aws

Guida per gli sviluppatori

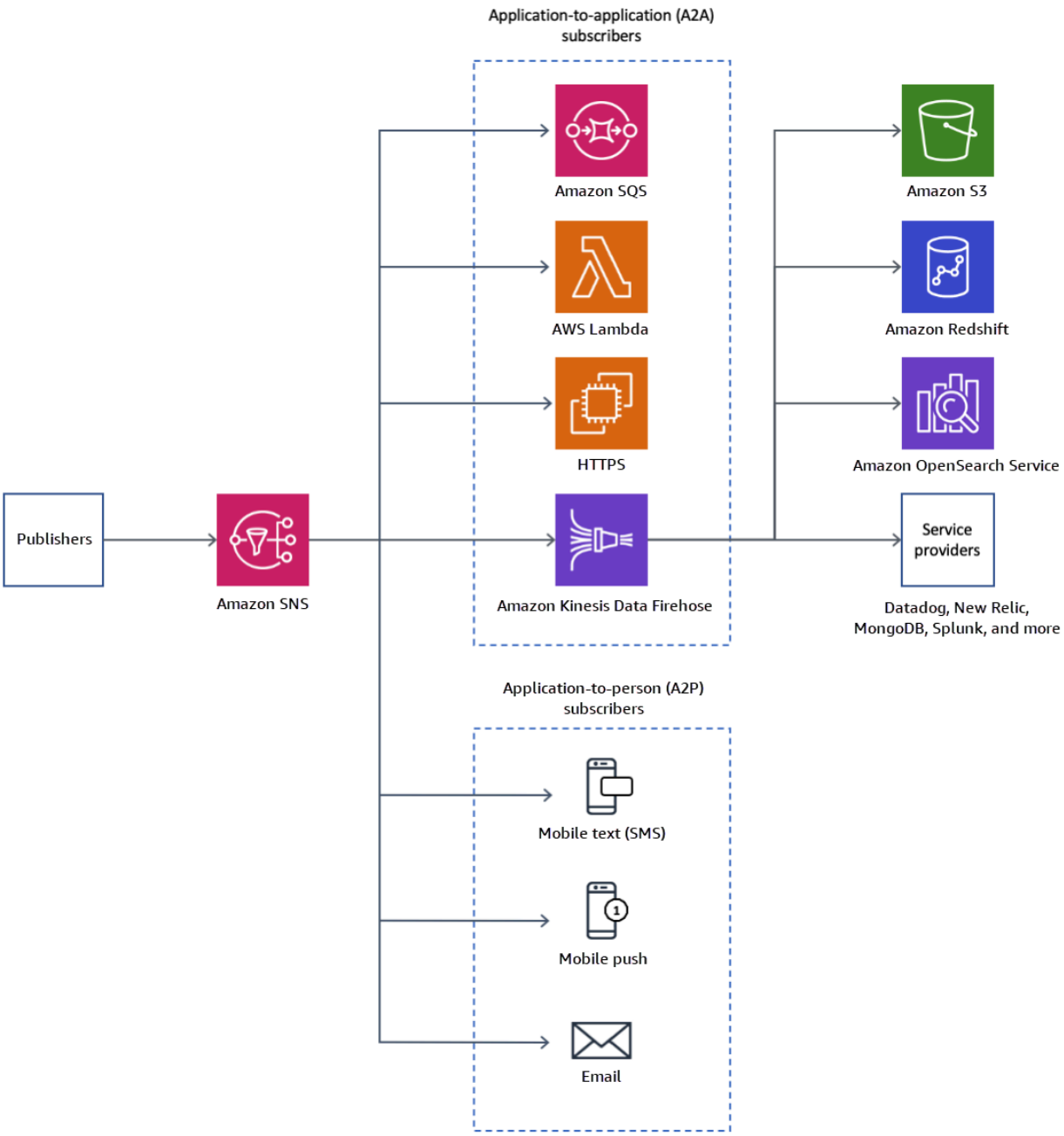
Amazon Simple Notification Service

Amazon Simple Notification Service: Guida per gli sviluppatori

Table of Contents

Che cos'è AmazonSNS?

Amazon Simple Notification Service (AmazonSNS) è un servizio gestito che fornisce il recapito dei messaggi dagli editori agli abbonati (noti anche come produttori e consumatori). 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'SNSargomento Amazon e ricevere messaggi pubblicati utilizzando un tipo di endpoint supportato, come Amazon Data Firehose, SQS Amazon AWS Lambda,, e-mailHTTP, notifiche push mobili e messaggi SMS di testo mobili ().



SNSCaratteristiche e funzionalità di Amazon

Amazon SNS offre un set completo di funzionalità progettate per migliorare la messaggistica tra applicazioni e utenti. Queste funzionalità consentono una comunicazione senza interruzioni, la

consegna sicura dei messaggi e una solida gestione dei messaggi, garantendo elevata disponibilità, durata e flessibilità per un'ampia gamma di casi d'uso della messaggistica.

- Application-to-application messaggistica

  Una pplication-to-application messaggistica supporta abbonati come i flussi di distribuzione di Amazon Data Firehose, le funzioni Lambda, le HTTP code SQS Amazon, gli endpoint /S e Event Fork Pipelines. AWS Ciò consente una consegna efficiente dei messaggi in architetture basate sugli eventi.

- Application-to-person notifiche

  pplication-to-personLe notifiche A forniscono notifiche agli utenti agli abbonati come applicazioni mobili, numeri di cellulare e indirizzi e-mail.

- Standard e argomenti FIFO

  FIFOgli argomenti garantiscono l'ordinamento, il raggruppamento e la deduplicazione rigorosi dei messaggi, consentendo FIFO la sottoscrizione di code standard per l'elaborazione dei messaggi. Gli argomenti standard vengono utilizzati quando l'ordinamento dei messaggi e la possibile duplicazione non sono fondamentali, e supportano tutti i protocolli di consegna per casi d'uso più ampi.

- Durabilità dei messaggi

  Amazon SNS utilizza una serie di strategie che collaborano 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 applica una politica di SNS 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 abbonando i flussi di distribuzione Firehose SNS agli argomenti, 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, SNS FIFO gli argomenti di Amazon supportano l'archiviazione e la riproduzione dei messaggi come archivio di messaggi locale senza codice che consente ai proprietari degli argomenti di archiviare (o 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 SNS messaggi su Amazon per argomenti FIFO.

- Attributi di messaggio

  Attributi SNS dei messaggi Amazonconsentono di fornire metadati arbitrari sul 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 SNS argomenti di Amazon, utilizzando le chiavi di crittografia fornite da. AWS KMS Per ulteriori informazioni, consulta the section called “Protezione dei dati con la crittografia lato server”.

  Puoi anche stabilire una connessione privata tra Amazon SNS e il tuo cloud privato virtuale (VPC). per ulteriori informazioni, consultathe section called “Protezione del traffico con endpoint VPC”.

AWS servizi comunemente usati con Amazon SNS

Puoi integrare Amazon SNS con diversi prodotti Servizi AWS per migliorare funzionalità e gestibilità. Questi servizi consentono una gestione ottimizzata dei messaggi, un controllo sicuro degli accessi, applicazioni basate sugli eventi e il provisioning automatico delle risorse.

- Amazon SQS offre una coda ospitata sicura, durevole e disponibile che consente di integrare e disaccoppiare sistemi e componenti software distribuiti. Amazon SQS è collegata ad Amazon SNS nei seguenti modi:
  - Amazon SNS fornisce code di lettere non recapitabili gestite da Amazon SQS per i messaggi non recapitabili.
  - Puoi iscrivere una SQS coda Amazon a un SNS argomento Amazon.

  - Puoi sottoscrivere una SQS FIFOcoda Amazon o una coda standard a un argomento Amazon SNS FIFO. Solo Amazon SQS FIFO Queues garantisce 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 SNS argomento.
- AWS Identity and Access Management (IAM) ti aiuta a controllare in modo sicuro l'accesso alle AWS risorse per i tuoi utenti. IAMUtilizzalo per controllare chi può usare i tuoi SNS argomenti Amazon (autenticazione), quali argomenti possono usare e come possono usarli (autorizzazione). Per ulteriori informazioni, consulta Utilizzo di politiche basate sull'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 SNS argomenti e abbonamenti Amazon. 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.

Accesso ad Amazon SNS

Puoi accedere e gestire Amazon SNS tramite la console o AWS CLI AWS SDKs, in base al tuo metodo di interazione preferito. La console offre un'interfaccia grafica per le attività di base, mentre SDKs fornisce funzionalità avanzate di configurazione e automazione per casi d'uso più complessi. AWS CLI

- La SNSconsole Amazon fornisce una comoda interfaccia utente per creare argomenti e abbonamenti, inviare e ricevere messaggi e monitorare eventi e registri.
- Il AWS Command Line Interface (AWS CLI) ti dà accesso diretto ad Amazon SNS API per casi d'uso avanzati di configurazione e automazione. Per ulteriori informazioni, consulta Usare Amazon SNS con AWS CLI.
- AWS fornisce SDKs in varie lingue. Per ulteriori informazioni, vedere SDKsand Toolkits.

Prezzi per Amazon SNS

Amazon SNS non prevede costi iniziali. Paghi in base al numero di messaggi pubblicati, al numero di notifiche che invii e a eventuali API chiamate aggiuntive per la gestione di argomenti e abbonamenti.

I prezzi di spedizione variano in base al tipo di endpoint. Puoi iniziare gratuitamente con il piano SNS gratuito di Amazon. Per informazioni, consulta la pagina SMS Prezzi mondiali.

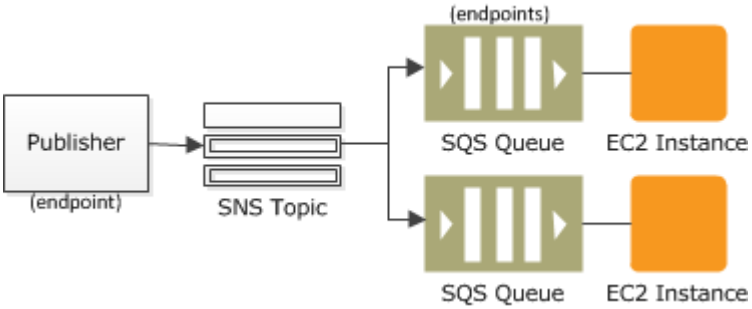
SNSScenari Amazon comuni

Utilizza questi SNS scenari Amazon comuni per implementare architetture scalabili e basate sugli eventi e garantire comunicazioni affidabili e in tempo reale tra applicazioni e utenti.

Integrazione di applicazioni

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

Ad esempio, è possibile sviluppare un'applicazione che pubblica un messaggio su un argomento ogni volta che viene effettuato un ordine per un SNS prodotto. Quindi, le SQS code iscritte all'SNSargomento ricevono notifiche identiche per il nuovo ordine. Un'istanza del server Amazon Elastic Compute Cloud (AmazonEC2) collegata a una delle SQS code può gestire l'elaborazione o l'evasione dell'ordine. E puoi collegare un'altra istanza EC2 del server Amazon 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. Ampliando l'esempio precedente, puoi iscrivere un'altra coda sullo stesso argomento per i nuovi ordini in arrivo. SQS SNS Quindi, collegando questa nuova SQS coda all'ambiente di test, potete continuare a migliorare e testare l'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
- Fanout SNS delle notifiche Amazon alle funzioni Lambda per l'elaborazione automatizzata
- Fanout SNS delle notifiche Amazon alle SQS code Amazon per l'elaborazione asincrona
- Fanout SNS delle notifiche Amazon agli endpoint HTTPS
- Elaborazione basata sugli eventi con Amazon SNS e AWS 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 queste notifiche a utenti specifici tramite SMS e-mail. Ad esempio, puoi ricevere una notifica immediata quando si verifica un evento, come 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 SNS delle notifiche Amazon nella Amazon CloudWatch User Guide.

Notifiche all'utente

Amazon SNS può inviare messaggi e-mail push e messaggi di testo (SMSmessaggi) a singoli o gruppi. Ad esempio, è possibile inviare conferme di ordine e-commerce come notifiche utente. Per ulteriori informazioni sull'utilizzo di Amazon SNS per inviare SMS messaggi, consultaMessaggi di testo mobili con Amazon SNS.

Notifiche push per dispositivi mobili

Le notifiche push per dispositivi mobili ti permettono di inviare messaggi di notifica direttamente alle app su dispositivi mobili. Ad 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 ulteriori informazioni sull'utilizzo di Amazon SNS per inviare messaggi di notifica push, consultaInvio di notifiche push per dispositivi mobili con Amazon SNS.

Usare Amazon SNS con un AWS SDK

AWS i kit di sviluppo software (SDKs) sono disponibili per molti linguaggi di programmazione popolari. Ciascuno di essi SDK fornisceAPI, esempi di codice e documentazione che semplificano agli sviluppatori la creazione di applicazioni nel linguaggio preferito.

SDKdocumentazione	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 AmazonSNS, consultaEsempi di codice per l'SNSutilizzo di Amazon AWS SDKs.

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.

Crea un SNS argomento Amazon e pubblica messaggi

Questo argomento fornisce i passaggi fondamentali per la gestione SNS delle risorse Amazon, concentrandosi in particolare su argomenti, abbonamenti e pubblicazione di messaggi. Innanzitutto, configurerai le autorizzazioni di accesso necessarie per AmazonSNS, assicurandoti di disporre delle autorizzazioni corrette per creare e gestire le risorse AmazonSNS. Successivamente, creerai un nuovo SNS argomento Amazon, che funge da hub centrale per la gestione e la consegna dei messaggi agli abbonati. Dopo aver creato l'argomento, procederai alla creazione di un abbonamento a questo argomento, che consentirà a endpoint specifici di ricevere i messaggi pubblicati su di esso.

Una volta definiti l'argomento e l'abbonamento, pubblicherai un messaggio sull'argomento, osservando in che SNS modo Amazon recapita in modo efficiente il messaggio a tutti gli endpoint sottoscritti. Infine, imparerai come eliminare sia l'abbonamento che l'argomento, completando il ciclo di vita delle SNS risorse Amazon che hai gestito. Questo approccio ti offre una chiara comprensione delle operazioni fondamentali di AmazonSNS, fornendoti le competenze pratiche necessarie per gestire i flussi di lavoro di messaggistica utilizzando la console AmazonSNS.

Configurazione dell'accesso per Amazon SNS

Prima di poter utilizzare Amazon SNS per la prima volta, devi completare i seguenti passaggi.

Argomenti

Crea un utente Account AWS e un IAM utente

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.

Registrati 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 https://portal.aws.amazon.com/billing/registrazione.

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 best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a https://aws.amazon.com/e scegliendo 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 utilizzare l'utente root 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. Attiva l'autenticazione a più fattori (MFA) per il tuo utente root.

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

Crea un utente con accesso amministrativo

1. Abilita IAM Identity Center.

   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 i valori predefiniti IAM Identity Center directory nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

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

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

Assegna l'accesso a ulteriori utenti

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

   Segui le istruzioni riportate nella pagina Creazione di un set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

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

Passaggi successivi

Ora che sei pronto a lavorare con AmazonSNS, inizia con:

1. Creare un SNS argomento Amazon
2. Creare un abbonamento a un SNS argomento di Amazon
3. Pubblicazione di un SNS messaggio Amazon
4. Eliminazione di un SNS argomento e di un abbonamento Amazon

Creare un SNS argomento Amazon

Un SNS argomento di Amazon è un punto di accesso logico che funge da canale di comunicazione. Un argomento consente di raggruppare più endpoint (ad esempio Amazon AWS Lambda SQS, 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 e più comune SNS attività di Amazon è la creazione di un argomento. Questa pagina mostra come utilizzare il AWS Management Console AWS SDK for Java, il e il AWS SDK for .NET per creare un argomento.

Durante la creazione, scegli un tipo di argomento (standard oFIFO) e assegna 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 aggiungete informazioni di identificazione personale (PII) o altre informazioni riservate o sensibili nei nomi degli 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

La creazione di un argomento in Amazon SNS pone le basi per la distribuzione dei messaggi, consentendoti di pubblicare messaggi che possono essere distribuiti a più abbonati. Questo passaggio è essenziale per configurare il tipo, le impostazioni di crittografia e le politiche di accesso dell'argomento, in modo da garantire che l'argomento soddisfi i requisiti di sicurezza, conformità e operativi dell'organizzazione.

1. Accedi alla SNSconsole Amazon.
2. Esegui una di queste operazioni:

   - Se non è mai stato creato alcun argomento sotto il tuo Account AWS profilo in precedenza, leggi la descrizione di SNS Amazon nella home page.

   - Se in Account AWS precedenza sono stati creati argomenti sotto il tuo 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 Tipo, scegli un tipo di argomento (Standard o FIFO).

   b. Immetti un nome per l'argomento. Per un FIFOargomento, aggiungete .fifo alla fine del nome.

   c. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.

      > **Important**
      >
      > Quando ci si iscrive a un endpoint di posta elettronica, il numero di caratteri combinato per il nome visualizzato dell'SNSargomento Amazon e l'indirizzo e-mail di invio (ad esempio, no-reply@sns.amazonaws.com) non deve superare i UTF 320-8 caratteri. Puoi utilizzare uno strumento di codifica di terze parti per verificare la lunghezza dell'indirizzo di invio prima di configurare un nome visualizzato per il tuo argomento AmazonSNS.

   d. (Facoltativo) Per un FIFO argomento, puoi scegliere la deduplicazione dei messaggi basata sul contenuto per abilitare la deduplicazione predefinita dei messaggi. Per ulteriori informazioni, consulta Deduplicazione dei SNS messaggi Amazon per argomenti FIFO.
5. (Facoltativo) Espandere la sezione Encryption (Crittografia) e procedere come segue. Per ulteriori informazioni, consulta Protezione dei SNS dati Amazon con la crittografia lato server.

   a. Scegliere Enable encryption (Abilita crittografia).

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

      Per ogni KMS tipo, KMSARNvengono visualizzati la Descrizione, l'Account e.

> **Important**
>
> Se non sei il proprietario diKMS, o se accedi con un account che non dispone delle `kms:DescribeKey` autorizzazioni `kms:ListAliases` e, non potrai visualizzare le informazioni in merito KMS sulla SNS console Amazon.
> Chiedi al proprietario di KMS concederti queste autorizzazioni. Per ulteriori informazioni, consulta la sezione AWS KMS APIAutorizzazioni: azioni e risorse di riferimento nella Guida per gli AWS Key Management Service sviluppatori.

- L'opzione AWS managed KMS for Amazon SNS (impostazione predefinita) alias/aws/sns è selezionata per impostazione predefinita.

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

- Per utilizzare una personalizzazione KMS del tuo AWS account, scegli il campo KMSchiave, quindi scegli la personalizzazione KMS dall'elenco.

  > **Note**
  >
  > Per istruzioni sulla creazione di chiavi personalizzateKMSs, consulta Creating Keys nella AWS Key Management Service Developer Guide

- Per utilizzare un codice personalizzato KMS ARN del tuo AWS account o di un altro AWS account, inseriscilo nel campo KMSchiave.

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 Gestione delle

identità e degli accessi in Amazon SNS e Casi di esempio per il controllo degli SNS accessi di Amazon.

> **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 in che modo Amazon SNS ritenta i tentativi di recapito dei messaggi non riusciti, espandi la sezione Delivery retry policy (HTTP/S). Per ulteriori informazioni, consulta Tentativi di recapito dei SNS messaggi Amazon.
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 SNS messaggi Amazon.
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 Etichettatura degli SNS argomenti di Amazon.
10. Scegliere Create topic (Crea argomento).

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

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

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

Per creare un argomento utilizzando un AWS SDK

Per utilizzare un AWS SDK, è necessario configurarlo con le proprie credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

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 specifici FIFO e di deduplicazione.

```
    /// <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 API i dettagli, vedere CreateTopicin AWS SDK for .NET API Reference.

C++

SDKper 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 API i dettagli, vedi CreateTopic AWS SDK for C++APIReference.

CLI

AWS CLI

Per creare un argomento SNS

L'`create-topic`esempio seguente crea un SNS argomento 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 AWS Command Line Interface User Guide.

- Per API i dettagli, consulta CreateTopic AWS CLICommand Reference.

Go

SDKper 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
}



// 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(ctx context.Context, 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(ctx, &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 API i dettagli, vedi CreateTopic AWS SDK for GoAPIReference.

Java

SDKper 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 API i dettagli, vedi CreateTopic AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere CreateTopicin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento CreateTopic AWSSDKa Kotlin API.

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 API i dettagli, vedi CreateTopic AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere CreateTopicPython (Boto3) Reference.AWS SDK API

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi CreateTopic AWS SDK for RubyAPIReference.

Rust

SDKper Rust

> **Note**
>
> c'è altro da fare 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 API i dettagli, CreateTopicconsulta AWS SDKRust API Reference.

SAP ABAP

SDKper SAP ABAP

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

```
   TRY.
       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 API i dettagli, vedi CreateTopicSAPABAPAPIcome riferimento.AWS SDK

Creare un abbonamento a un SNS argomento di Amazon

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**
>
> HTTPGli endpoint (S), gli indirizzi e-mail e AWS le risorse di altro tipo Account AWS richiedono la conferma dell'abbonamento prima di poter ricevere messaggi.

Per sottoscrivere un endpoint a un argomento Amazon SNS

La sottoscrizione di un endpoint a un SNS argomento Amazon consente il recapito dei messaggi all'endpoint specificato, garantendo che i sistemi o gli utenti giusti ricevano notifiche quando

un messaggio viene pubblicato sull'argomento. Questo passaggio è essenziale per collegare l'argomento ai consumatori, siano essi applicazioni, destinatari di posta elettronica o altri servizi, e consente una comunicazione senza interruzioni tra i sistemi.

1. Accedi alla SNSconsole Amazon.
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 AWS ARN quello che è stato generato quando hai creato l'SNSargomento Amazon, 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
      - E-mail/E-mail- JSON
      - Amazon Data Firehose
      - Amazon SQS

      > **Note**
      >
      > Per iscriverti a un SNSFIFOargomento, scegli questa opzione.

