Ray	È possibile risolvere i problemi dei messaggi che passano attraverso gli argomenti SNS utilizzando X-Ray.	24 luglio 2019

Supporto per la corrispondenza delle chiavi degli attributi tramite l'operatore "`exists`"	È possibile utilizzare l'operatore `exists` per controllare se un messaggio in ingresso dispone di un attributo la cui chiave compare nella policy di filtro.	5 luglio 2019
Support per "Anything-but" consente la corrispondenza di più valori numerici	Oltre alle stringhe multiple, Amazon SNS consente la corrispondenza "Anything-but" di più valori numerici.	5 luglio 2019
Le note di rilascio di Amazon SNS sono disponibili come feed RSS	Seguendo il titolo in questa pagina (Cronologia della documentazione), scegliere RSS.	22 giugno 2019

Glossario per AWS

Per la terminologia AWS più recente, consultare il glossario AWS nella documentazione di riferimento per Glossario AWS.

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.