      - AWS Lambda
      - Endpoint applicazione piattaforma
      - SMS

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

   d. Solo endpoint Firehose: per il ruolo Subscription ARN, specifica il IAM ruolo che hai creato per la scrittura nei flussi ARN di distribuzione Firehose. Per ulteriori informazioni, consulta Prerequisiti per sottoscrivere i flussi di distribuzione di Firehose agli argomenti di Amazon SNS.

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

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

g. (Facoltativo) Per abilitare il filtro basato sul payload, imposta `Filter Policy Scope` su `MessageBody`. Per ulteriori informazioni, consulta Ambito della politica di filtro degli SNS abbonamenti Amazon.

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

i. Scegli Create Subscription (Crea sottoscrizione).

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

Pubblicazione di un SNS messaggio Amazon

Dopo aver creato un SNS argomento Amazon e aver sottoscritto un endpoint, puoi pubblicare messaggi sull'argomento. Quando viene pubblicato un messaggio, Amazon SNS tenta di recapitarlo agli endpoint sottoscritti.

Argomenti

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

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

1. Accedi alla SNSconsole Amazon.
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 FIFOargomento, inserisci un ID del gruppo di messaggi. I messaggi nello stesso gruppo di messaggi vengono recapitati nell'ordine in cui vengono pubblicati.

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

   d. (Facoltativo) Per le notifiche push su dispositivi mobili, inserisci un valore Time to Live (TTL) in secondi. Si tratta del periodo di tempo che un servizio di notifica push, come Apple Push Notification Service (APNs) o Firebase Cloud Messaging (FCM), ha a disposizione per recapitare 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. Scegli Payload personalizzato per ogni protocollo di consegna, quindi inserisci un JSON oggetto per definire il messaggio da inviare per ogni protocollo di consegna.

      Per ulteriori informazioni, consulta Pubblicazione di SNS notifiche Amazon con payload specifici della piattaforma.

6. Nella sezione Attributi del messaggio, aggiungi tutti gli attributi che desideri che Amazon abbini SNS all'`FilterPolicy`attributo subscription 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 message in base alla politica di filtro di un abbonamento (se presente) prima di inviare il messaggio all'abbonamento su cui l'ambito della politica di filtro non è impostato in modo esplicito. `MessageBody`

Per ulteriori informazioni, consulta Attributi SNS dei messaggi Amazon.

7. Seleziona Publish message (Pubblica messaggio).

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

Per pubblicare un messaggio su un argomento utilizzando un AWS SDK

Per utilizzare un AWS SDK, è necessario configurarlo con le proprie credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

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

.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 API i dettagli, consulta Publish in AWS SDK for .NET APIReference.

C++

SDKper 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.

```
//! 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 API i dettagli, consulta Publish in AWS SDK for C++ APIReference.

CLI

AWS CLI

Esempio 1: pubblicazione di un messaggio in un argomento

L'`publish`esempio seguente pubblica il messaggio specificato SNS sull'argomento 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 SMS messaggio 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 API i dettagli, consulta Pubblica in AWS CLI Command Reference.

Go

SDKper 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(ctx context.Context, 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(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Per API i dettagli, consulta Publish in AWS SDK for Go APIReference.

Java

SDKper Java 2.x

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

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

```
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class 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 API i dettagli, consulta Publish in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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;
    let deduplicationId;
```

```
    let 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 API i dettagli, consulta Publish in AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Publish in AWS SDKfor Kotlin reference API.

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;


/**
 * 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 API i dettagli, consulta Publish in AWS SDK for PHP APIReference.

PowerShell

Strumenti 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 API i dettagli, vedere Publish in AWS Tools for PowerShell Cmdlet Reference.

Python

SDKper Python (Boto3)

> **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.

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 API i dettagli, consulta Publish in AWS SDKfor Python (Boto3) Reference. API

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Publish in AWS SDK for Ruby APIReference.

Rust

SDKper Rust

> **Note**
>
> c'è altro da fare 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 API i dettagli, consulta Publish in AWS SDKfor Rust API reference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, consulta Pubblica AWS SDKin SAP ABAP API come riferimento.

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

Per pubblicare SNS messaggi Amazon di grandi dimensioni, puoi utilizzare Amazon SNS Extended Client Library per Java o Amazon SNS Extended Client Library for 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 archiviato nell'argomento Amazon S3. SNS Le SQS code Amazon abbonate possono utilizzare Amazon SQS Extended Client Library for Java per dereferenziare 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 sia con gli standard che con gli FIFO argomenti.

Argomenti

Libreria client SNS estesa Amazon per Java

Argomenti

Prerequisiti

Di seguito sono riportati i prerequisiti per l'utilizzo di Amazon SNS Extended Client Library for Java:

- Un AWS SDK.

L'esempio in questa pagina utilizza AWS JavaSDK. Per installare e configurareSDK, consulta la sezione Configurazione AWS SDK per Java nella Guida per gli AWS SDK for Java 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.
- Amazon SNS Extended Client Library 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. Per esempio:

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

- KMSChiave 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 archiviato 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. L'Amazon SQS Extended Client viene utilizzato per ricevere il messaggio.

SDKper 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, usa Amazon SNS Extended Client Library for 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

Entrambe le SQS librerie Amazon SNS e Amazon utilizzano la Payload Offloading Java Common Library per AWS archiviare e recuperare i payload dei messaggi con Amazon S3. Qualsiasi endpoint

abilitato a Java (ad esempio, un HTTPS endpoint implementato in Java) può utilizzare la stessa libreria per eliminare 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 SNS estesa Amazon per Python

Argomenti

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

Prerequisiti

Di seguito sono riportati i prerequisiti per l'utilizzo di Amazon SNS Extended Client Library for Python:

- Un. AWS SDK

  L'esempio in questa pagina utilizza AWS Python SDK Boto3. Per installare e configurareSDK, consulta la documentazione AWS SDKdi 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 informazioni sulle credenziali, consulta Credentials nella AWS SDKfor Python Developer Guide.
- Python 3.x (o versioni successive) e pip.
- La Amazon SNS Extended Client Library per Python (disponibile anche in PyPI).

Configurazione dello storage dei messaggi

Gli attributi seguenti sono disponibili su Boto3 Amazon SNS Client, Topic e PlatformEndpointObjects per configurare le opzioni di archiviazione 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 il payload archiviato in Amazon S3

L'esempio di codice seguente mostra come:

- Crea un SNS argomento Amazon di esempio e una SQS coda Amazon.
- 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, usa Amazon SNS Extended Client Library for 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 SNS dei messaggi Amazon

Amazon SNS supporta la consegna degli attributi dei messaggi, che consentono di fornire elementi di metadati strutturati (come timestamp, dati geospaziali, firme e identificatori) sul messaggio. Per SQS gli abbonamenti, è possibile inviare un massimo di 10 attributi del messaggio quando è abilitata la consegna di messaggi non elaborati. Per inviare più di 10 attributi di messaggio, è necessario disabilitare Consegna di messaggi non elaborati. I messaggi con più di 10 attributi di messaggio indirizzati verso SQS abbonamenti Amazon abilitati per Raw Message Delivery 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 informazioni sull'invio di messaggi con attributi utilizzando AWS Management Console o il AWS SDK for Java, consulta il tutorial. Per pubblicare messaggi su SNS argomenti di Amazon utilizzando il AWS Management Console

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

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 consegnati all'endpoint come avviene quando si inviano messaggi con attributi di messaggio agli SQS endpoint Amazon.

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 SNS dei messaggi Amazon.

> **Note**
>
> Quando gli attributi dei messaggi vengono utilizzati per il filtraggio, il valore deve essere una stringa valida. JSON 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. 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'azione Pubblica in Amazon Simple Notification Service API Reference.

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 del messaggio identificano il modo in cui i valori degli attributi del messaggio vengono gestiti da AmazonSNS. Ad esempio, se il tipo è un numero, Amazon SNS conferma che si tratta di un numero.

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

- Stringa: le stringhe sono Unicode con codifica binaria UTF -8. Per un elenco dei valori di codice, consultate http://en.wikipedia. org/wiki/ASCII# ASCII _caratteri_stampabili.

  > **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 gli abbonamenti. AWS Lambda Se specifichi questo tipo di dati per gli endpoint Lambda, viene passato come tipo di `String` dati nel JSON payload 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 gli abbonamenti. AWS Lambda Se specifichi questo tipo di dati per gli endpoint Lambda, viene passato come tipo di `String` dati nel JSON payload 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`
APNs[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`
	`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`

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

[1] Apple rifiuterà SNS le notifiche di Amazon se gli attributi dei messaggi non soddisfano i requisiti. Per ulteriori dettagli, consulta Invio di richieste di notifica a APNs sul sito Web per sviluppatori Apple.

Raggruppamento di SNS messaggi Amazon

Cos'è il batch di messaggi?

Un'alternativa alla pubblicazione di messaggi su Standard o FIFO topic in singole `Publish` API richieste consiste nell'utilizzare Amazon SNS `PublishBatch` API per pubblicare fino a 10 messaggi in una singola API richiesta. L'invio di messaggi in batch può aiutarti a ridurre fino a 10 volte i costi associati alla connessione di applicazioni distribuite (messaggistica A2A) o all'invio di notifiche alle persone (messaggistica A2P) con AmazonSNS. Amazon SNS impone delle quote sul numero di messaggi che puoi pubblicare su un argomento al secondo in base alla regione in cui operi. Consulta la pagina SNSdegli endpoint e delle quote di Amazon nella Riferimenti generali di AWSguida per ulteriori informazioni sulle API quote.

> **Note**
>
> La dimensione aggregata totale di tutti i messaggi che invii in una singola `PublishBatch` API richiesta non può superare i 262.144 byte (256 KB).
> `PublishBatch`APIUtilizza la stessa azione per le politiche. `Publish` API IAM

Come funziona il batch di messaggi?

La pubblicazione di messaggi con `PublishBatch` API è simile alla pubblicazione di messaggi con `Publish`API. La differenza principale è che a ogni messaggio all'interno di una `PublishBatch` API richiesta deve essere assegnato un ID batch univoco (fino a 80 caratteri). In questo modo, Amazon SNS può restituire API risposte 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 FIFO argomenti, oltre a includere l'assegnazione di un ID batch univoco, devi comunque includere un `MessageDeduplicationID` e `MessageGroupId` per ogni singolo messaggio.

Esempi

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 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 FIFO argomento

```
// 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);
    }
}
```

Eliminazione di un SNS argomento e di un abbonamento Amazon

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 editore tenta di pubblicare un messaggio sull'argomento eliminato, 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. Amazon elimina SNS automaticamente gli abbonamenti non confermati dopo 48 ore.

Argomenti

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

- Per eliminare un abbonamento e un argomento utilizzando un AWS SDK

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

L'eliminazione di un SNS argomento o di un abbonamento Amazon garantisce una gestione efficiente delle risorse, previene l'utilizzo non necessario e mantiene organizzata la SNS console Amazon. Questo passaggio consente di evitare i potenziali costi derivanti dalle risorse inattive e semplifica l'amministrazione rimuovendo argomenti o abbonamenti che non sono più necessari.

Per eliminare un argomento utilizzando il AWS Management Console

1. Accedi alla SNSconsole Amazon.
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 SNSconsole Amazon.
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 un abbonamento e un argomento utilizzando un AWS SDK

Per usare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

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 per argomentoARN.

```
/// <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 API i dettagli, DeleteTopicconsulta AWS SDK for .NET APIReference.

C++

SDKper 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 API i dettagli, vedi DeleteTopic AWS SDK for C++APIReference.

CLI

AWS CLI

Per eliminare un SNS argomento

L'`delete-topic`esempio seguente elimina l'SNSargomento specificato.

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

Questo comando non produce alcun output.

- Per API i dettagli, vedere DeleteTopicin AWS CLI Command Reference.

Go

SDKper 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
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &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 API i dettagli, vedi DeleteTopic AWS SDK for GoAPIReference.

Java

SDKper 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 API i dettagli, vedi DeleteTopic AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere DeleteTopicin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento DeleteTopic AWSSDKa Kotlin API.

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 API i dettagli, vedi DeleteTopic AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere DeleteTopicPython (Boto3) Reference.AWS SDK API

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi DeleteTopicSAPABAPAPIcome riferimento.AWS SDK

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.
- Abilita la crittografia lato server per un argomento con una coda crittografata di Amazon Simple Queue Service (AmazonSQS) sottoscritta.
- Sottoscrivere AWS Pipeline Event Fork a un argomento.

Strategie di ordinamento e deduplicazione dei messaggi utilizzando argomenti di Amazon SNS FIFO

Questo argomento fornisce informazioni sulle caratteristiche e le funzionalità degli argomenti di Amazon SNS FIFO (First-In-First-Out) e su come si integrano con le code Amazon. SQS FIFO Imparerai come utilizzare questi servizi insieme per garantire l'ordinamento e la deduplicazione rigorosi dei messaggi, essenziali per le applicazioni che richiedono la coerenza dei dati. Questo contenuto tratta i casi d'uso specifici in cui SNS FIFO gli argomenti di Amazon sono utili, fornendo informazioni sugli scenari in cui l'ordine e l'unicità dei messaggi sono fondamentali.

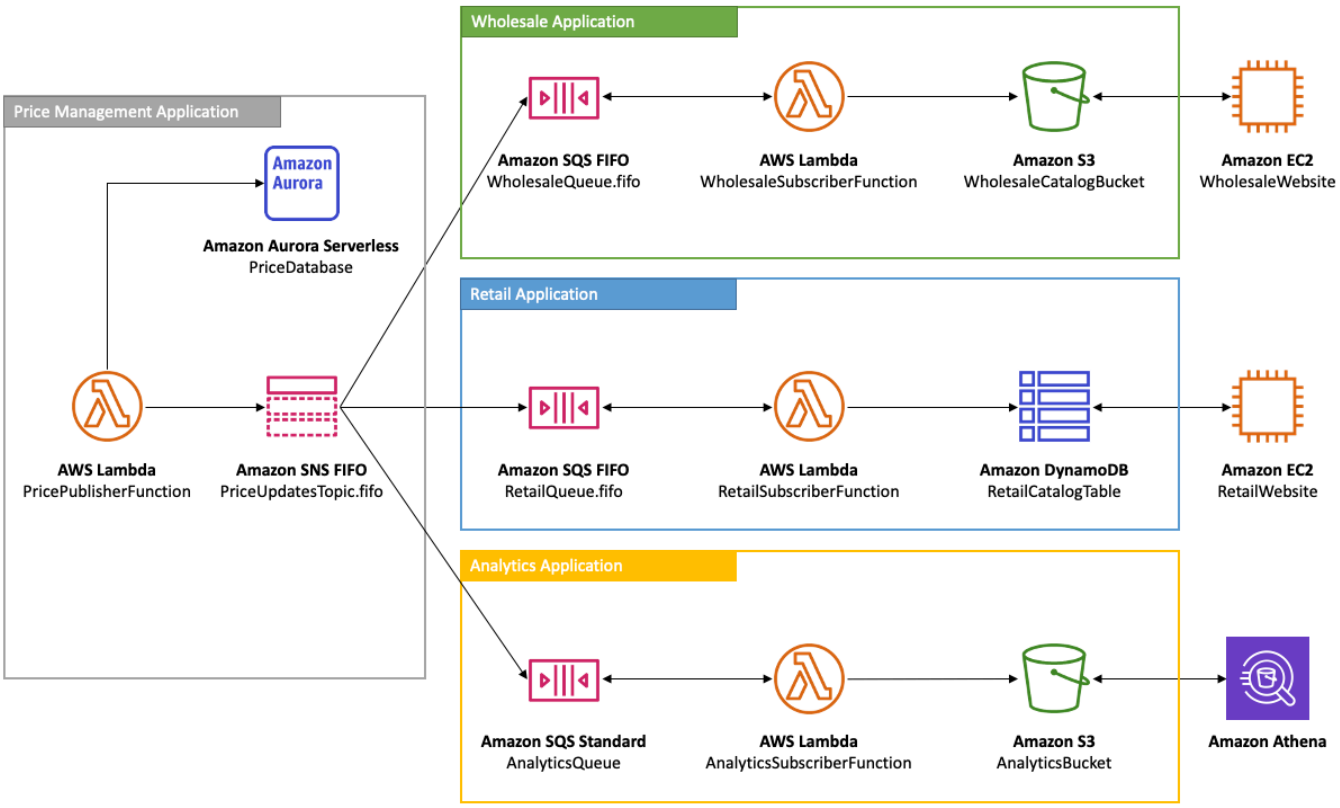
Scoprirai anche i dettagli tecnici dell'ordine e del raggruppamento dei messaggi e come questi influiscono sulla consegna dei messaggi. L'argomento sulla deduplicazione dei messaggi spiega i meccanismi che impediscono la duplicazione dei messaggi, garantendo che ogni messaggio venga elaborato una sola volta. Inoltre, imparerai a utilizzare il filtro, la sicurezza e la durabilità dei messaggi, elementi importanti per mantenere l'integrità e l'affidabilità del tuo sistema di messaggistica. Questo contenuto include anche informazioni sull'archiviazione e la riproduzione dei messaggi, che offrono strategie per la gestione della cronologia dei messaggi. Vengono inoltre forniti esempi pratici di codice per aiutarti a implementare queste funzionalità nelle tue applicazioni, offrendoti un'esperienza pratica con SNS FIFO gli argomenti di Amazon e la loro integrazione con Amazon SQS FIFO Queues.

Esempio di utilizzo di un SNS FIFO argomento Amazon

L'esempio seguente descrive una piattaforma di e-commerce creata da un produttore di ricambi auto utilizzando SNS FIFO argomenti Amazon e SQS code Amazon. 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 AWS Lambda funzione che pubblica gli aggiornamenti dei prezzi su un SNS FIFO argomento Amazon 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 ricevere notifiche sulle variazioni di prezzo, l'applicazione di vendita all'ingrosso sottoscrive la propria SQS FIFO coda Amazon all'argomento Amazon dell'applicazione di gestione dei prezzi. SNS FIFO

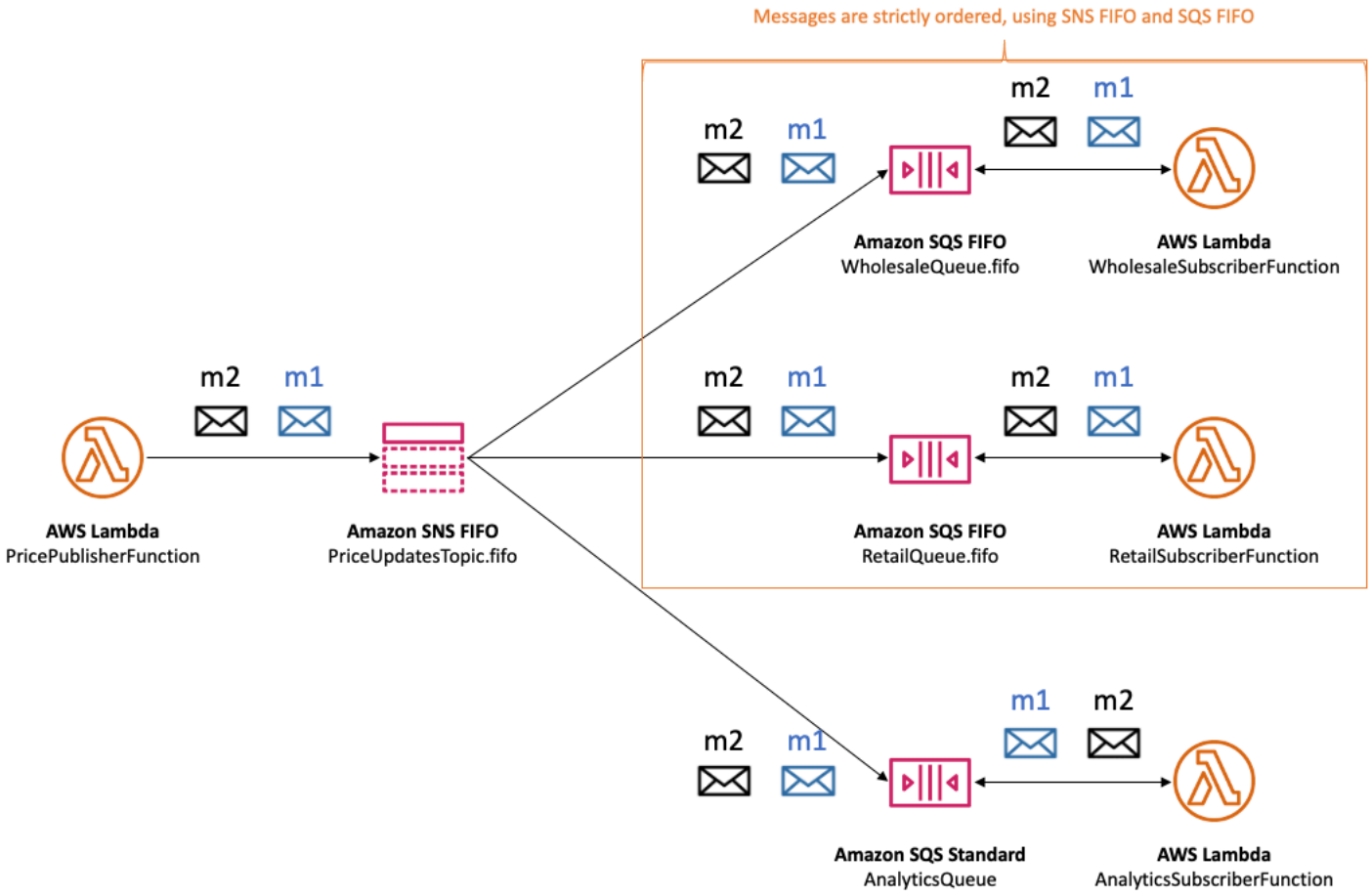
- Un'applicazione di vendita al dettaglio fornisce il backend per un altro sito web in cui i proprietari di auto e gli appassionati di tuning auto possono acquistare singoli pezzi di auto per i loro veicoli. Per ricevere notifiche sulle variazioni di prezzo, l'applicazione di vendita al dettaglio iscrive anche la sua SQS FIFO coda Amazon all'argomento Amazon dell'applicazione di gestione dei prezzi. SNS FIFO
- 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 ricevere notifiche sulle variazioni di prezzo, l'applicazione di analisi sottoscrive la propria coda SQS standard Amazon all'argomento Amazon SNS FIFO 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 SNS FIFO degli argomenti e delle SQS FIFO code Amazon consente l'elaborazione dei messaggi in ordine e senza duplicazioni. Per ulteriori informazioni, consulta Dettagli sull'ordinazione dei SNS messaggi Amazon per argomenti FIFO. Per gli snippet di codice che implementano questo caso d'uso, vedere Esempi di SNS codice Amazon per FIFO argomenti.

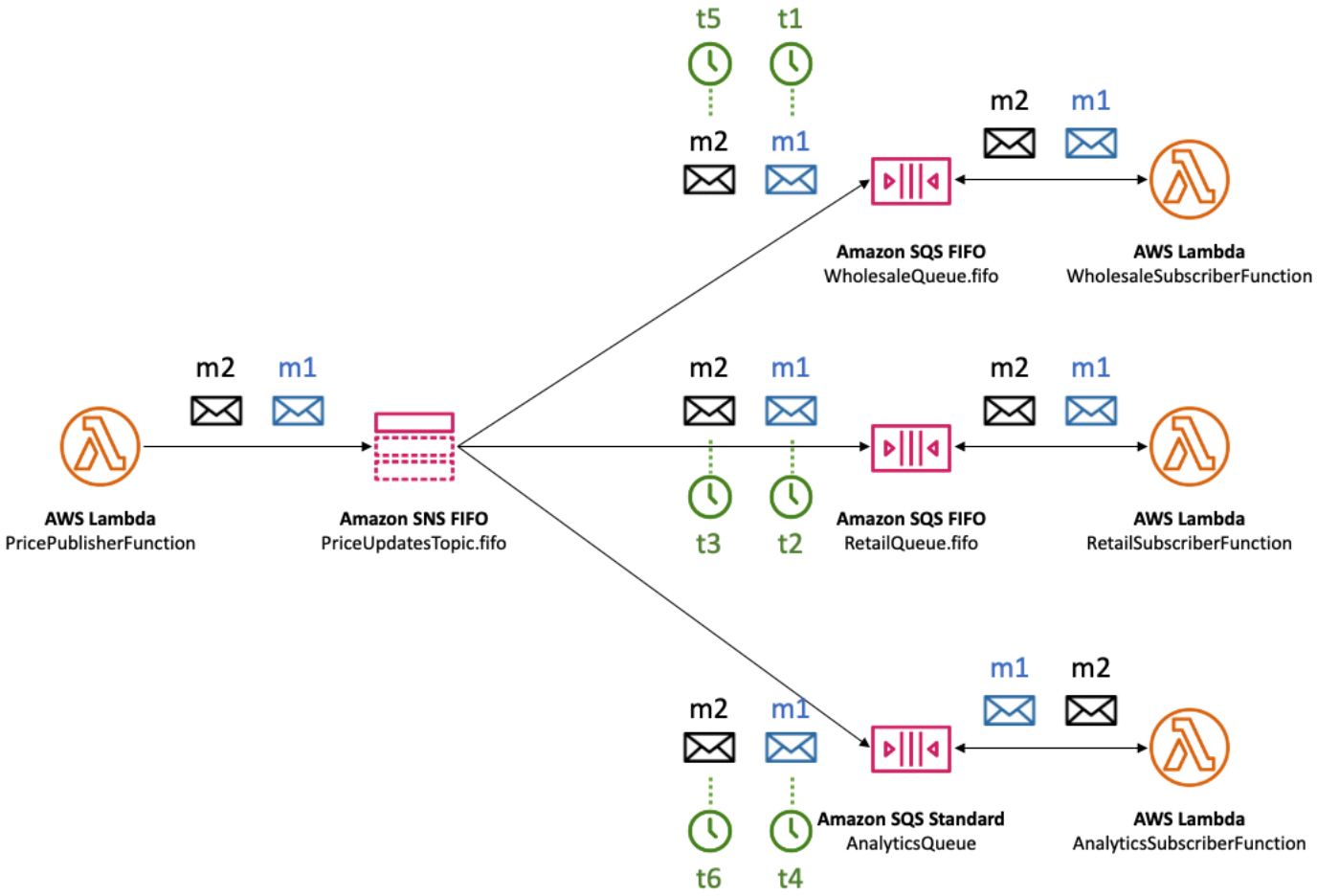
Dettagli sull'ordinazione dei SNS messaggi Amazon per argomenti FIFO

Un SNS FIFO argomento Amazon recapita sempre i messaggi alle SQS code Amazon sottoscritte nell'ordine esatto in cui i messaggi vengono pubblicati sull'argomento e solo una volta. Con un abbonamento Amazon SQS FIFO Queue, l'utente della coda riceve i messaggi nell'ordine esatto in cui i messaggi vengono consegnati alla coda e senza duplicati. Con un abbonamento Amazon SQS Standard Queue, tuttavia, l'utente della coda può ricevere messaggi non corretti 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 utilizzo di un SNS FIFO argomento Amazon.

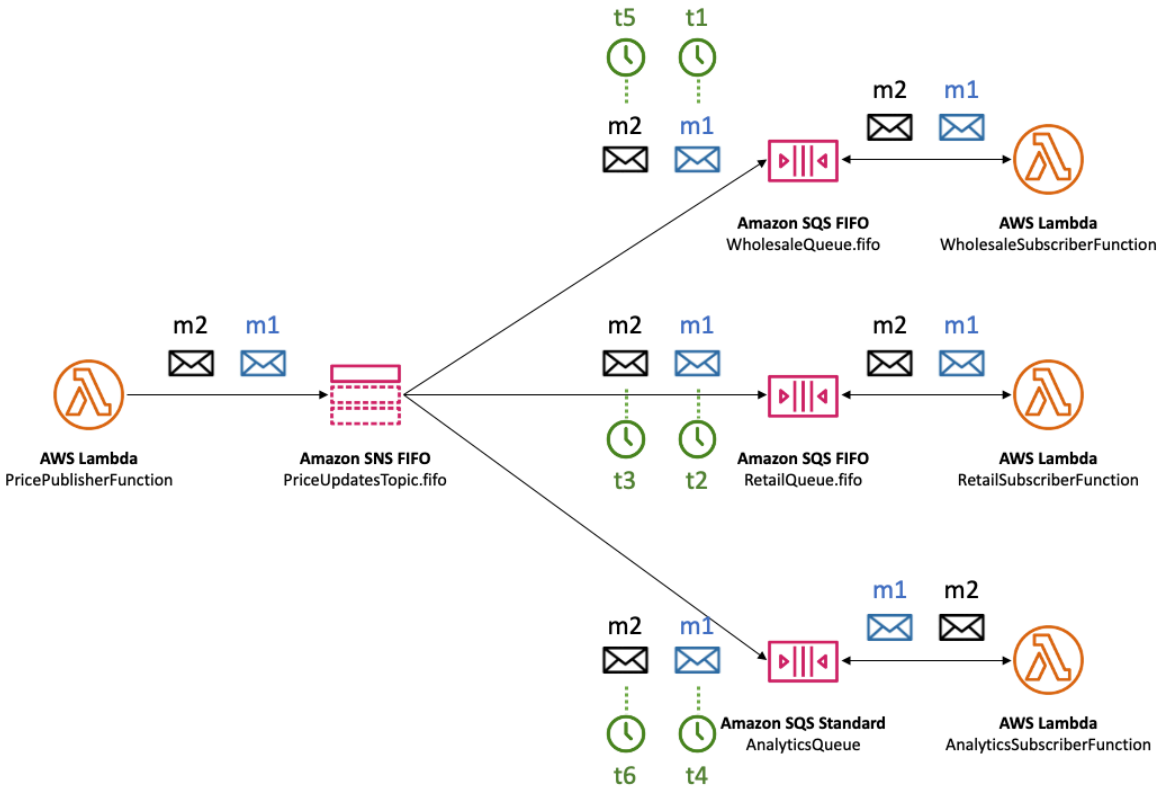


Si noti che non vi è alcun ordinamento implicito degli abbonati. L'esempio seguente ne è un'illustrazione m1 viene consegnato prima all'abbonato all'ingrosso, poi all'abbonato al dettaglio e quindi all'abbonato di analisi. Messaggio m2 viene consegnato prima all'abbonato al dettaglio, poi all'abbonato all'ingrosso e infine all'abbonato di analisi. Sebbene i due messaggi vengano recapitati

agli abbonati in un ordine diverso, l'ordine dei messaggi viene mantenuto per ogni abbonato Amazon SQSFIFO. Ogni abbonato è percepito in isolamento da qualsiasi altro abbonato.

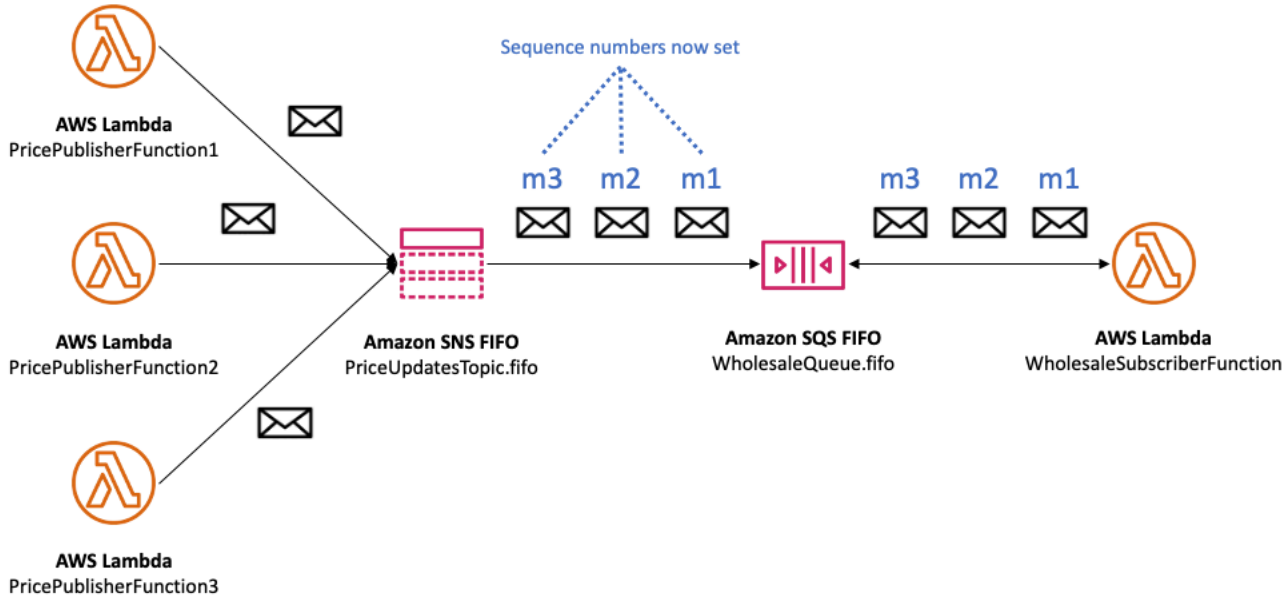


Se un abbonato Amazon SQS Queue diventa irraggiungibile, può perdere la sincronizzazione. Ad esempio, supponiamo che il proprietario della coda delle applicazioni all'ingrosso modifichi erroneamente la politica di SQS coda di Amazon in modo da impedire al responsabile del SNS servizio Amazon di recapitare i 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 di vendita all'ingrosso corregge la sua politica di coda, Amazon SNS riprende a recapitare i messaggi alla coda degli abbonati. 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 avere più applicazioni (o più thread all'interno della stessa applicazione) che pubblicano messaggi su un SNS FIFO argomento in parallelo. In questo modo, deleghi efficacemente il sequenziamento dei messaggi al servizio AmazonSNS. Per determinare la sequenza stabilita di messaggi, è possibile controllare il numero di sequenza.

Il numero di sequenza è un numero grande e 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 SQS code Amazon sottoscritte come parte del corpo del messaggio. Tuttavia, se abiliti il recapito di messaggi non elaborati, il messaggio che viene recapitato alla SQS coda di Amazon non include il numero di sequenza o altri metadati dei SNS messaggi Amazon.



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

Raggruppamento di SNS messaggi Amazon per argomenti FIFO

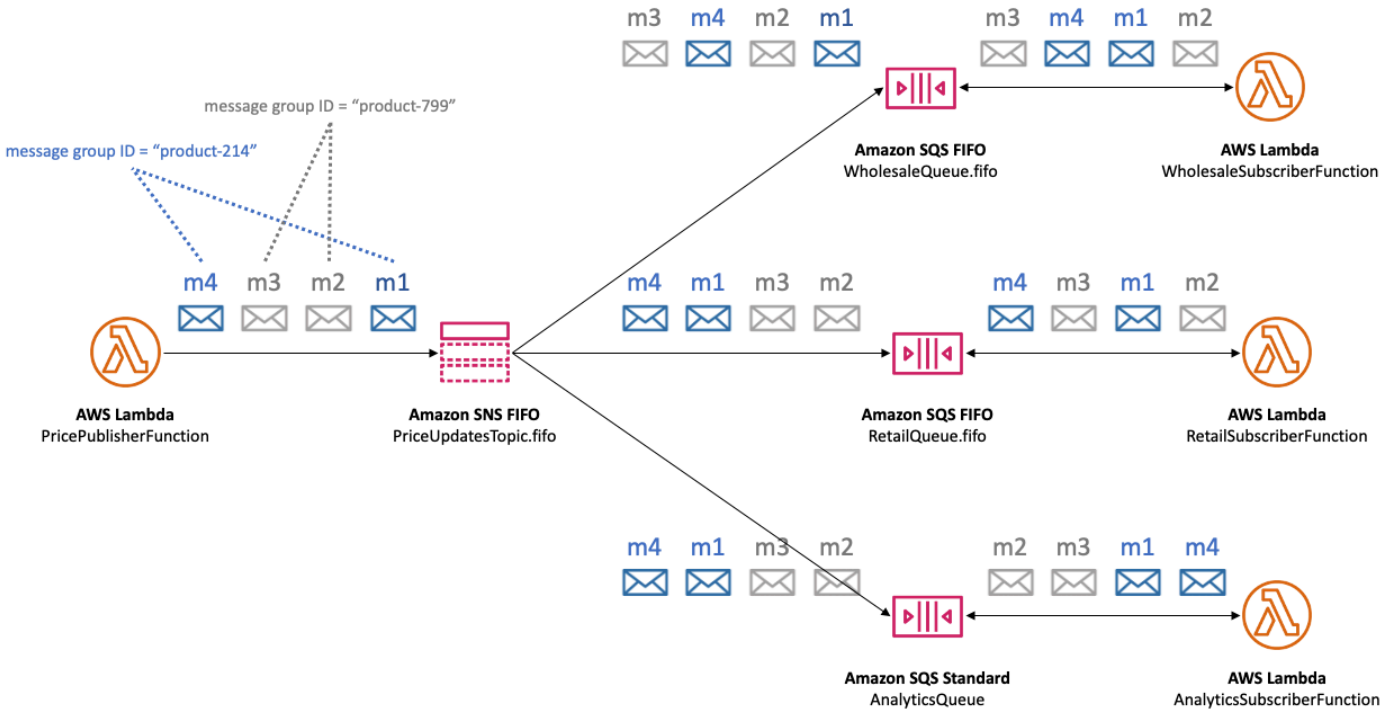
I messaggi che appartengono allo stesso gruppo vengono elaborati uno per uno, in un ordine rigoroso rispetto al gruppo.

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

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 Invio di SNS messaggi su Amazon per FIFO argomenti e Filtraggio dei SNS messaggi Amazon 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 SNS FIFO argomento Amazon 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. Nota che, per il prodotto il cui ID del gruppo di messaggi

è product-214, il messaggio m4 viene elaborato prima del messaggio m1. Questa sequenza viene preservata in tutti i flussi di lavoro che utilizzano Amazon SNS FIFO to Amazon SQSFIFO. Allo stesso modo, per il prodotto il cui ID del gruppo di messaggi è product-799, il messaggio m3 viene elaborato prima del messaggio m2, a condizione che i flussi di lavoro utilizzino Amazon e Amazon SNSFIFO. SQS FIFO Tuttavia, quando si utilizzano le code SQS standard di Amazon, 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 per gruppo di messaggi per migliorare le prestazioni IDs

Per ottimizzare la velocità di consegna, SNS FIFO gli argomenti di Amazon 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 un throughput elevato per un singolo argomento, utilizza un gran numero di gruppi di messaggi distinti. IDs Utilizzando un set diversificato di gruppi di messaggi, Amazon SNS FIFO Topics distribuisce automaticamente i messaggi su un numero maggiore di partizioni parallele.

> **Note**
>
> SNSFIFOGli argomenti di Amazon sono ottimizzati per la distribuzione uniforme dei messaggi tra i gruppi di messaggiIDs, indipendentemente dal numero di gruppi. AWS consiglia di utilizzare un gran numero di gruppi di messaggi distinti IDs per prestazioni ottimizzate.

Quando pubblichi sul tuo SNS FIFO argomento Amazon con un throughput elevato e hai sottoscritto una o più SQS FIFO code Amazon, ti consigliamo di abilitare un throughput elevato nelle code. Per ulteriori informazioni, consulta High throughput for FIFO queues nella Amazon Simple Queue Service Developer Guide.

Invio di SNS messaggi su Amazon per FIFO argomenti

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

Per i carichi di lavoro che devono mantenere un ordine rigoroso dei messaggi o la deduplicazione, la combinazione di argomenti di Amazon SNS FIFO con SQSFIFOcode Amazon sottoscritte come endpoint di consegna consente di migliorare la 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 ordini e consegne con il massimo impegno at-least-once, l'iscrizione alle SQS code standard di Amazon agli SNS FIFO argomenti Amazon offre la possibilità di ridurre i costi, oltre a condividere le code tra carichi di lavoro che non vengono utilizzati. FIFO

> **Note**
>
> Per distribuire i messaggi dagli SNS FIFO argomenti di Amazon alle AWS Lambda funzioni, sono necessari passaggi aggiuntivi. Innanzitutto, iscriviti ad Amazon SQS FIFO o standard queues all'argomento. Quindi configurare le code per attivare le funzioni. Per ulteriori informazioni, consulta il post SQSFIFO«as an event source» sul blog di AWS Compute.

SNSFIFOgli argomenti non possono recapitare messaggi agli endpoint gestiti dai clienti, come indirizzi e-mail, app per dispositivi mobili, numeri di telefono per la messaggistica di testo (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 dai clienti ad SNS FIFO argomenti generano errori.

SNSFIFOgli argomenti supportano le stesse funzionalità di filtro dei messaggi degli argomenti standard. Per ulteriori informazioni, consulta Filtraggio dei SNS messaggi Amazon per argomenti FIFO il post Simplify Your Pub/Sub Messaging with Amazon SNS Message Filtering sul blog di Compute.AWS

Filtraggio dei SNS messaggi Amazon per argomenti FIFO

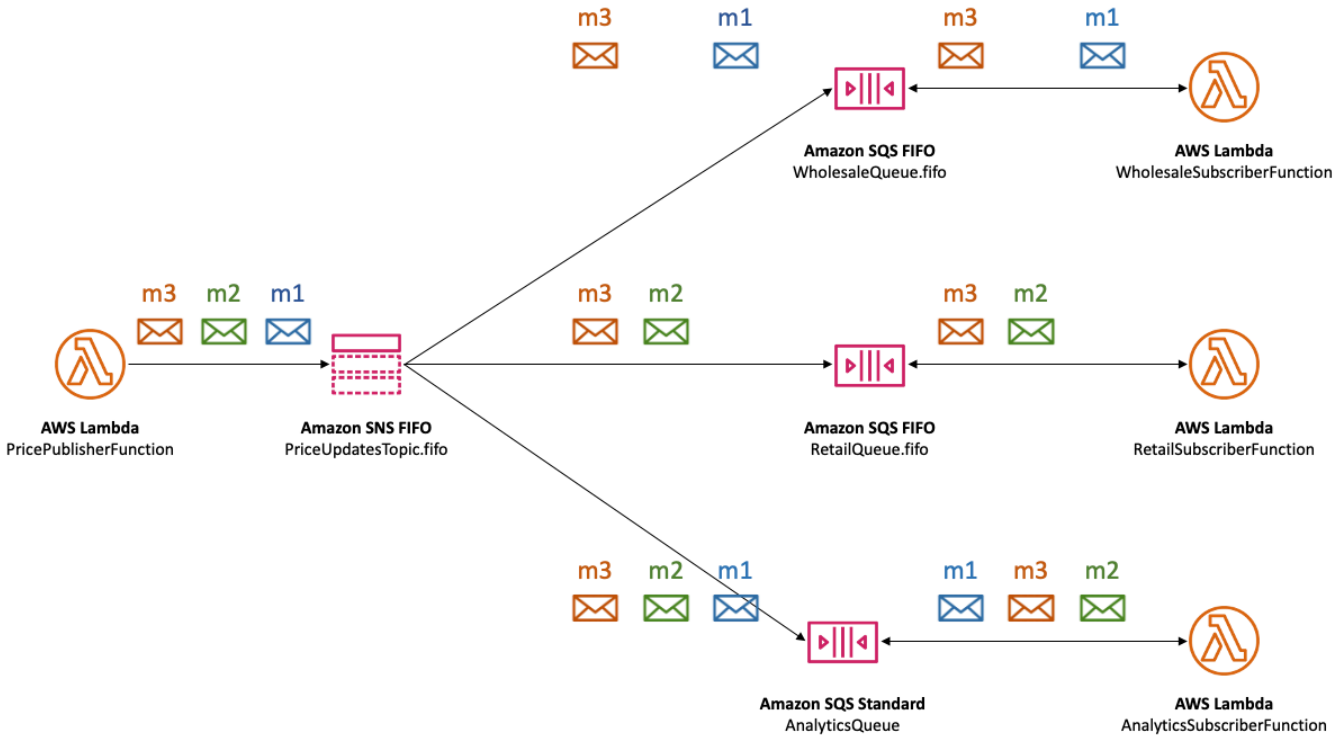
SNSFIFOGli argomenti di Amazon supportano il filtraggio dei messaggi. L'utilizzo del filtro dei messaggi semplifica l'architettura scaricando la logica di instradamento dei messaggi dai sistemi di pubblicazione e la logica di filtro dei messaggi dai sistemi di sottoscrizione.

Quando sottoscrivi un abbonamento Amazon SQS FIFO o una coda standard a un SNS FIFO argomento, 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.

Nell'esempio di gestione dei prezzi dei ricambi auto, supponiamo che siano impostate le seguenti politiche di SNS filtro di Amazon e che l'ambito della politica di filtro 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.



SNSFIFOgli argomenti 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 SNS dei messaggi Amazon.

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

Deduplicazione dei SNS messaggi Amazon per argomenti FIFO

SNSFIFOGli argomenti di Amazon e le SQS FIFO code Amazon supportano la deduplicazione dei messaggi, che prevede la consegna e l'elaborazione dei messaggi esattamente una volta, purché siano soddisfatte le seguenti condizioni:

- La SQS FIFO coda Amazon sottoscritta esiste e dispone di autorizzazioni che consentono al responsabile del SNS servizio Amazon di recapitare i messaggi alla coda.

- L'utente Amazon SQS FIFO Queue elabora il messaggio e lo elimina dalla coda prima della scadenza del timeout di visibilità.
- L'argomento relativo agli SNS abbonamenti Amazon non prevede filtri per i messaggi. Quando configuri il filtraggio dei messaggi, SNS FIFO gli argomenti di Amazon supportano la at-most-once consegna, poiché i messaggi possono essere filtrati in base alle politiche di filtro degli abbonamenti.
- Non ci sono interruzioni di rete che impediscono il riconoscimento del recapito dei messaggi.

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

Quando pubblichi un messaggio su un SNS FIFO argomento di Amazon, il messaggio deve includere un ID di deduplicazione. Questo ID è incluso nel messaggio che l'SNSFIFOargomento Amazon invia alle SQS FIFO code di abbonati Amazon.

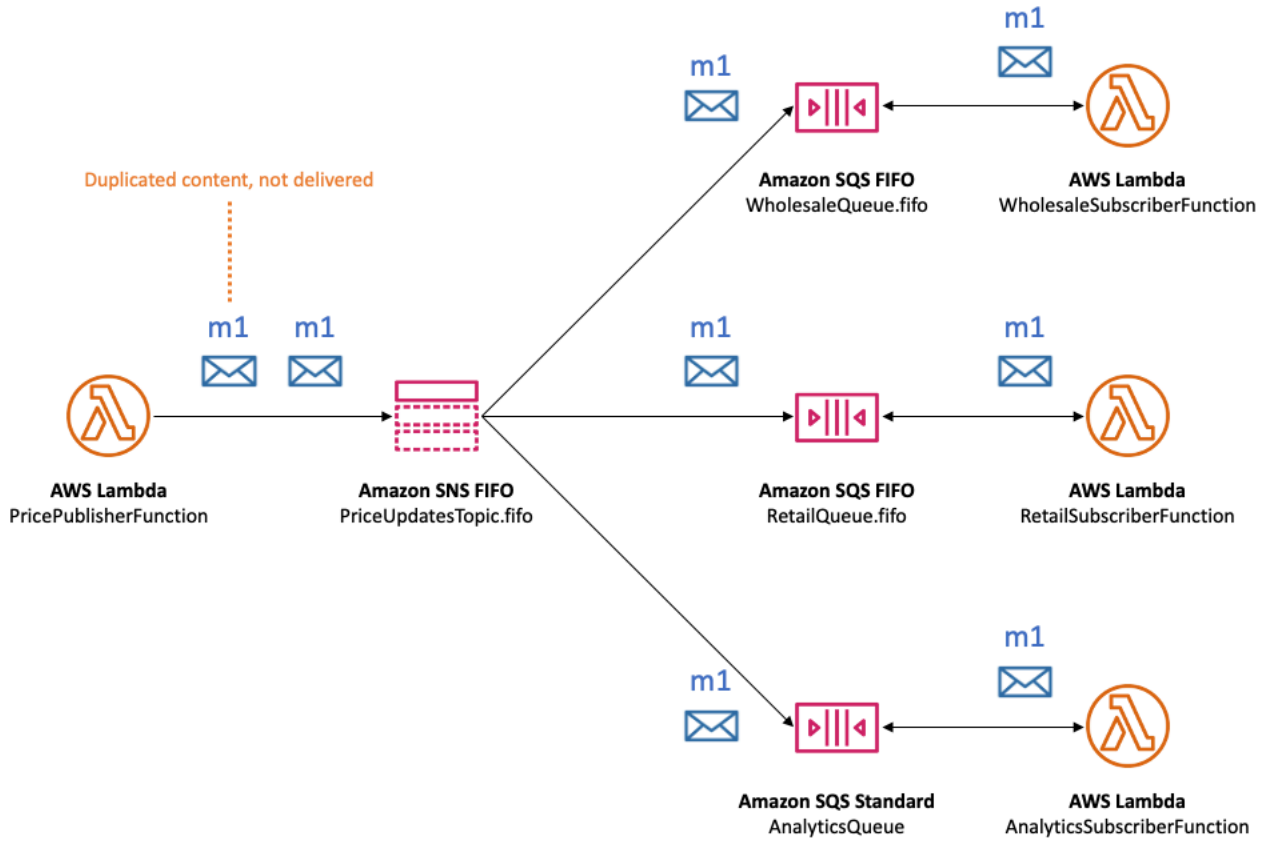
Se un messaggio con un particolare ID di deduplicazione viene pubblicato con successo su un SNS FIFO argomento di Amazon, qualsiasi messaggio pubblicato con lo stesso ID di deduplicazione, entro l'intervallo di deduplicazione di cinque minuti, viene accettato ma non recapitato. L'SNSFIFOargomento Amazon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

Se è garantito che il corpo del messaggio sia unico per ogni messaggio pubblicato, puoi abilitare la deduplicazione basata sul contenuto per un SNS FIFO argomento Amazon e le code Amazon sottoscritte. SQS FIFO Amazon SNS utilizza il corpo del messaggio per generare un valore hash univoco da utilizzare come ID di deduplicazione per ogni messaggio, quindi 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 SNS FIFO argomento di Amazon 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 aggiungesse 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 sui contenuti nell'SNSFIFOargomento Amazon e nelle code Amazon degli abbonati. SQS FIFO



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

Sicurezza dei SNS messaggi Amazon per FIFO argomenti

Puoi scegliere di far sì che Amazon SNS e Amazon SQS crittografino i messaggi inviati ad FIFO argomenti e code, utilizzando AWS Key Management Service (AWS KMS) le chiavi master del cliente (CMKs). Puoi creare FIFO argomenti e code crittografati o scegliere di crittografare argomenti e code esistentiFIFO. 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 della crittografia a un FIFO argomento o a una coda esistente non comporta la crittografia dei messaggi in arretrato, mentre la rimozione della crittografia da un argomento o da una coda lascia i messaggi in backlog crittografati.

SNSFIFOgli argomenti decrittografano i messaggi immediatamente prima di recapitarli agli endpoint sottoscritti. SQSFIFOle code decrittografano il messaggio appena prima di restituirlo all'applicazione consumer. Per ulteriori informazioni, consulta Crittografia SNS dei dati Amazon i messaggi di crittografia pubblicati su Amazon SNS con AWS KMS post sul AWS Compute Blog.

Inoltre, SNS FIFO gli argomenti e le SQS FIFO code supportano la privacy dei messaggi con VPCendpoint di interfaccia forniti da. AWS PrivateLink Utilizzando gli endpoint dell'interfaccia, puoi inviare messaggi dalle sottoreti Amazon Virtual Private Cloud (AmazonVPC) ad FIFO argomenti e code senza attraversare la rete Internet pubblica. Questo modello mantiene la messaggistica all'interno dell' AWS infrastruttura e della rete, il che migliora la sicurezza complessiva dell'applicazione. Quando si utilizza AWS PrivateLink, non è necessario configurare un gateway Internet, la traduzione degli indirizzi di rete (NAT) o una rete privata virtuale (VPN). Per ulteriori informazioni, consulta Protezione del SNS traffico Amazon con VPC endpoint i messaggi di sicurezza pubblicati su Amazon SNS con AWS PrivateLink post sul AWS Security Blog.

SNSFIFOgli argomenti supportano anche le code di lettere non scritte e l'archiviazione dei messaggi nelle zone di disponibilità. Per ulteriori informazioni, consulta Durabilità dei SNS messaggi Amazon per FIFO argomenti.

Durabilità dei SNS messaggi Amazon per FIFO argomenti

SNSFIFOGli argomenti di Amazon e le SQS code Amazon 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 AmazonSNS, la consegna dei messaggi non riesce quando l'SNSargomento Amazon non riesce ad accedere a una SQS coda Amazon sottoscritta a causa di un errore lato client o lato server:

- Gli errori lato client si verificano quando l'SNSFIFOargomento Amazon contiene metadati di abbonamento obsoleti. Due cause comuni di errori lato client si verificano quando il proprietario della SQS coda Amazon esegue una delle seguenti operazioni:
  - Elimina la coda.
  - Modifica la politica di coda in modo da impedire al responsabile del SNS servizio Amazon di recapitargli messaggi.

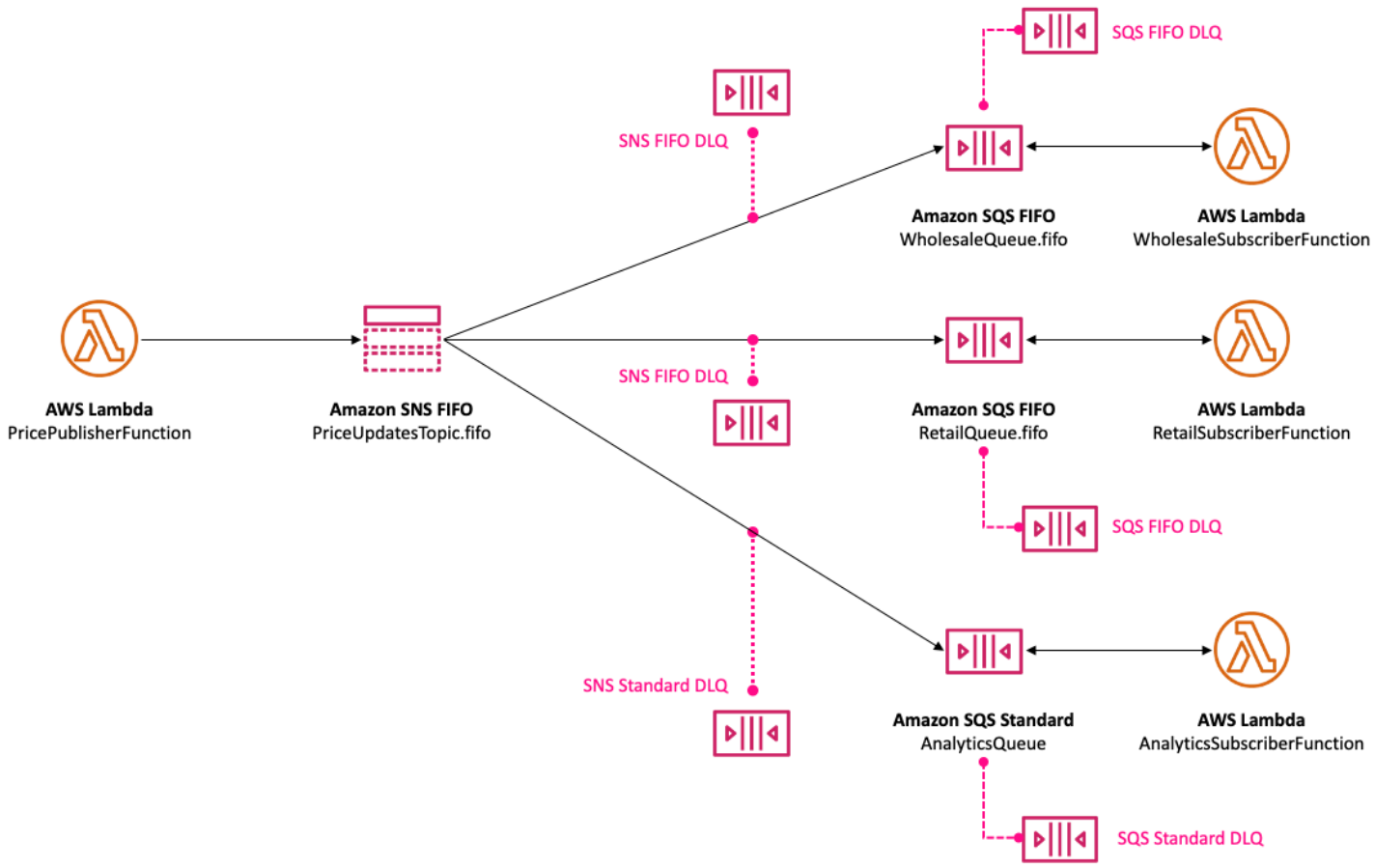
  Amazon SNS non riprova a recapitare i messaggi che non sono riusciti a causa di errori sul lato client.
- Errori sul lato server possono verificarsi nelle seguenti situazioni:
  - Il SQS servizio Amazon non è disponibile.
  - Amazon SQS non riesce a elaborare una richiesta valida dal SNS servizio Amazon.

  Quando si verificano errori sul lato server, Amazon SNS FIFO Topics riprova le consegne non riuscite fino a 100.015 volte nell'arco di 23 giorni. Per ulteriori informazioni, consulta Tentativi di recapito dei SNS messaggi Amazon.

Per qualsiasi tipo di errore, Amazon SNS può mettere da parte i messaggi nelle code di SQS posta indesiderata di Amazon in modo da non perdere i dati.

In AmazonSQS, l'elaborazione dei messaggi fallisce quando l'applicazione consumer non riesce a ricevere il messaggio, elaborarlo ed eliminarlo dalla coda. Quando non viene raggiunto il numero massimo di richieste di ricezione, Amazon SQS può mettere da parte i messaggi e metterli in coda di lettere non scritte in modo da non perdere i dati.

Nel caso di esempio relativo alla gestione dei prezzi dei ricambi auto, l'azienda può assegnare un Amazon SQS dead-letter queue () DLQ a ogni abbonamento SNS FIFO tematico Amazon, nonché a ciascuna coda Amazon sottoscritta. SQS Questo protegge l'azienda da qualsiasi perdita di aggiornamento dei prezzi.



La coda di lettere non scritte associata a un SNS abbonamento Amazon deve essere una SQS coda Amazon dello stesso tipo della coda di sottoscrizione. Ad esempio, l'SNSFIFOabbonamento Amazon per una SQS FIFO coda Amazon deve avere una coda Amazon come SQS FIFO coda di lettere non scritte. Allo stesso modo, l'SNSFIFOabbonamento Amazon per una coda Amazon SQS standard deve avere una coda Amazon SQS standard come coda di lettere non scritte. Per ulteriori informazioni, consulta Code di lettere non SNS ricevute su Amazon il AWS Lambda post Progettazione di app serverless DLQs durevoli con Amazon SNSSQS, Amazon sul AWS Compute Blog.

Per una maggiore durabilità e facilitare il ripristino dagli errori a valle, i proprietari degli argomenti possono utilizzare FIFO gli argomenti anche 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 SNS messaggi su Amazon per argomenti FIFO.

Archiviazione e riproduzione dei SNS messaggi su Amazon per argomenti FIFO

Che cosa è l'archiviazione e riproduzione dei messaggi?

Amazon SNS offre una funzionalità di archiviazione e riproduzione dei messaggi senza codice, progettata specificamente per argomenti FIFO (First-In-First-Out). Questa funzionalità consente ai proprietari degli argomenti di archiviare i messaggi direttamente nell'archivio degli argomenti per un massimo di 365 giorni e di riprodurli agli abbonati quando necessario. L'archiviazione e la riproduzione dei messaggi sono essenziali per recuperare i messaggi persi e sincronizzare le applicazioni tra regioni o sistemi mediante la replica degli stati.

È possibile accedere a questa funzionalità tramite,, e AWS API. SDK AWS CloudFormation AWS Management Console

Casi d'uso principali

- Recupero dei messaggi: recupera i messaggi persi a causa di errori delle applicazioni downstream riproducendoli sull'endpoint dell'abbonato.
- Replica dello stato: replica lo stato di un sistema esistente in un nuovo ambiente riproducendo i messaggi a partire da un timestamp specifico.
- Correzione degli errori: invia nuovamente i messaggi persi durante le interruzioni per garantire che tutti gli eventi vengano elaborati correttamente.

Componenti dell'archiviazione e della riproduzione dei messaggi

Gestisci l'archiviazione e la riproduzione dei messaggi per SNS FIFO gli argomenti di Amazon, tra cui l'impostazione dei periodi di conservazione, il monitoraggio dell'utilizzo dei messaggi archiviati CloudWatch, l'avvio dei replay tramite gli attributi dell'abbonamento e la comprensione delle autorizzazioni necessarie per modificare e avviare i replay.

Archiviazione dei messaggi

- Il proprietario dell'argomento abilita la funzionalità di archiviazione e imposta il periodo di conservazione dei messaggi, che può arrivare fino a 365 giorni. Per ulteriori informazioni, consulta Archiviazione dei SNS messaggi Amazon per i proprietari di FIFO argomenti
- CloudWatch le metriche aiutano a monitorare i messaggi archiviati.

Riproduzione dei messaggi

- Un abbonato avvia una riproduzione, selezionando la finestra temporale in cui i messaggi devono essere rielaborati sull'endpoint sottoscritto. Per ulteriori informazioni, consulta Riproduzione dei SNS messaggi Amazon per gli abbonati all'FIFOargomento.
- È possibile gestire la riproduzione tramite gli attributi dell'abbonamento utilizzando la funzione. `ReplayPolicy`

Autorizzazioni pertinenti

- `SetSubscriptionAttributes`— Necessario per configurare o modificare le impostazioni di riproduzione utilizzando l'`ReplayPolicy`attributo su un abbonamento.
- `Subscribe`— Necessario per allegare un nuovo abbonamento e avviare i replay.
- `GetTopicAttributes`— Consente di visualizzare le proprietà dell'argomento, ma l'avvio del replay ruota principalmente attorno alla gestione dell'abbonamento.

Archiviazione dei SNS messaggi Amazon per i proprietari di FIFO argomenti

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 la pagina SNSdei prezzi di Amazon.

Crea una politica di archiviazione dei messaggi utilizzando il AWS Management Console

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

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

> **Note**
>
> L'archiviazione e la riproduzione dei SNS messaggi di Amazon sono disponibili solo per argomenti application-to-application (A2A). FIFO

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. Scegli Save changes (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 politica di archiviazione è impostata su un oggetto vuoto. JSON
  - 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.
  - JSONpreview: l'JSONanteprima della politica 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 politica 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.

Crea una politica di archiviazione dei messaggi utilizzando API

Per creare una politica di archiviazione dei messaggi utilizzandoAPI, devi aggiungere l'attributo `ArchivePolicy` al tuo argomento. È possibile impostarne una `ArchivePolicy` utilizzando le API azioni `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:

```
{}
```

Crea una politica di archiviazione dei messaggi utilizzando SDK

Per utilizzare un AWS SDK, è necessario configurarlo con le proprie credenziali. Per ulteriori informazioni, consulta `configShared and credentials files` nella AWS SDKsand Tools Reference Guide.

Il seguente esempio di codice mostra come impostare un SNS argomento Amazon in modo che conservi tutti i messaggi pubblicati sull'argomento per 30 giorni. `ArchivePolicy`

```
// 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);
```

Crea una politica di archiviazione dei messaggi utilizzando AWS CloudFormation

Per creare una politica di archiviazione, AWS CloudFormation `AWS::SNS::Topic`consulta la Guida AWS CloudFormation per l'utente.

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 SNS necessario fornire ad Amazon `Decrypt` l'accesso alla KMS chiave utilizzata per crittografare i messaggi nell'archivio.

1. Quando crittografi i messaggi con una KMS chiave e li memorizzi all'interno dell'argomento, devi concedere ad Amazon SNS la possibilità di decrittografare questi messaggi tramite Key Policy. Per ulteriori informazioni, consulta Concedi autorizzazioni di decrittografia ad Amazon SNS.
2. Abilita AWS KMS per AmazonSNS. Per ulteriori informazioni, consulta Configurazione delle autorizzazioni AWS KMS.

> **Important**
>
> Quando aggiungi le nuove sezioni alla tua politica KMS chiave, non modificare le sezioni esistenti della politica. Se la crittografia è abilitata su un argomento e la KMS chiave è disabilitata o eliminata o la politica della KMS chiave non è configurata correttamente per AmazonSNS, Amazon SNS non può riprodurre i messaggi ai tuoi abbonati.

Concedi autorizzazioni di decrittografia ad Amazon SNS

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

Di seguito è riportato un esempio di politica necessaria per consentire al responsabile del SNS servizio Amazon di decrittografare i messaggi archiviati durante una riproduzione dei messaggi storici dall'argomento.

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

Monitora i parametri di archiviazione dei messaggi utilizzando Amazon CloudWatch

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

Parametro	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

Parametro	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.

`GetTopicAttributes`APIHa una `BeginningArchiveTime` proprietà che rappresenta il timestamp più vecchio in base al quale un abbonato può avviare un replay. Quanto segue rappresenta un esempio di risposta per questa azione: API

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

Riproduzione dei SNS messaggi Amazon per gli abbonati all'FIFOargomento

Amazon SNS replay consente agli abbonati agli argomenti di recuperare i messaggi archiviati dal Topic Data Store e di reinviarli (o riascoltarli) 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.

Crea una politica di riproduzione dei messaggi utilizzando AWS Management Console

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

1. Accedi alla SNSconsole Amazon.
2. Scegli una sottoscrizione all'argomento o creane una nuova. Per ulteriori informazioni sulla creazione di sottoscrizioni, consultaCreare un abbonamento a un SNS argomento di Amazon.
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 la data e l'ora di inizio della riproduzione: scegli la data (YYYY/MM/DDformato) e l'ora (formato 24 ore hh:mm:ss) da cui desideri iniziare a riprodurre i messaggi archiviati. L'ora di inizio deve essere successiva all'inizio dell'orario di archiviazione approssimativo.
   b. (Facoltativo) Scegli la data e l'ora di fine della riproduzione: scegli la data (YYYY/MM/DDformato) e l'ora (formato hh:mm:ss 24 ore) in cui desideri interrompere 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, vedi. CloudWatch Monitora le metriche di riproduzione dei messaggi 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.

Aggiungi una politica di riproduzione all'abbonamento utilizzando il API

Per riprodurre i messaggi archiviati usa l'attributo. `ReplayPolicy` `ReplayPolicy`può essere usato con le azioni `Subscribe` and `SetSubscriptionAttributes`API. 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 a valle e inviare nuovamente tutti i messaggi per un periodo di due ore il 1° ottobre 2023, utilizza l'`SetSubscriptionAttributes`APIazione per impostare a `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 nuovi messaggi pubblicati sull'argomento, utilizza l'`SetSubscriptionAttributes`APIazione per impostare un `ReplayPolicy` 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.

Aggiungi una politica di riproduzione all'abbonamento utilizzando il SDK

Per usare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta `configShared and credentials files` nella AWS SDKsand Tools Reference Guide.

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

```
// 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);
```

Comprendere il EndingPoint

Quando applichi un `ReplayPolicy` SNS abbonamento Amazon, il `EndingPoint` valore è facoltativo. Se non `EndingPoint` viene fornito alcun valore, la riproduzione inizierà dal valore specificato `StartingPoint` e continuerà fino al raggiungimento dell'ora corrente, inclusa l'elaborazione dei messaggi appena pubblicati. Una volta recuperato, l'abbonamento funzionerà come un normale abbonamento e riceverà nuovi messaggi man mano che vengono pubblicati.

Se `EndingPoint` viene specificato un, il servizio riprodurrà i messaggi dall'inizio alla fine `EndingPoint` e poi `StartingPoint` si interromperà. Questa azione mette effettivamente in pausa

l'abbonamento. Mentre l'abbonamento è in pausa, i messaggi appena pubblicati non verranno recapitati all'endpoint sottoscritto.

Per riprendere il recapito dei messaggi, applicane uno nuovo `ReplayPolicy` senza specificarne uno `EndingPoint` e impostalo sul momento desiderato `StartingPoint` a partire dal quale continuare a ricevere i messaggi. Ad esempio, per riprendere un abbonamento dal punto in cui era terminato un replay precedente, imposta il nuovo su `StartingPoint` quello fornito in precedenza. `EndingPoint`

Filtro dei messaggi riprodotti

SNSIl filtro dei messaggi di Amazon ti consente di controllare i messaggi riprodotti che Amazon SNS riproduce sul tuo endpoint di abbonato. Quando il filtraggio dei messaggi e l'archiviazione dei messaggi sono entrambi abilitati, Amazon recupera SNS prima il messaggio dall'archivio dati dell'argomento, quindi lo applica a quello dell'abbonamento. `FilterPolicy` Il messaggio viene recapitato all'endpoint sottoscritto quando c'è una corrispondenza, altrimenti il messaggio viene filtrato. Per ulteriori informazioni, consulta Politiche di filtro degli SNS abbonamenti Amazon.

Monitora le metriche di riproduzione dei messaggi utilizzando Amazon CloudWatch

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

Parametro	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 SNS codice Amazon per FIFO argomenti

Puoi utilizzare i seguenti esempi di codice per integrare l'esempio di gestione dei prezzi dei ricambi auto utilizzando un SNS FIFO argomento Amazon con una SQS FIFO coda Amazon o una coda standard.

Utilizzando un AWS SDK

Utilizzando un AWS SDK, crei un SNS FIFO argomento Amazon impostando il relativo `FifoTopic` attributo su**`true`**. Puoi creare una SQS FIFO coda Amazon impostando il relativo `FifoQueue` attributo su**`true`**. Inoltre, devi aggiungere il **`.fifo`** suffisso al nome di ogni FIFO risorsa. Dopo aver creato un FIFO argomento o una coda, non è possibile convertirlo in un argomento o una coda standard.

I seguenti esempi di codice creano queste FIFO e altre risorse di coda standard:

- L'SNSFIFOargomento di Amazon che distribuisce gli aggiornamenti dei prezzi
- Le SQS FIFO code Amazon che forniscono questi aggiornamenti alle applicazioni di vendita all'ingrosso e al dettaglio
- La coda SQS standard di Amazon per l'applicazione di analisi che archivia i record, che possono essere interrogati per la business intelligence (BI)
- SNSFIFOGli abbonamenti Amazon che collegano 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 Filtraggio dei SNS messaggi Amazon per argomenti FIFO.

Java

SDKper 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

- crea un SNS FIFO argomento Amazon, due SQS FIFO code Amazon 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 API i dettagli, consulta i seguenti argomenti in AWS SDK for Java 2.x API Reference.
  - CreateTopic
  - Pubblicare
  - Subscribe

Python

SDKper Python (Boto3)

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

Crea un SNS FIFO argomento Amazon, sottoscrivi Amazon SQS FIFO e le code standard all'argomento e pubblica un messaggio sull'argomento.

```
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 API i dettagli, consulta i seguenti argomenti in AWS SDKPython (Boto3) Reference. API
  - CreateTopic
  - Pubblicare
  - Subscribe

SAP ABAP

SDKper SAP ABAP

> **Note**
>
> C'è 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 FIFO argomento, iscriviti a una SQS FIFO coda Amazon all'argomento e pubblica un messaggio su un SNS argomento Amazon.

```
    " 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 API i dettagli, consulta i seguenti argomenti SAPABAPAPIcome AWS SDK riferimento.
  - CreateTopic
  - Pubblicare
  - Subscribe

Ricezione di messaggi dagli FIFO abbonamenti

Ora puoi ricevere aggiornamenti dei prezzi nelle tre applicazioni sottoscritte. Come illustrato inthe section called “FIFOcaso d'uso dell'argomento”, il punto di accesso per ogni applicazione consumer è la SQS coda di Amazon, che la AWS Lambda funzione corrispondente può interrogare automaticamente. Quando una SQS coda Amazon è un'origine di eventi per una funzione Lambda, Lambda ridimensiona la sua flotta di poller in base alle esigenze per consumare in modo efficiente i messaggi.

Per ulteriori informazioni, consulta Using AWS Lambda with Amazon SQS nella AWS Lambda Developer Guide. Per informazioni sulla creazione di sondaggi di coda personalizzati, consulta Recommendations for Amazon SQS standard and FIFO queues nella Amazon Simple Queue Service Developer Guide e nell'Amazon ReceiveMessageSimple Queue Service Reference. API

Usando AWS CloudFormation

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

- L'SNSFIFOargomento di Amazon che distribuisce gli aggiornamenti dei prezzi
- Le SQS FIFO code Amazon che forniscono questi aggiornamenti alle applicazioni di vendita all'ingrosso e al dettaglio
- La coda SQS standard di Amazon per l'applicazione di analisi che archivia i record, che possono essere interrogati per la business intelligence (BI)
- SNSFIFOGli abbonamenti Amazon che collegano 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 AWS delle risorse utilizzando un AWS CloudFormation modello, consulta Get Started nella Guida per l'AWS CloudFormation utente.

Filtraggio SNS dei messaggi Amazon

Per impostazione predefinita, un abbonato a un SNS argomento Amazon riceve tutti i messaggi pubblicati sull'argomento. Per ricevere solo un sottoinsieme dei messaggi, un sottoscrittore deve assegnare una policy di filtro alla sottoscrizione all'argomento.

Una politica di filtro è un JSON oggetto contenente proprietà che definiscono i messaggi ricevuti dall'abbonato. Amazon SNS supporta politiche che agiscono sugli attributi del messaggio o sul corpo del messaggio, in base all'ambito della politica di filtro che hai impostato per l'abbonamento. Le politiche di filtro per il corpo del messaggio presuppongono che il payload del messaggio sia un oggetto ben formatoJSON.

Se non dispone di una policy di filtro, il sottoscrittore riceve ogni messaggio pubblicato nell'argomento. Quando pubblichi un messaggio su un argomento con una politica di filtro in atto, Amazon SNS confronta gli attributi o il corpo del messaggio con le proprietà della politica di filtro per ciascuno degli abbonamenti dell'argomento. Se tutti gli attributi o le proprietà del corpo del messaggio soddisfano le condizioni specificate nella politica di filtro, Amazon SNS invia il messaggio all'abbonato. Altrimenti, Amazon SNS non invia il messaggio a quell'abbonato.

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

Ambito della politica di filtro degli SNS abbonamenti Amazon

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`.

Politiche di filtro degli SNS abbonamenti Amazon

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 SNS dei messaggi Amazon.

Quando Amazon SNS valuta gli attributi o le proprietà del corpo del messaggio rispetto alla politica di filtro degli abbonamenti, ignora quelli che non sono specificati nella politica.

> **Important**
>
> AWS servizi come Amazon IAM e Amazon SNS utilizzano un modello di calcolo distribuito chiamato eventuale coerenza. 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.

Amazon SNS attualmente supporta i seguenti operatori di filtro:

- ANDlogica
- 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

SNSEsempi di politiche di filtro di Amazon

L'esempio seguente mostra un payload di messaggi recapitato da un SNS argomento di Amazon che elabora le transazioni dei 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 SNS dei messaggi Amazon.

```
{
   "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"
   ]
}
```

Filtra i vincoli delle policy in Amazon SNS

Quando crei una policy di filtro, devi considerare i seguenti vincoli:

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"]
}
```

La prima matrice contiene tre valori, la seconda un valore e la terza due valori. La combinazione totale si calcola nel seguente modo:

```
3 x 1 x 2 = 6
```

- La politica JSON di filtro può contenere quanto segue:
  - Stringhe tra virgolette
  - Numeri

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

- Quando usi Amazon SNSAPI, devi passare la politica JSON di filtro come una stringa UTF -8 valida.

- 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 Amazon SQS Queue con. `Subscribe` API Tuttavia, fallirà quando alleghi la politica di filtro alla `Subscribe` API chiamata (o alla `SetSubscriptionAttributes` API chiamata).

  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 politica di filtri annidati per il filtraggio basato sugli attributi.
- Amazon SNS confronta le proprietà delle policy solo con gli attributi dei messaggi 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 dei messaggi con il tipo di `Binary` dati.
- 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 filtri annidati 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"]
    }
}
```

- Il primo array ha quattro valori in una chiave annidata a tre livelli e il secondo ha tre valori in una chiave annidata a due livelli. La combinazione totale si calcola nel seguente modo:

```
4 x 3 x 3 x 2 = 72
```

- Una policy di filtro può contenere fino a cinque nomi di attributo. Per una policy annidata, vengono conteggiate solo le chiavi padre.
- Per passare dal filtro basato sugli attributi (predefinito) al filtro basato sul payload, è necessario impostare `FilterPolicyScope` su `MessageBody` nella sottoscrizione.

ANDLogica /OR

Puoi utilizzare operazioni che includono la logica AND /OR per abbinare gli attributi o le proprietà del corpo del messaggio.

Argomenti

- ANDlogica

- Logica OR
- Operatore OR

ANDlogica

Puoi applicare AND la logica 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 AND la logica 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 riconosce una `"$or"` relazione SNS solo quando la politica soddisfa tutte le seguenti condizioni. Se tutte queste condizioni non sono soddisfatte, `"$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. I valori numerici non sono racchiusi tra virgolette doppie nella politica. JSON 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. I valori delle stringhe sono racchiusi tra virgolette doppie nella policy. JSON 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"
}
```

Equals-ignore-case abbinamento

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 in base all'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 politica di filtro degli abbonamenti in Amazon SNS

Il filtraggio dei messaggi in Amazon ti SNS consente di recapitare messaggi in modo selettivo agli abbonati in base a politiche di filtro. Queste politiche definiscono le condizioni che i messaggi devono soddisfare per essere recapitati a un abbonamento. Sebbene l'invio di messaggi non elaborati sia un'opzione che può influire sull'elaborazione dei messaggi, non è necessario che i filtri di abbonamento funzionino.

Puoi applicare una politica di filtro a un SNS abbonamento Amazon utilizzando la SNS console Amazon. Oppure, per applicare le politiche a livello di codice, puoi utilizzare Amazon SNSAPI, il AWS Command Line Interface (AWS CLI) o qualsiasi altro AWS SDK che supporti Amazon. SNS Puoi anche usare. AWS CloudFormation

Abilitazione del recapito di messaggi non elaborati

La consegna non elaborata dei messaggi garantisce che i payload dei messaggi vengano consegnati così come sono agli abbonati senza alcuna codifica o trasformazione aggiuntiva. Ciò può essere utile quando gli abbonati richiedono il formato originale dei messaggi per l'elaborazione. Tuttavia, il recapito dei messaggi non elaborati non è direttamente correlato alla funzionalità dei filtri di abbonamento.

Applicazione dei filtri di abbonamento

Per applicare i filtri dei messaggi a un abbonamento, è necessario definire una politica di filtro utilizzando la JSON sintassi. Questa politica specifica le condizioni che un messaggio deve soddisfare per essere recapitato all'abbonamento. I filtri possono essere basati sugli attributi del messaggio, come gli attributi del messaggio, la struttura del messaggio o persino il contenuto del messaggio.

Relazione tra i filtri di invio di messaggi non elaborati e di sottoscrizione

Sebbene l'attivazione del recapito dei messaggi non elaborati possa influire sul modo in cui i messaggi vengono recapitati ed elaborati dagli abbonati, non è un prerequisito per l'utilizzo dei filtri di abbonamento. Tuttavia, negli scenari in cui gli abbonati richiedono il formato originale dei messaggi senza alcuna modifica, l'attivazione del recapito dei messaggi non elaborati potrebbe essere utile oltre ai filtri di abbonamento.

Considerazioni per un filtraggio efficace

Quando implementate il filtraggio dei messaggi, tenete conto dei requisiti specifici dell'applicazione e degli abbonati. Definite politiche di filtro che soddisfino accuratamente i criteri di recapito dei messaggi per garantire una distribuzione efficiente e mirata dei messaggi.

> **Important**
>
> AWS servizi come Amazon IAM e Amazon SNS utilizzano un modello di calcolo distribuito chiamato eventuale coerenza. 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 SNSconsole Amazon.
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 dell'JSONeditor, fornisci il JSONcorpo della tua politica di filtro.
7. Scegli Save changes (Salva modifiche).

   Amazon SNS applica la tua politica di filtro all'abbonamento.

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 tua JSONpolitica.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

Per fornire una politica validaJSON, racchiudi i nomi e i valori degli attributi tra virgolette doppie. Devi inoltre racchiudere l'intero argomento della policy tra virgolette. Per evitare virgolette errate, è possibile utilizzare le virgolette singole per racchiudere la politica e le virgolette doppie per racchiudere i JSON nomi e i valori, come illustrato nell'esempio precedente.

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 SDKs

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

> **Important**
>
> Se si utilizza l'esempio SDK per Java 2.x, la classe non `SNSMessageFilterPolicy` è disponibile immediatamente. Per istruzioni su come installare questa classe, consultate l'esempio dal GitHub sito Web.

CLI

AWS CLI

Impostazione degli attributi della sottoscrizione

L'`set-subscription-attributes`esempio seguente imposta l'`RawMessageDelivery`attributo su un SQS abbonamento.

```
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.

L'`set-subscription-attributes`esempio seguente imposta un `FilterPolicy` attributo a un SQS abbonamento.

```
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.

L'`set-subscription-attributes`esempio seguente rimuove l'`FilterPolicy`attributo da una SQS sottoscrizione.

```
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 API i dettagli, vedere SetSubscriptionAttributesin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi SetSubscriptionAttributes AWS SDK for Java 2.xAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere SetSubscriptionAttributesPython (Boto3) Reference.AWS SDK API

Amazon SNS API

Per applicare una politica di filtro con Amazon SNSAPI, invia una richiesta all'`SetSubscriptionAttributes`azione. Imposta il `AttributeName` parametro su `FilterPolicy` e imposta il `AttributeValue` parametro sulla tua politica di filtroJSON.

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 YAML modello JSON or per creare uno AWS CloudFormation stack. Per ulteriori informazioni, consultate la `FilterPolicy`proprietà della `AWS::SNS::Subscription` risorsa nella Guida per l'AWS CloudFormation utente e il AWS CloudFormation modello di esempio.

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'HTTPendpoint a cui iscriverti al tuo argomento.

   > **Tip**
   >
   > Se non disponi di un HTTP endpoint, creane uno.

5. Nella pagina Opzioni, scegli Next (Avanti).
6. Nella pagina Revisione scegli Create (Crea).

Rimuovere una politica di filtro degli abbonamenti in Amazon SNS

Per interrompere il filtraggio dei messaggi inviati a un abbonamento, rimuovi la politica di filtro dell'abbonamento sovrascrivendola con un corpo vuotoJSON. Dopo aver rimosso la policy, la sottoscrizione accetta ogni messaggio pubblicato.

Usando il AWS Management Console

1. Accedi alla SNSconsole Amazon.
2. Nel riquadro di navigazione, scegli Sottoscrizioni.
3. Seleziona una sottoscrizione e quindi scegli Edit (Modifica).
4. Nella sezione Modifica ***`EXAMPLE1-23bc-4567-d890-ef12g3hij456`***pagina, espandi la sezione Politica di filtro degli abbonamenti.
5. Nel campo dell'JSONeditor, fornisci un JSON corpo vuoto per la tua politica di filtro:`{}`.
6. Scegli Save changes (Salva modifiche).

   Amazon SNS applica la tua politica di filtro all'abbonamento.

Utilizzando il AWS CLI

Per rimuovere una politica di filtro con AWS CLI, usa il `set-subscription-attributes`comando e fornisci un JSON corpo vuoto per l'`--attribute-value`argomento:

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

Usare Amazon SNS API

Per rimuovere una politica di filtro con Amazon SNSAPI, invia una richiesta all'`SetSubscriptionAttributes`azione. Imposta il `AttributeName` parametro su `FilterPolicy` e fornisci un JSON corpo vuoto per il `AttributeValue` parametro.

Protezione dei dati dei messaggi in Amazon SNS

Argomenti

Cos'è la protezione dei dati dei messaggi?

La protezione dei dati dei messaggi salvaguarda i dati pubblicati sui tuoi SNS argomenti Amazon utilizzando politiche di protezione dei dati per controllare, mascherare, oscurare o bloccare le informazioni sensibili che si spostano tra applicazioni o AWS servizi.

Message Data Protection analizza i dati in movimento alla ricerca di informazioni di identificazione personale (PII) e informazioni sanitarie protette () utilizzando identificatori di dati. PHI Puoi scegliere di utilizzare identificatori di dati predefiniti (o SNS gestiti da Amazon) (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 fino al 99% dei dati pubblicati su un SNS argomento di Amazon. 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.
- Nega: blocca la trasmissione di dati tra applicazioni e AWS risorse se all'interno del payload sono presenti dati sensibili.

> **Note**
>
> Amazon SNS supporta la protezione dei dati dei messaggi solo per gli argomenti SNS standard di Amazon.

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 comeHIPAA, GDPRPCI, e RAMP Fed. 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 SNS standard di Amazon.

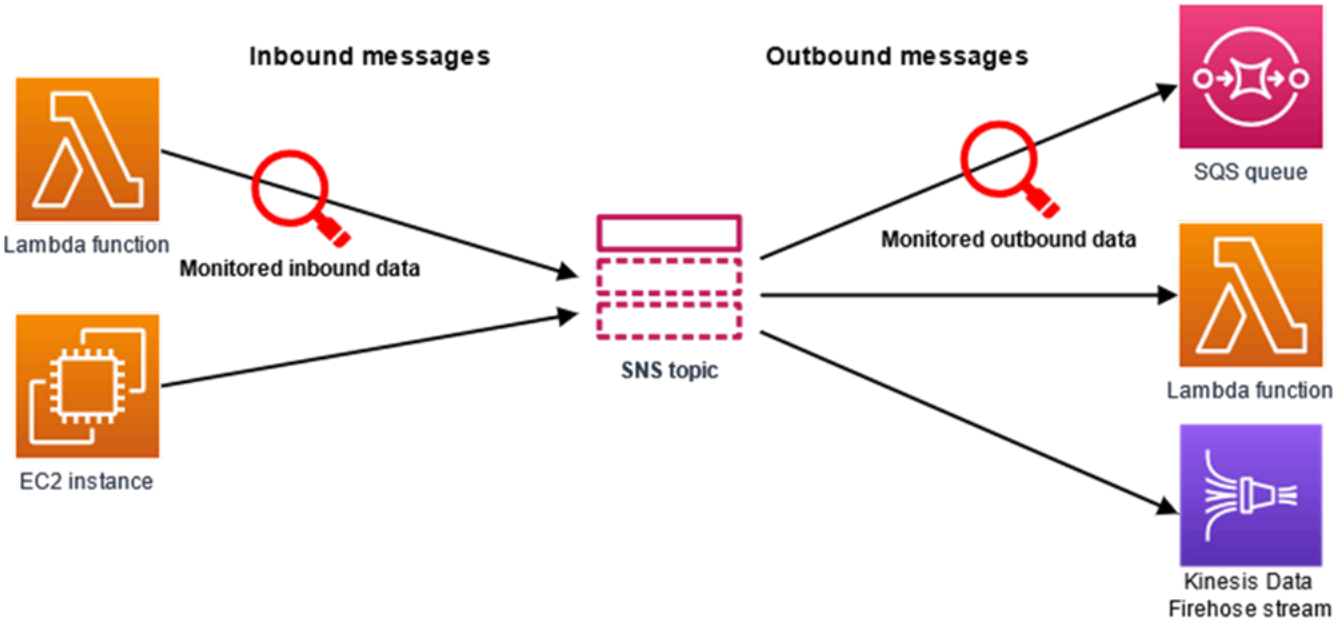
Comprendere le politiche di protezione SNS dei dati di Amazon

Argomenti

- Eliminazione delle politiche di protezione dei dati in Amazon SNS

Cosa sono le policy di protezione dei dati?

Amazon SNS utilizza le politiche di protezione dei dati per selezionare i dati sensibili che desideri scansionare e le azioni da intraprendere per proteggere tali dati dallo scambio tra i tuoi SNS argomenti Amazon. Per selezionare i dati sensibili di interesse, utilizza gli identificatori di dati. La protezione dei dati dei SNS messaggi di Amazon rileva quindi i dati sensibili utilizzando l'apprendimento automatico e il pattern matching. 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.

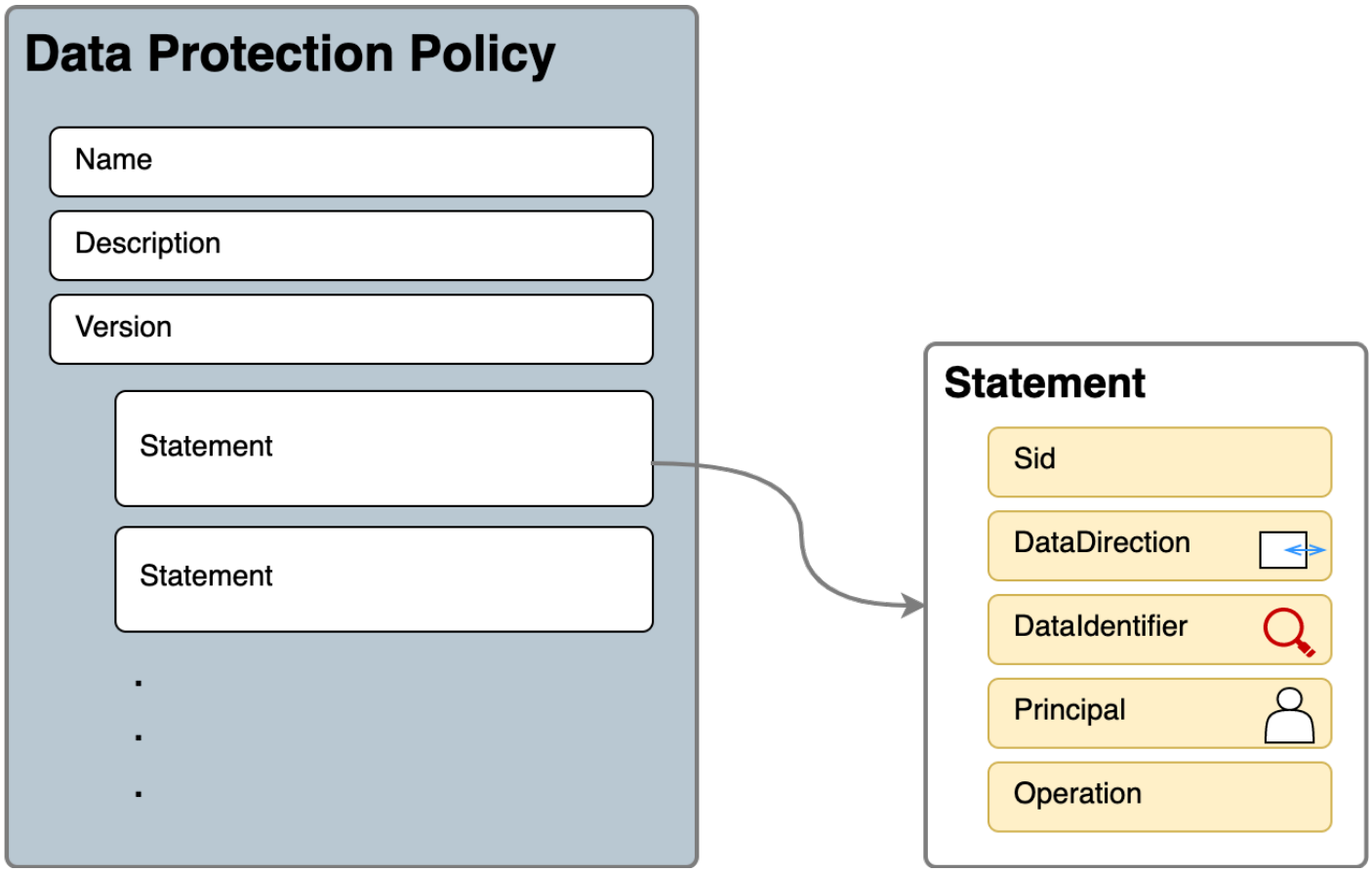


Come è strutturata una policy di protezione dei dati?

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 una sola politica di protezione dei dati per SNS argomento Amazon. La policy di protezione dei dati può includere una o più dichiarazioni di rifiuto o deidentificazione, ma solo una dichiarazione di verifica.

JSONproprietà per la politica 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": [
        ...
  ]
}
```

JSONproprietà per una dichiarazione politica

Una dichiarazione di policy definisce il contesto di rilevamento per l'operazione di protezione dei dati.

- Sid: (facoltativo) l'identificatore della dichiarazione.
- DataDirection— In entrata (per API le richieste di pubblicazione) o in uscita (per le consegne di notifiche) per quanto riguarda l'argomento Amazon. SNS
- DataIdentifier— I dati sensibili che l'SNSargomento Amazon deve cercare. Ad esempio, nome, indirizzo o numero di telefono.
- IAMPrincipal: il principale pubblicato sull'argomento o il IAM principale sottoscritto all'argomento.
- Funzionamento: l'azione successiva, Audit, De-identify (mascherare o oscurare) o Deny (bloccare), che l'SNSargomento Amazon esegue una volta trovati 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": {
        ...
    }
}
```

JSONproprietà per un'operazione relativa a una dichiarazione politica

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.
- Rifiuta: blocca la richiesta di SNS pubblicazione di Amazon o non riesce a recapitare il messaggio.

Come posso determinare i IAM principi della mia politica di protezione dei dati?

La protezione dei dati dei messaggi utilizza due IAM principi che interagiscono con AmazonSNS.

1. Publish API Principal (in entrata): il IAM principale autenticato che chiama Amazon. SNS `Publish` API
2. Subscription Principal (in uscita): l'account IAM principale autenticato che ha effettuato la chiamata durante la creazione dell'`Subscribe`APIabbonamento.

`SubscriptionPrincipal`È una proprietà di SNS abbonamento Amazon disponibile al pubblico che può essere recuperata da. `GetSubscriptionAttributes` API

```
{
  "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 relative alla politica di protezione dei dati in Amazon SNS

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 SNS sul blog Introducing message data protection for Amazon.

Argomenti

Operazione di verifica

L'operazione di audit campiona i messaggi in entrata tematici e registra i dati sensibili rilevati 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'SNSargomento Amazon trova dati sensibili nel payload.
2. NoFindingsDestination— La destinazione di registrazione quando l'SNSargomento Amazon non trova dati sensibili nel payload.

Puoi utilizzare quanto segue Servizi AWS in ciascuno dei seguenti tipi di destinazione di log:

- 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. Sono necessarie azioni aggiuntive per utilizzare il bucket Amazon S3 con SSE - KMS crittografia abilitata.

```
{
  "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 politica di protezione dei dati, devi aggiungere le seguenti autorizzazioni alla politica di IAM identità del IAM principale che chiama Amazon SNS `PutDataProtectionPolicy` API o a quella `CreateTopic` API con il `--data-protection-policy` parametro.

Destinazione della verifica	IAMautorizzazione
Predefinita	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies

Destinazione della verifica	IAMautorizzazione
	logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Sono necessarie azioni aggiuntive per utilizzare il bucket Amazon S3 con SSE - KMS crittografia abilitata.

```
{
  "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"
      ]
    }
  ]
}
```

Politica chiave richiesta per l'uso con - SSE KMS

Se utilizzi un bucket Amazon S3 come 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 scegli -S3, non è richiesta alcuna configurazione aggiuntiva. SSE Amazon S3 gestisce la chiave di crittografia.

Se scegli SSE -KMS, devi 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

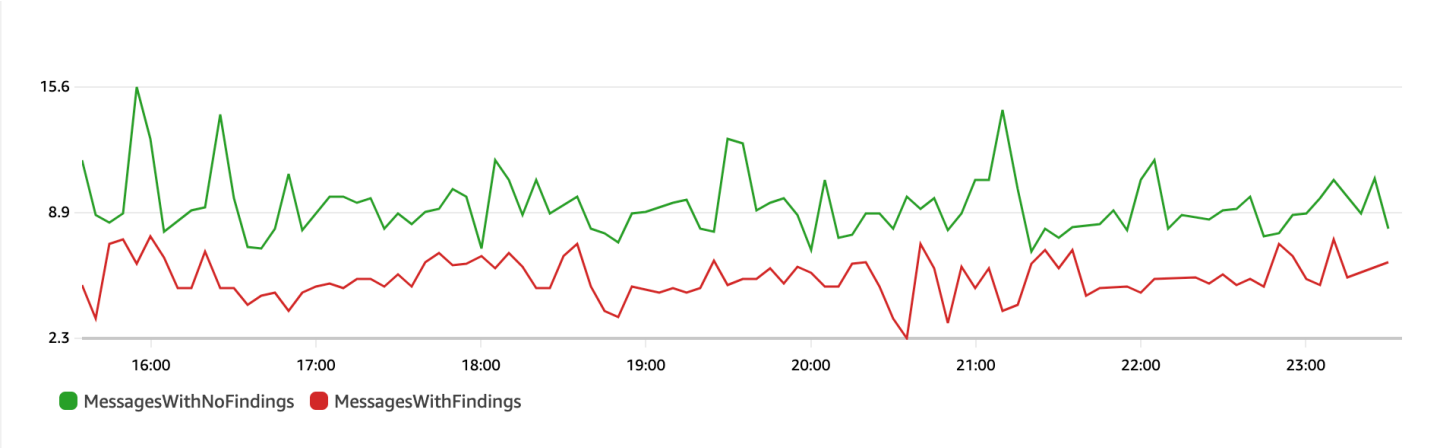
Esempi di registro della destinazione della verifica

Nell'esempio seguente, `callerPrincipal` viene utilizzato per identificare l'origine del contenuto sensibile e `messageID` viene utilizzato come riferimento per verificare la `Publish` API risposta.

```
{
  "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 CloudWatch `MessagesWithFindings` le `MessagesWithNoFindings` metriche.



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
A-Z, a-z e 0-9	Carattere alfanumerico
	Spazio
!	Punto esclamativo

Carattere di mascheramento supportato	Nome
$	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
`	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 resi anonimi dopo l'operazione di controllo e il `SNS:Publish` API chiamante riceve il seguente errore di parametro non valido quando l'intero messaggio è riservato.

```
Error code: AuthorizationError ...
```

Operazione di rifiuto

L'operazione Deny interrompe la `Publish` API richiesta 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 `SNS:Publish` API chiamante riceve un errore di autorizzazione.

```
Error code: AuthorizationError ...
```

In un messaggio in uscita, l'SNSargomento Amazon non recapita il messaggio all'abbonamento. 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 SNS dati di Amazon

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 SNS sul blog Introducing message data protection for Amazon.

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 API risposta hanno un codice di stato di»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 impedisce a un AWS account di ricevere messaggi contenenti`CreditCardNumber`.

```
{
  "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 politiche di protezione dei dati in Amazon SNS

Le politiche di protezione dei dati ti aiutano a salvaguardare i dati pubblicati nei tuoi SNS argomenti Amazon controllando, deidentificando (mascherando o oscurando) e negando (bloccando) le informazioni sensibili che si spostano tra applicazioni o. Servizi AWS Puoi utilizzare AWS API, AWS CLI AWS CloudFormation, o AWS Management Console creare politiche di protezione dei dati in AmazonSNS. È possibile definire una sola policy per SNS argomento Amazon. 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 politiche di protezione dei dati in Amazon SNS utilizzando API
- Creazione di politiche di protezione dei dati in Amazon SNS utilizzando CLI
- Creazione di politiche di protezione dei dati in Amazon SNS utilizzando CloudFormation
- Creazione di politiche di protezione dei dati in Amazon SNS utilizzando la console
- Creazione di politiche di protezione dei SNS dati di Amazon per proteggere i dati dei messaggi utilizzando SDK

Creazione di politiche di protezione dei dati in Amazon SNS utilizzando API

Il numero e le dimensioni delle SNS risorse Amazon in un AWS account sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di una politica di protezione dei dati utilizzando API

Puoi creare una policy di protezione SNS dei dati di Amazon utilizzando AWS API.

Per creare una politica di protezione dei dati insieme a un SNS argomento di Amazon (AWS API)

Usa la `DataProtectionPolicy` proprietà di un SNS argomento Amazon standard:

- `CreateTopic`

Per recuperare o creare una policy di protezione dei dati per un SNS argomento Amazon esistente ()AWS API

Chiamare una delle seguenti operazioni:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Creazione di politiche di protezione dei dati in Amazon SNS utilizzando CLI

Il numero e le dimensioni delle SNS risorse Amazon in un AWS account sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di politiche di protezione dei dati utilizzando il AWS CLI

Puoi creare una policy di protezione SNS dei dati di Amazon utilizzando AWS Command Line Interface.

Per creare una politica di protezione dei dati insieme a un SNS argomento di Amazon (AWS CLI)

Usa questa opzione per creare una nuova politica di protezione dei dati insieme a un SNS argomento Amazon standard:

- create-topic

Per creare o recuperare una politica di protezione dei dati per un SNS argomento Amazon esistente ()AWS CLI

Chiamare una delle seguenti operazioni:

- get-data-protection-policy
- put-data-protection-policy

Creazione di politiche di protezione dei dati in Amazon SNS utilizzando CloudFormation

Il numero e le dimensioni delle SNS risorse Amazon in un AWS account 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 SNS dei dati di Amazon utilizzando AWS CloudFormation.

Per creare una politica di protezione dei dati insieme a un SNS argomento di Amazon (CloudFormation)

Usa questa opzione per creare una nuova politica di protezione dei dati insieme a un SNS argomento Amazon standard:

- AWS::SNS: Argomento

Creazione di politiche di protezione dei dati in Amazon SNS utilizzando la console

Il numero e le dimensioni delle SNS risorse Amazon in un AWS account sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Per creare una politica di protezione dei dati insieme a un SNS argomento Amazon (Console)

Utilizza questa opzione per creare una nuova politica di protezione dei dati insieme a un SNS argomento Amazon standard.

1. Accedi alla SNSconsole Amazon.
2. Scegli un argomento o creane uno nuovo. Per maggiori dettagli sulla creazione di argomenti, consulta Creazione di un SNS argomento Amazon.

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.

   - Avanzato: definisci una politica di protezione dei dati personalizzata utilizzandoJSON.

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 non SNS risponderà alla API chiamata. 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. Per Destinazione di controllo, seleziona Servizi AWS a quale inviare i risultati dei risultati dell'audit e inserisci un nome di destinazione per ciascuna destinazione Servizio AWS che utilizzi. 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. Per Definire questa dichiarazione di anonimizzazione per, seleziona AWS gli account o IAM i principali a cui si applica questa dichiarazione di anonimizzazione. Puoi applicarlo a tutti gli AWS account o a conti o IAMentità specifici AWS (radici, ruoli o utenti dell'account) che utilizzano l'account IDs o l'entità. IAM ARNs Separare più ARNs elementi IDs o utilizzare una virgola (,).

Sono supportati IAMi seguenti principi:

- IAMprincipi contabili: ad esempio,. `arn:aws:iam::AWS-account-ID:root`
- IAMprincipi di ruolo: ad esempio,. `arn:aws:iam::AWS-account-ID:role/role-name`
- IAMuser principals: 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. Scegli i IAMprincipi che si applicano a questa dichiarazione di rifiuto. Puoi applicarla a tutti gli AWS account, a entità specifiche Account AWS o a IAMentità (ad esempio, account root, ruoli o utenti) che utilizzano l'account IDs o IAM l'entità. ARNs Separare più ARNs elementi IDs o utilizzare una virgola (,). Sono supportati IAMi seguenti principi:

- IAMprincipi contabili: ad esempio,. `arn:aws:iam::AWS-account-ID:root`
- IAMprincipi di ruolo: ad esempio,. `arn:aws:iam::AWS-account-ID:role/role-name`
- IAMuser principals: ad esempio,. `arn:aws:iam::AWS-account-ID:user/user-name`

iv. (Facoltativo) Continua ad aggiungere le dichiarazioni di rifiuto, se necessario.

Creazione di politiche di protezione dei SNS dati di Amazon per proteggere i dati dei messaggi utilizzando SDK

Il numero e le dimensioni delle SNS risorse Amazon in un AWS account sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Creazione di politiche di protezione dei dati utilizzando AWS SDK

Puoi creare una policy di protezione SNS dei dati di Amazon utilizzando AWS SDK.

Per creare una politica di protezione dei dati insieme a un SNS argomento di Amazon (AWS SDK)

Utilizza le seguenti opzioni per creare una nuova politica di protezione dei dati insieme a un SNS argomento Amazon standard:

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();
```

Per creare o recuperare una politica di protezione dei dati per un SNS argomento Amazon esistente ()AWS SDK

Utilizza le seguenti opzioni per creare o recuperare una nuova politica di protezione dei dati insieme a un SNS argomento standard di Amazon:

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 delle politiche di protezione dei dati in Amazon SNS

Puoi eliminare le policy di protezione dei SNS dati di Amazon utilizzando AWS API AWS CLI, AWS CloudFormation, o AWS Management Console.

Per informazioni generali sulle politiche di protezione SNS dei dati di Amazon, consultaComprendere le politiche di protezione SNS dei dati di Amazon.

Il numero e la dimensione delle risorse relative alla politica di protezione dei SNS dati di Amazon in un AWS account sono limitati. Per ulteriori informazioni, consulta Amazon SNS API throttling in. Riferimenti generali di AWS

Argomenti

- Eliminazione delle politiche di protezione dei dati tramite la console
- Eliminazione di una politica di protezione dei dati utilizzando una stringa vuota JSON
- Eliminazione di una politica di protezione dei dati utilizzando il AWS CLI

Eliminazione delle politiche di protezione dei dati tramite la console

Per eliminare una politica di protezione dei dati gestita utilizzando la console

1. Accedi alla SNSconsole Amazon.
2. Scegli l'argomento contenente la policy di protezione dei dati che desideri eliminare.
3. Scegli 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. Scegli Save changes (Salva modifiche).

Eliminazione di una politica di protezione dei dati utilizzando una stringa vuota JSON

È possibile eliminare una politica di protezione dei dati aggiornandola con una JSON stringa vuota.

Eliminazione di una politica di protezione dei dati utilizzando il 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 SNS di dati Amazon

Amazon SNS utilizza una combinazione di criteri e tecniche, tra cui l'apprendimento automatico e il pattern matching, per rilevare 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 SNS gestiti di Amazon offrono tipi di dati preconfigurati per proteggere dati finanziari, informazioni sanitarie personali (PHI) e 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 SNS sensibili di Amazon: credenziali
- Tipi di dati SNS sensibili di Amazon: dispositivi
- Tipi di dati SNS sensibili di Amazon: Financial
- Tipi di dati SNS sensibili di Amazon: informazioni sanitarie protette (PHI)
- Tipi di dati SNS sensibili di Amazon: informazioni di identificazione personale () PII

Cosa sono gli identificatori di dati gestiti?

Gli identificatori di dati SNS gestiti di Amazon sono progettati per rilevare un tipo specifico di dati sensibili, come numeri di carte di credito, chiavi di accesso AWS segrete o numeri di passaporto per un particolare paese o regione. Quando crei una politica di protezione dei dati, puoi SNS configurare Amazon in modo che utilizzi questi identificatori per analizzare i messaggi che riguardano l'argomento e intraprendere azioni quando vengono rilevati.

Amazon SNS è in grado di rilevare le seguenti categorie di dati sensibili utilizzando identificatori di dati gestiti:

- Credenziali, come chiavi private o chiavi di accesso AWS segrete
- Identificatori del dispositivo, come l'indirizzo IP o l'indirizzo MAC
- Informazioni finanziarie, ad esempio i numeri di carte di credito
- Informazioni sanitarie, PHI come l'assicurazione sanitaria o i numeri di identificazione medica
- Informazioni personali, PII ad esempio patenti di guida o numeri di previdenza sociale

All'interno di ogni categoria, Amazon SNS è in grado di rilevare diversi 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 abbina SNS automaticamente le varianti di parole chiave che non contengono lo spazio o contengono un carattere di sottolineatura (_) o un trattino (-) anziché lo spazio. In alcuni casi, Amazon amplia o abbrevia SNS anche una parola chiave per rispondere alle varianti comuni della parola chiave.

Identificatori di dati SNS gestiti da Amazon per tipi di dati sensibili

La tabella seguente elenca e descrive i tipi di credenziali, dispositivi, informazioni finanziarie, mediche e sanitarie personali (PHI) che Amazon SNS può rilevare utilizzando identificatori di dati gestiti. Questi si aggiungono a determinati tipi di dati che potrebbero anche essere considerati informazioni di identificazione personale (). PII

Gli identificatori di dati dipendenti dalla regione richiedono il nome dell'identificatore con un trattino e i codici a due lettere (3166-1 alpha-2). ISO Ad DriversLicense esempio, -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
IndividualTaxIdentificationNumber	Personale	US
InseeCode	Personale	FR
MedicareBeneficiaryNumber	Integrità	US
NationalDrugCode	Integrità	US
NationalIdentificationNumber	Personale	DE, ES, IT
NationalInsuranceNumber	Personale	GB

Identificatore	Categoria	Paesi/lingue
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

Identificatore	Categoria
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 SNS sensibili di Amazon: credenziali

La tabella seguente elenca e descrive i tipi di credenziali che Amazon SNS può rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e Regioni
AWS chiave di accesso segreta	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e Regioni
Aprire la chiave SSH privata	OpenSshPrivateKey	No	Qualsiasi
PGPchiave privata	PgpPrivateKey	No	Qualsiasi
chiave privata Public-Key Cryptography Standard () PKCS	PkcsPrivateKey	No	Qualsiasi
Chiave privata Pu TTY	PuttyPrivateKey	No	Qualsiasi

Identificatore di dati ARNs per i tipi di dati relativi alle credenziali

Di seguito sono elencati gli Amazon Resource Names (ARNs) per gli identificatori di dati che puoi aggiungere alle tue politiche di protezione dei dati.

Identificatore di dati di credenziali ARNs
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 SNS sensibili di Amazon: dispositivi

La tabella seguente elenca e descrive i tipi di identificatori di dispositivo che Amazon SNS può 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

Identificatore di dati ARNs per i tipi di dati del dispositivo

Di seguito sono elencati gli Amazon Resource Names (ARNs) per gli identificatori di dati che puoi aggiungere alle tue politiche di protezione dei dati.

Identificatore dei dati del dispositivo ARN
arn:aws:protezione dei dati: :aws:data-identifier/ IpAddress

Tipi di dati SNS sensibili di Amazon: Financial

La tabella seguente elenca e descrive i tipi di informazioni finanziarie che Amazon SNS può 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.	Ciò include: numeri di conto bancari internazionali (IBANs) composti da un massimo di 34 caratteri alfanumerici, inclusi elementi come il codice del paese.	Francia, Germania, Italia, Regno Unito, Spagna

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
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 di carte di credito hanno riservato ai 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 chiave per rilevare i numeri di conto bancari internazionali (IBANs) composti da un massimo di 34 caratteri alfanumerici, inclusi elementi come il codice del paese.

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			

Identificatore di dati ARNs per tipi di dati finanziari

Di seguito sono elencati gli Amazon Resource Names (ARNs) per gli identificatori di dati che puoi aggiungere alle tue politiche di protezione dei dati.

Identificatore di dati finanziari ARNs
arn:aws:protezione dei dati: :aws:data-identifier/ -DE BankAccountNumber
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 SNS sensibili di Amazon: informazioni sanitarie protette (PHI)

La tabella seguente elenca e descrive i tipi di informazioni sanitarie protette (PHI) che Amazon SNS può rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e Regioni
Numero di registrazione della Drug Enforcement Agency (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	US
Numero della tessera sanitaria (EHIC)	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	UE

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e Regioni
		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	
Numero di richiesta di assicurazione sanitaria (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 del sistema di codifica delle procedure comuni sanitarie (HCPCS)	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 del beneficiario Medicare () MBN	MedicareBeneficiaryNumber	mbi, medicare beneficiary	US
Codice nazionale sulle droghe () NDC	NationalDrugCode	national drug code, ndc	US
Identificatore nazionale del fornitore () NPI	NationalProviderId	hipaa, n.p.i, national provider, npi	US
Numero del Servizio Sanitario Nazionale (NHS)	NhsNumber	national health service, NHS	GB
Codice sanitario personale (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 numeri di assicurazione sanitaria e di identificazione medica, Amazon SNS richiede che una parola chiave si trovi in prossimità dei numeri. Ciò include i numeri della tessera sanitaria europea (UE, Finlandia), i numeri dell'assicurazione sanitaria (Francia), gli identificativi dei beneficiari Medicare (Stati Uniti), i numeri della previdenza nazionale (Regno Unito), NHS i numeri (Regno Unito) e i numeri sanitari personali (Canada).

La tabella seguente elenca le parole chiave che Amazon SNS riconosce per paesi e regioni specifici.

Paese o regione	Parole chiave
Canada	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé

Paese o regione	Parole chiave
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, sairausva kuutuskortti, 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
Francia	carte d'assuré social, carte vitale, insurance card
UK	servizio sanitario nazionale, NHS

Paese o regione	Parole chiave
US	mbi, medicare beneficiary

Identificatore di dati ARNs per i tipi di dati relativi alle informazioni sanitarie protette () PHI

Di seguito sono elencati gli identificatori di dati Amazon Resource Names (ARNs) che possono essere utilizzati nelle politiche di protezione PHI dei dati.

PHIidentificatore di dati ARNs
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 SNS sensibili di Amazon: informazioni di identificazione personale () PII

La tabella seguente elenca e descrive i tipi di informazioni di identificazione personale (PII) che Amazon SNS può 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
Code de Endereçamento Postal () CEP	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brasile
Catastro Nacional da Pessoa Jurídica () CNPJ	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	Brasile
Catastro de Pessoas Físicas () CPF	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de	–	Brasile

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
		pessoa física, cadastro de pessoa fisica, cpf		
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	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
Identificazione del singolo contribuente	Individua lTaxIdentification Number	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	US
Istituto nazionale di statistica e studi economici () 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.	Ciò include gli identificatori del Documento Nacional de Identidad (DNI) (Spagna), i codici del Codice fiscale (Italia) e i numeri della carta d'identità nazionale (tedesco).	Germania, Italia, Spagna
Numero di previdenza nazionale () NINO	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
Numero di identificazione fiscale 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
Numero di previdenza sociale () 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
Numero di previdenza sociale (SSN)	Ssn	Spagna: número de la seguridad social, n. di previdenza sociale, numero di previdenz a 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.	Ciò include TIN (Francia), Steueridentifikationsnummer (Germania), (Spagna) e CIF (Regno Unito). TRN UTR	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 una parola chiave non sia obbligatoria, il rilevamento richiede che l'indirizzo includa il nome di una città o di un luogo e un codice postale. ZIP	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 Global Positioning System (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS è in grado di rilevare GPS le coordinate se le coordinate di latitudine e longitudine sono memorizzate come coppia e sono in formato Decimal Degrees (DD), ad esempio 41,948614, -87,655311. Support non include le coordinate in formato Degrees Decimal Minutes (DDM), ad esempio 41°56.9168'N 87°39.3187'W, o Degrees, Minutes, Seconds (), ad esempio 41°56'55.0104"N 87°39'19.1196"W. DMS	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
Numero di identificazione 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 VINs che consistono in una sequenza di 17 caratteri e aderisce 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 della patente di guida, Amazon SNS richiede che una parola chiave si trovi in prossimità dei numeri. La tabella seguente elenca le parole chiave che Amazon SNS riconosce 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 identificazione nazionali, Amazon SNS richiede che una parola chiave sia in prossimità dei numeri. Ciò include gli identificatori del Documento Nacional de Identidad (DNI) (Spagna), i codici dell'Istituto nazionale francese di statistica e studi economici (INSEE), i numeri delle carte d'identità nazionali tedesche e i numeri del Registro Geral (RG) (Brasile).

La tabella seguente elenca le parole chiave che Amazon SNS riconosce 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 number, national id number, nationalid#,

Paese o regione	Parole chiave
	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 che una parola chiave sia in prossimità dei numeri. La tabella seguente elenca le parole chiave che Amazon SNS riconosce 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, geburtsda tum, passport, passports, reisepass, reisepass– nr, reisepassnummer
Italia	italian passport number, numéro passeport , numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaport o
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 codice identificativo e di riferimento dei contribuenti, Amazon SNS richiede che una parola chiave si trovi in prossimità dei numeri. La tabella seguente elenca le parole chiave che Amazon SNS riconosce 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.)

Identificatore di dati ARNs per informazioni di identificazione personale () PII

La tabella seguente elenca gli Amazon Resource Names (ARNs) per gli identificatori di dati che puoi aggiungere alle tue politiche di protezione dei dati.

PIIidentificatore di dati ARNs
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

PIIidentificatore di dati ARNs
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

PIIidentificatore di dati ARNs
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

PIIidentificatore di dati ARNs
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

PIIidentificatore di dati ARNs
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

Gli identificatori di dati personalizzati (CDIs) consentono di definire espressioni regolari personalizzate che possono essere utilizzate nella politica di protezione dei dati. Utilizzando identificatori di dati personalizzati, puoi indirizzare le informazioni di identificazione personale specifiche dell'azienda (PII) a casi d'uso che gli identificatori di dati gestiti non sono in grado di fornire. Ad esempio, puoi utilizzare un identificatore di dati personalizzato per cercare dipendenti specifici dell'azienda. IDs Gli identificatori di dati personalizzati possono essere utilizzati insieme agli identificatori di dati gestiti.

Argomenti

- Cosa sono gli identificatori di dati personalizzati?
- Utilizzo degli identificatori di dati personalizzati nella policy di protezione dei dati
- Vincoli degli identificatori di dati personalizzati

Cosa sono gli identificatori di dati personalizzati?

Gli identificatori di dati personalizzati (CDIs) consentono di definire espressioni regolari personalizzate che possono essere utilizzate nella politica di protezione dei dati. Utilizzando identificatori di dati personalizzati, puoi indirizzare le informazioni di identificazione personale specifiche dell'azienda (PII) a casi d'uso che gli identificatori di dati gestiti non sono in grado di fornire. Ad esempio, puoi utilizzare un identificatore di dati personalizzato per cercare dipendenti specifici dell'azienda. IDs 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 politica di protezione dei dati indica all'SNSargomento Amazon di rilevare i payload che trasportano dipendenti specifici dell'aziendaIDs, quindi di mascherarli IDs 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 SNS personalizzati di Amazon 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: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - RegEx caratteri riservati: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- 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 politica di protezione dei dati per ogni SNS argomento di Amazon.

Consegna dei SNS messaggi su Amazon

Questo argomento descrive come Amazon SNS gestisce la consegna dei messaggi in vari scenari. Imparerai a conoscere la consegna dei messaggi non elaborati, in cui Amazon SNS recapita i messaggi nel loro formato originale e non modificato all'endpoint. Scoprirai anche come inviare messaggi da un SNS argomento Amazon a una SQS coda Amazon in un altro modo Account AWS, fornendo informazioni sulla messaggistica tra account.

Questo argomento fornisce informazioni sulla consegna dei SNS messaggi Amazon a una SQS coda Amazon o a una funzione Lambda in Regioni AWS diversi modi, su come funziona la consegna tra regioni e sulle considerazioni coinvolte.

Inoltre, imparerai come monitorare e interpretare lo stato di consegna dei messaggi, il che fornisce informazioni cruciali sul fatto che i messaggi siano stati recapitati correttamente o che si siano verificati problemi. Nel caso in cui il recapito dei messaggi non vada a buon fine, conoscerai la procedura di nuovo tentativo di recapito dei messaggi, incluso il modo in cui Amazon tenta SNS automaticamente di recapitare i messaggi per garantire che raggiungano le destinazioni previste. Questo argomento illustra anche l'uso delle code di lettere non scritte per acquisire messaggi che non è stato possibile recapitare dopo più tentativi, consentendoti di analizzare e risolvere questi errori in modo efficace.

Argomenti

Consegna di messaggi non SNS elaborati su Amazon

Per evitare che gli endpoint Amazon Data FirehoseSQS, Amazon e HTTP/S elaborino la JSON formattazione dei messaggi, Amazon SNS consente la consegna dei messaggi non elaborati:

- Quando abiliti la consegna di messaggi non elaborati per Amazon Data Firehose o gli SQS endpoint Amazon, tutti SNS i metadati Amazon vengono rimossi dal messaggio pubblicato e il messaggio viene inviato così com'è.
- Quando abiliti la consegna di messaggi non elaborati per gli endpoint HTTP /S, l'HTTPintestazione `x-amz-sns-rawdelivery` con il valore impostato su `true` viene aggiunta al messaggio, a indicare che il messaggio è stato pubblicato senza formattazione. JSON
- Quando si abilita il recapito di messaggi non elaborati per gli endpoint HTTP /S, vengono recapitati 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 il recapito di messaggi non elaborati utilizzando un AWS SDK, è necessario utilizzare l'`SetSubscriptionAttribute`APIazione e impostare il valore dell'`RawMessageDelivery`attributo su. `true`

Abilitazione della consegna di messaggi non elaborati utilizzando la AWS Management Console

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Argomenti, scegli un argomento sottoscritto a un endpoint Firehose, SQS Amazon HTTP o /S.
4. Nella ***`MyTopic`*** nella sezione Abbonamento, scegli un abbonamento e scegli Modifica.
5. Nella sezione Modifica ***`EXAMPLE1-23bc-4567-d890-ef12g3hij456`***nella sezione Dettagli, scegli Abilita il recapito dei messaggi non elaborati.
6. Scegli Save changes (Salva modifiche).

Esempi di formati di messaggi

Negli esempi seguenti, lo stesso messaggio viene inviato due volte alla stessa SQS coda Amazon. 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 recapito dei messaggi non elaborati per SQS gli abbonamenti Amazon

Amazon SNS supporta la consegna degli attributi dei messaggi, che consentono di fornire elementi di metadati strutturati, come timestamp, dati geospaziali, firme e identificatori, sul messaggio. Per SQS gli abbonamenti Amazon con Raw Message Delivery abilitato, è possibile inviare un massimo di 10 attributi del 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 SQS abbonamenti Amazon con Raw Message Delivery abilitata, trattandoli come errori lato client.

Invio SNS di messaggi Amazon a una SQS coda Amazon in un altro account

Questo documento descrive come pubblicare una notifica su un SNS argomento di Amazon con uno o più abbonamenti ad Amazon SQS Queues 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 Fanout SNS delle notifiche Amazon alle SQS code Amazon per l'elaborazione asincrona). 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 SQS coda Amazon ha un volume elevato di messaggi, consigliamo al proprietario della coda di creare l'abbonamento.

Argomenti

Creazione della sottoscrizione da parte del proprietario della coda

L'account che ha creato la SQS coda Amazon 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 SNSconsole Amazon.

2. Nel pannello di navigazione, scegliere Argomenti.
3. Selezionare un argomento, quindi scegliere Edit (Modifica).
4. Nella sezione Modifica ***MyTopic***pagina, espandi la sezione Politica 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 dell'account di delegare l'autorizzazione al relativo IAM utente/ruolo. Solo l'account root o gli utenti amministratori saranno autorizzati a richiamare `sns:Subscribe`. L'IAMutente/ruolo deve inoltre consentire `sns:subscribe` alla propria coda di iscriversi.
6. Scegli Save changes (Salva modifiche).

   Un utente con le credenziali per l'account `111122223333` può iscriversi a. MyTopic

Fase 2: Per aggiungere un abbonamento Amazon SQS Queue a un argomento in un altro argomento Account AWS utilizzando il AWS Management Console

Prima di iniziare, assicurati ARNs di disporre dell'argomento e della coda e di aver autorizzato l'argomento a inviare messaggi alla coda.

1. Accedi alla SQSconsole Amazon.
2. Nel pannello di navigazione, scegliere Code.

3. Dall'elenco delle code, scegli la coda per iscriverti all'argomento AmazonSNS.
4. Scegli l'SNSargomento Abbonati ad Amazon.
5. Dal menu Specificare un SNS argomento Amazon disponibile per questa coda, scegli l'SNSargomento Amazon per la tua coda.
6. Scegli Inserisci l'SNSargomento ARN Amazon, quindi inserisci il nome della risorsa Amazon dell'argomento (ARN).
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 utilizzi l'`Subscribe`azione, Amazon SNS invia una conferma dell'abbonamento alla coda. L'abbonamento viene visualizzato nella SNS console Amazon, con l'ID di abbonamento impostato su In attesa di conferma.

Per confermare l'abbonamento, un utente con l'autorizzazione a leggere i messaggi dalla coda deve recuperare la conferma URL dell'abbonamento e il proprietario dell'abbonamento deve confermare l'abbonamento utilizzando la conferma dell'abbonamento. URL Fino alla conferma della sottoscrizione, nessuna delle notifiche pubblicate nell'argomento viene inviata alla coda. Per confermare l'abbonamento, puoi utilizzare la SQS console Amazon o l'`ReceiveMessage`azione.

> **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 Passaggio 2: autorizza l'SNSargomento Amazon a inviare messaggi alla SQS coda Amazon.

Fase 1: Per aggiungere un abbonamento Amazon SQS Queue a un argomento in un altro, Account AWS utilizzare il AWS Management Console

Prima di iniziare, assicurati di disporre delle ARNs impostazioni relative all'argomento e alla coda e di aver autorizzato l'argomento a inviare messaggi alla coda.

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegliere Create subscription (Crea sottoscrizione).
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
   a. Per Argomento ARN, inserisci ARN l'argomento.
   b. Per Protocol, scegli Amazon SQS.
   c. Per Endpoint, entra nella ARN coda.
   d. Scegli Crea sottoscrizione.

> **Note**
>
> - Per poter comunicare con il servizio, la coda deve disporre delle autorizzazioni per Amazon. SNS

Di seguito è riportato un esempio di dichiarazione politica che consente all'SNSargomento Amazon di inviare un messaggio alla SQS coda Amazon.

```
{
   "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"
```

```
        }
    }
}
```

Passaggio 2: Per confermare un abbonamento utilizzando il AWS Management Console

1. Accedi alla SQSconsole Amazon.
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 Dettagli del messaggio, trova e annota il URL valore Subscribe. Questo è il collegamento di sottoscrizione (esempio qui di seguito). Per ulteriori dettagli sulla convalida dei API token, consulta `ConfirmSubscription`Amazon SNS API Reference.

      ```
      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. URLDeve essere passato dal proprietario della coda al proprietario dell'abbonamento. Il proprietario dell'abbonamento deve URL inserirlo nella SNSconsole Amazon.
5. Accedi come proprietario dell'abbonamento alla SNSconsole Amazon Il proprietario dell'abbonamento esegue 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 XML risposta, 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 l'argomento abbonamento nella SNS console Amazon, puoi vedere che il messaggio di conferma in sospeso è stato sostituito dall'abbonamento ARN nella colonna ID abbonamento.

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 SNS di messaggi Amazon a una SQS coda o AWS Lambda funzione Amazon in un'altra regione

Amazon SNS supporta le consegne interregionali, sia per le regioni abilitate per impostazione predefinita che per le regioni opt-in. Per l'elenco aggiornato delle AWS regioni SNS supportate da Amazon, incluse le regioni che richiedono l'iscrizione, consulta gli endpoint e le quote di Amazon Simple Notification Service nel.Riferimenti generali di Amazon Web Services

Amazon SNS supporta l'invio interregionale di notifiche alle SQS code e alle AWS Lambda funzioni di Amazon. Quando una delle regioni è una regione che accetta l'iscrizione, devi specificare un diverso SNS servizio Amazon principale nella politica della risorsa sottoscritta.

Il comando Amazon SNS subscription deve essere eseguito nella regione in cui SNS è ospitato Amazon, nella regione corrispondente. Ad esempio, se Amazon SNS si trova nell'account «A» nella regione us-east-1 e la funzione Lambda è nell'account «B» nella regione us-east-2, il comando CLI subscription deve essere eseguito nell'account «A» nella regione us-east-1.

Regioni con consenso esplicito

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 (UAE)	me-central-1

Per informazioni sull'attivazione di una regione con consenso esplicito, consulta Gestione AWS delle regioni in.Riferimenti generali di Amazon Web Services

Quando utilizzi Amazon SNS per recapitare messaggi da regioni che hanno aderito all'iscrizione a regioni abilitate per impostazione predefinita, devi modificare la politica delle risorse creata per la coda. Sostituire l'entità `sns.amazonaws.com` con `sns.<opt-in-region>.amazonaws.com`. Per esempio:

- Per iscrivere una SQS coda Amazon negli Stati Uniti orientali (Virginia settentrionale) a un SNS argomento Amazon in Asia Pacifico (Hong Kong), modifica il principale nella politica di 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 in più 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 SNS argomento Amazon in una regione opt-in (af-south-1) di effettuare la consegna a una SQS coda

Amazon 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 SNS argomento Amazon in Asia Pacifico (Hong Kong), modifica il principale nella politica della AWS Lambda 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	

Tipo di consegna tra regioni	Supportato/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 SNS messaggi Amazon

Amazon SNS fornisce supporto per registrare lo stato di consegna dei messaggi di notifica inviati agli argomenti con i seguenti SNS endpoint Amazon:

- 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 recapitato all'SNSendpoint Amazon.
- Identificazione della risposta inviata dall'SNSendpoint Amazon ad AmazonSNS.
- Determinazione del tempo di permanenza del messaggio (il tempo che intercorre tra il timestamp di pubblicazione e il momento prima della consegna a un endpoint Amazon). SNS

Per configurare gli attributi dell'argomento per lo stato di consegna dei messaggi, puoi utilizzare AWS software development kits () AWS Management Console SDKs, query o. API AWS CloudFormation

Argomenti

Configurazione della registrazione dello stato di consegna utilizzando la AWS Management Console

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti), selezionare un argomento quindi scegliere Edit (Modifica).
4. Nella sezione Modifica ***MyTopic***pagina, espandi la sezione Registrazione dello stato della 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 IAMruoli, effettuate una delle seguenti operazioni:

   - Per scegliere un ruolo di servizio esistente dal tuo account, scegli Usa il ruolo di servizio esistente, quindi specifica IAM i ruoli per le consegne riuscite e non riuscite.

   - Per creare un nuovo ruolo di servizio nel tuo account, scegli Crea nuovo ruolo di servizio, scegli Crea nuovi ruoli per definire i IAM ruoli per le consegne riuscite e non riuscite nella IAM console.

     Per consentire ad Amazon l'accesso in SNS scrittura per utilizzare CloudWatch Logs per tuo conto, scegli Consenti.
8. Scegli Save changes (Salva 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 utilizzando il AWS SDKs

AWS SDKsForniscono APIs in diverse lingue per l'utilizzo degli attributi dello stato di consegna dei messaggi con AmazonSNS.

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 invio corretto dei messaggi per un SNS argomento Amazon sottoscritto a un HTTP endpoint.
- `HTTPSuccessFeedbackSampleRate`— Indica la percentuale di messaggi riusciti da campionare per un SNS argomento Amazon sottoscritto a un HTTP endpoint.
- `HTTPFailureFeedbackRoleArn`— Indica lo stato di mancato recapito dei messaggi per un SNS argomento Amazon sottoscritto a un HTTP endpoint.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— Indica lo stato di invio corretto dei messaggi per un SNS argomento Amazon sottoscritto a un endpoint Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate`— Indica la percentuale di messaggi riusciti da campionare per un SNS argomento Amazon sottoscritto a un endpoint Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn`— Indica lo stato di invio non riuscito dei messaggi per un SNS argomento Amazon sottoscritto a un endpoint Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— Indica lo stato di invio corretto dei messaggi per un SNS argomento Amazon sottoscritto a un endpoint Lambda.
- `LambdaSuccessFeedbackSampleRate`— Indica la percentuale di messaggi riusciti da campionare per un SNS argomento Amazon sottoscritto a un endpoint Lambda.

- `LambdaFailureFeedbackRoleArn`— Indica lo stato di invio non riuscito dei messaggi per un SNS argomento Amazon sottoscritto a un endpoint Lambda.

Endpoint applicazione piattaforma

- `ApplicationSuccessFeedbackRoleArn`— Indica lo stato di invio corretto dei messaggi per un SNS argomento Amazon sottoscritto a un endpoint AWS dell'applicazione.
- `ApplicationSuccessFeedbackSampleRate`— Indica la percentuale di messaggi riusciti da campionare per un SNS argomento Amazon sottoscritto a un endpoint AWS applicativo.
- `ApplicationFailureFeedbackRoleArn`— Indica lo stato di mancato recapito dei messaggi per un SNS argomento Amazon sottoscritto a un endpoint AWS dell'applicazione.

> **Note**
>
> Oltre a poter configurare gli attributi degli argomenti per lo stato di consegna dei messaggi di notifica inviati agli endpoint delle SNS applicazioni Amazon, puoi anche configurare gli attributi dell'applicazione per lo stato di consegna dei messaggi di notifica push inviati ai servizi di notifica push. Per ulteriori informazioni, consulta Using Amazon SNS Application Attributes for Message Delivery Status.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— Indica lo stato di invio corretto dei messaggi per un SNS argomento Amazon sottoscritto a un SQS endpoint Amazon.
- `SQSSuccessFeedbackSampleRate`— Indica la percentuale di messaggi riusciti da campionare per un SNS argomento Amazon sottoscritto a un SQS endpoint Amazon.
- `SQSFailureFeedbackRoleArn`— Indica lo stato di invio non riuscito del messaggio per un SNS argomento Amazon sottoscritto a un SQS endpoint Amazon.

> **Note**
>
> `<ENDPOINT>FailureFeedbackRoleArn`Gli attributi `<ENDPOINT>SuccessFeedbackRoleArn` and vengono utilizzati per consentire ad Amazon l'accesso in SNS 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>FailureFeedbackRoleArnattributo, tutte le consegne di messaggi non riuscite generano CloudWatch log.

AWS SDKesempi per configurare gli attributi degli argomenti

I seguenti esempi di codice mostrano come utilizzareSetTopicAttributes.

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 API i dettagli, vedere SetTopicAttributesin AWS CLI Command Reference.

Java

SDKper 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 API i dettagli, vedi SetTopicAttributes AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere SetTopicAttributesin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento SetTopicAttributes AWSSDKa Kotlin API.

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 API i dettagli, vedi SetTopicAttributes AWS SDK for PHPAPIReference.

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi SetTopicAttributes AWS SDK for RubyAPIReference.

SAP ABAP

SDKper SAP ABAP

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

```
TRY.
    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 API i dettagli, vedi SetTopicAttributesSAPABAPAPIcome riferimento.AWS SDK

Configurazione della registrazione dello stato di consegna utilizzando AWS CloudFormation

Per configurare `DeliveryStatusLogging` l'utilizzo AWS CloudFormation, usa un YAML modello JSON or per creare uno AWS CloudFormation stack. Per ulteriori informazioni, consulta la `DeliveryStatusLogging` proprietà della `AWS::SNS::Topic` risorsa nella Guida per l' AWS CloudFormation utente. Di seguito sono riportati alcuni esempi di AWS CloudFormation JSON modelli YAML per creare un nuovo argomento o aggiornare un argomento esistente con tutti `DeliveryStatusLogging` gli attributi per il SQS protocollo Amazon.

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 recapito dei SNS messaggi Amazon

Amazon SNS definisce una politica di consegna per ogni protocollo di consegna. La politica di consegna definisce in che modo Amazon SNS ritenta la 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 interrompe il tentativo di consegna e scarta

il messaggio, a meno che all'abbonamento non sia allegata una coda di lettere non scritte. Per ulteriori informazioni, consulta Code di lettere non SNS ricevute su Amazon.

Argomenti

Protocolli e policy di consegna

> **Note**
>
> - Ad eccezione dei HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S supporti per le politiche personalizzate. Per informazioni, consulta Creazione di una politica di consegna HTTP /S.
> - Amazon SNS applica il jittering ai nuovi tentativi di consegna. Per ulteriori informazioni, vedere il post Backoff esponenziale e Jitter in Blog architettura AWS .
> - Il tempo totale di ripetizione della policy per un endpoint HTTP /S non può essere superiore a 3.600 secondi. Questo è un limite fissato che non può essere modificato.

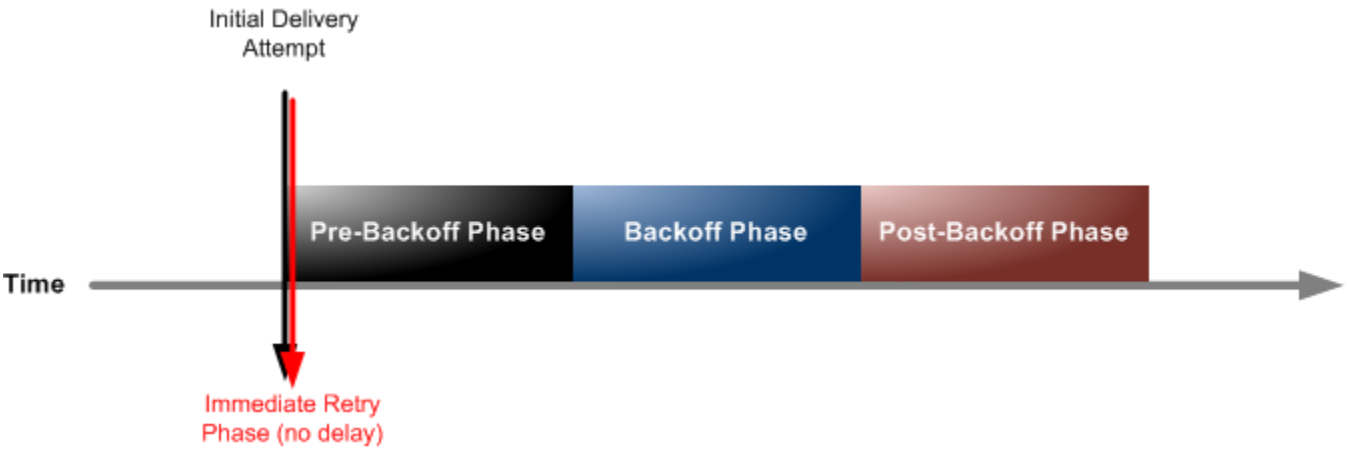
Tipo di endpoint	Protocolli di consegna	Fase dei nuovi tentativi immediati (nessun ritardo)	Fase di prebackoff	Fase di backoff	Fase di postbackoff	Totale tentativi
AWS endpoint gestiti	Amazon Data Firehose[1] AWS Lambda	3 volte, senza ritardo	2 volte, 1 secondo di distanza	10 volte, con backoff esponenziale, da 1 secondo a 20 secondi	100.000 volte, 20 secondi di distanza	100.015 volte, oltre 23 giorni

Tipo di endpoint	Protocolli di consegna	Fase dei nuovi tentativi immediati (nessun ritardo)	Fase di prebackoff	Fase di backoff	Fase di postbackoff	Totale tentativi
	Amazon SQS					
Endpoint gestiti dal cliente	SMTP SMS Push per dispositivi mobili	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

[1] Per gli errori di limitazione con il protocollo Firehose, Amazon SNS utilizza la stessa politica di consegna 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 tentare una serie di nuovi 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 politica di consegna HTTP /S

Puoi utilizzare una politica di consegna e le sue quattro fasi per definire in che modo Amazon SNS ritenta la consegna dei messaggi agli endpoint HTTP /S. Amazon ti SNS consente di ignorare la politica di ripetizione dei tentativi predefinita per gli HTTP endpoint quando, ad esempio, potresti voler personalizzare la politica in base alla capacità del tuo HTTP server.

Puoi impostare HTTP/S delivery policy as a JSON object at the subscription or topic level. When you define the policy at the topic level, it applies to all HTTP/S gli abbonamenti associati all'argomento. Per impostare la politica di consegna a livello di abbonamento, puoi utilizzare l'`SetSubscriptionAttributes`APIazione `Subscribe`o. Per impostare la politica di consegna a livello di argomento, puoi utilizzare l'`SetTopicAttributes`APIazione `CreateTopic`o. In alternativa, puoi anche utilizzare la risorsa AWS::SNS: :Subscription nei tuoi AWS CloudFormation modelli.

È necessario personalizzare la politica di distribuzione in base al HTTP/S server's capacity. You can set the policy as a topic attribute or a subscription attribute. If all HTTP/S subscriptions in your topic target the same HTTP/S server, we recommend that you set the delivery policy as a topic attribute, so that it remains valid for all HTTP/S subscriptions in the topic. Otherwise, you must compose a delivery policy for each HTTP/S subscription in your topic, according the capacity of the HTTP/S server a cui è destinata la politica.

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 `text/plain; charset=UTF-8` su. Amazon

SNS ti consente di ignorare la politica di richiesta predefinita. Per `headerContentType` supportata e i vincoli, consulta la tabella sottostante.

L'JSONoggetto seguente rappresenta una politica di consegna che indica SNS ad Amazon di riprovare un tentativo di consegna HTTP /S fallito, nel modo seguente:

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 questo esempio di politica di consegna, Amazon SNS effettua un totale di 50 tentativi prima di scartare il messaggio. Per conservare il messaggio dopo aver esaurito i tentativi specificati nella politica di recapito, configura l'abbonamento in modo da spostare i messaggi non recapitabili in una coda di lettere non recapitate (). DLQ Per ulteriori informazioni, consulta Code di lettere non SNS ricevute su Amazon.

> **Note**
>
> Questa politica di consegna impone inoltre SNS ad Amazon di limitare le consegne a non più di 10 al secondo, utilizzando la proprietà. `maxReceivesPerSecond` 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 politica di recapito sostituisce il tipo di contenuto predefinito per la notifica /S a. HTTP `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

Policy	Descrizione	Vincolo
`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
`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 inviata 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

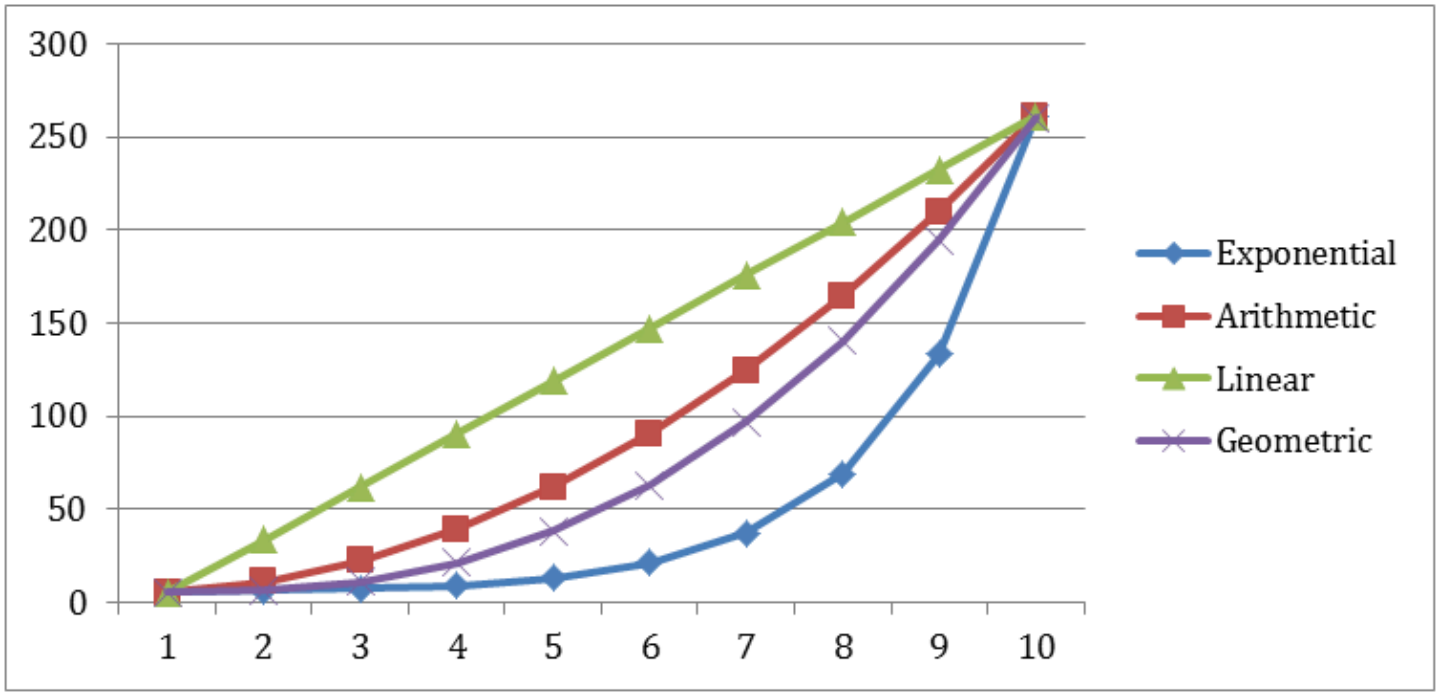
Amazon SNS utilizza la seguente formula per calcolare il numero di nuovi tentativi nella fase di backoff:

```
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 SNS utilizzato da Amazon per calcolare i ritardi associati a tutti i tentativi tra il primo e l'ultimo tentativo 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 di lettere non SNS ricevute su Amazon

Una coda di lettere morte è una coda Amazon a SQS cui un SNS abbonamento Amazon può indirizzare 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 un abbonamento e Tentativi di recapito dei SNS messaggi Amazon.

> **Note**
>
> - L'SNSabbonamento Amazon e la SQS coda Amazon devono appartenere allo stesso AWS account e alla stessa regione.
> - Per un FIFOargomento, puoi utilizzare una SQS coda Amazon come coda di lettere non scritte per l'abbonamento Amazon. SNS FIFOgli abbonamenti agli argomenti utilizzano le FIFO code e gli abbonamenti agli argomenti standard utilizzano le code standard.
> - Per utilizzare una SQS coda Amazon crittografata come coda di lettere non scritte, devi utilizzare una politica personalizzata KMS con una chiave che garantisca al SNS servizio Amazon l'accesso principale alle azioni. AWS KMS API Per ulteriori informazioni, consulta Protezione dei SNS dati Amazon con la crittografia lato server questa guida e Protection

Amazon SQS Data Using Server-Side Encryption (SSE) e AWS KMS la Amazon Simple Queue Service Developer Guide.

Argomenti

Perché le consegne dei messaggi non riescono?

In generale, la consegna dei messaggi non riesce quando Amazon non SNS riesce ad 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 oltre il numero di tentativi specificato dalla politica di ripetizione corrispondente, Amazon SNS scarta il messaggio, a meno che non sia allegata una coda di lettere morte all'abbonamento. Le consegne non riuscite non modificano lo stato delle sottoscrizioni. Per ulteriori informazioni, consulta Tentativi di recapito dei SNS messaggi Amazon.

Errori lato client

Gli errori sul lato client possono verificarsi quando Amazon SNS ha metadati di abbonamento obsoleti. Questi errori si verificano in genere quando un proprietario elimina l'endpoint (ad esempio, una funzione Lambda sottoscritta a un argomento di SNS Amazon) o quando un proprietario modifica la politica allegata all'endpoint sottoscritto in modo da impedire ad SNS Amazon di recapitare messaggi all'endpoint. Amazon SNS non riprova il recapito del messaggio che non va a buon fine a causa di un errore sul lato client.

Errori lato server

Gli errori sul 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 sul lato server, Amazon SNS ritenta le consegne non riuscite utilizzando una funzione di backoff lineare o esponenziale. In caso di errori lato

server causati da endpoint AWS gestiti supportati da Amazon oppureSQS, AWS Lambda Amazon SNS riprova la consegna fino a 100.015 volte, nell'arco di 23 giorni.

Anche gli endpoint gestiti dal cliente (come HTTP SMTPSMS, o mobile push) possono causare errori sul lato server. Amazon SNS riprova la consegna anche a questi tipi di endpoint. Sebbene gli HTTP endpoint supportino politiche di riprova definite dal cliente, Amazon SNS imposta una politica interna di ripetizione dei tentativi di consegna per 50 volte nell'arco di 6 ore SMTPSMS, per gli endpoint push mobili.

Come funzionano le code DLQ?

Una coda di lettere non scritte è allegata a un SNS abbonamento Amazon (anziché a un argomento) perché le consegne dei messaggi avvengono a livello di abbonamento. In questo modo è possibile identificare più facilmente l'endpoint di destinazione originale per ogni messaggio.

Una coda di lettere non scritte associata a un SNS abbonamento Amazon è una normale coda Amazon. SQS Per ulteriori informazioni sul periodo di conservazione dei messaggi, consulta Quote correlate ai messaggi nella Guida per sviluppatori Amazon Simple Queue Service. Puoi modificare il periodo di conservazione dei messaggi utilizzando l'SQS`SetQueueAttributes`APIazione Amazon. 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 politica di redrive è un JSON oggetto che fa riferimento alla coda ARN delle lettere morte. L'attributo specifica il. `deadLetterTargetArn` ARN ARNDeve indicare una SQS coda Amazon nella stessa Account AWS regione del tuo SNS abbonamento Amazon. Per ulteriori informazioni, consulta Configurazione di una coda Amazon SNS dead-letter per un abbonamento.

Il seguente JSON oggetto è un esempio di politica di redrive, allegata a un SNS abbonamento.

```
{
  "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:

- Evita di scrivere la logica di SQS consumo di Amazon: imposta la coda di lettere non scritte come origine di eventi per la funzione Lambda per svuotare la coda di lettere non scritte.
- Scrivi la logica di SQS consumo di Amazon: usa Amazon SQS API o per AWS CLI scrivere una logica di consumo personalizzata per il polling, l'elaborazione e l'eliminazione dei messaggi nella coda delle lettere morte. AWS SDK

Come posso monitorare e registrare code DLQ?

Puoi utilizzare i CloudWatch parametri di Amazon per monitorare le code di posta indesiderata associate ai tuoi abbonamenti Amazon. SNS Tutte le SQS code di Amazon emettono CloudWatch parametri a intervalli di un minuto. Per ulteriori informazioni, consulta CloudWatch Parametri disponibili per Amazon SQS nella Amazon Simple Queue Service Developer Guide. Tutti gli SNS abbonamenti Amazon con code di lettera morta emettono anche parametri. CloudWatch Per ulteriori informazioni, consulta Monitoraggio SNS degli argomenti di Amazon tramite CloudWatch.

Per ricevere notifiche sull'attività nelle code di posta indesiderata, puoi utilizzare metriche e allarmi. CloudWatch L'impostazione di un allarme per la `NumberOfMessagesSent` metrica non è adatta perché questa metrica non acquisisce i messaggi inviati a a DLQ come risultato di tentativi di elaborazione falliti. Utilizza invece la `ApproximateNumberOfMessagesVisible` metrica, che acquisisce tutti i messaggi attualmente disponibili inDLQ, inclusi quelli spostati a causa di errori di elaborazione.

Esempio di configurazione degli allarmi CloudWatch

1. Crea un CloudWatchallarme per la **`ApproximateNumberOfMessagesVisible`** metrica.
2. Imposta la soglia di allarme su 1 (o su un altro valore appropriato in base alle aspettative e al DLQ traffico).
3. Specificate un SNS argomento Amazon per ricevere una notifica quando scatta l'allarme. Questo SNS argomento di Amazon può inviare una notifica di allarme a qualsiasi tipo di endpoint (ad esempio un indirizzo e-mail, un numero di telefono o un'app per dispositivi mobili).

Puoi utilizzare CloudWatch Logs per esaminare le eccezioni che causano il fallimento delle SNS consegne Amazon e per l'invio dei messaggi a code di lettera morta. Amazon SNS può registrare sia le consegne riuscite che quelle non riuscite. CloudWatch Per ulteriori informazioni, consulta Attributi SNS delle app mobili Amazon.

Configurazione di una coda Amazon SNS dead-letter per un abbonamento

Una coda di lettere morte è una coda Amazon a SQS cui un SNS abbonamento Amazon può indirizzare 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 di lettere non SNS ricevute su Amazon e Tentativi di recapito dei SNS messaggi Amazon.

Questa pagina mostra come utilizzare AWS Management Console, an, the AWS SDK AWS CLI, e AWS CloudFormation per configurare una coda di lettere non scritte per un abbonamento Amazon. SNS

> **Note**
>
> Per un FIFOargomento, puoi utilizzare una SQS coda Amazon come coda di lettere non scritte per l'abbonamento Amazon. SNS FIFOle sottoscrizioni agli argomenti utilizzano le FIFO code e le sottoscrizioni agli argomenti standard utilizzano le code standard.

Prerequisiti

Prima di configurare una coda DLQ, completare i seguenti prerequisiti:

1. Crea un SNS argomento Amazon denominato`MyTopic`.
2. Crea una SQS coda Amazon denominata`MyEndpoint`, da utilizzare come endpoint per l'abbonamento AmazonSNS.
3. (Salta per AWS CloudFormation) Iscriviti alla coda per accedere all'argomento.
4. Crea un'altra SQS coda Amazon denominata`MyDeadLetterQueue`, da utilizzare come coda di lettere non scritte per l'abbonamento Amazon. SNS
5. Per consentire ad Amazon l'accesso SNS principale all'SQSAPIazione Amazon, imposta la seguente politica di coda 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 utilizzando SNS AWS Management Console
- Per configurare una coda di lettere non scritte per un abbonamento Amazon SNS utilizzando un AWS SDK
- Per configurare una coda di lettere non scritte per un abbonamento Amazon utilizzando SNS AWS CLI
- Per configurare una coda di lettere non scritte per un abbonamento Amazon utilizzando SNS AWS CloudFormation

Per configurare una coda di lettere non scritte per un abbonamento Amazon utilizzando SNS AWS Management Console

Prima di iniziare questo tutorial, completare i prerequisiti descritti di seguito.

1. Accedi alla SQSconsole Amazon.
2. Crea una SQS coda Amazon o usa una coda esistente e annota la ARN coda nella scheda Dettagli della coda, ad esempio:

   ```
   arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
   ```

3. Accedi alla SNSconsole Amazon.
4. Nel riquadro di navigazione, scegli Sottoscrizioni.
5. Sulla pagina Subscriptions (Abbonamenti), selezionare una sottoscrizione esistente, quindi scegliere Edit (Modifica).

6. Nella sezione Modifica ***1234a567-bc89-012d-3e45-6fg7h890123i***espandi la sezione Redrive policy (dead-letter queue), quindi procedi come segue:
   a. Scegli Enabled (Abilitato).
   b. Specificare il nome ARN di una SQS coda Amazon.
7. Scegli 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 AWS SDK

Prima di eseguire questo esempio, completare i prerequisiti.

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

Il seguente esempio di codice mostra come utilizzare. `SetSubscriptionAttributesRedrivePolicy`

Java

SDKper 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.

```
// 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 utilizzando SNS 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 Guida per l'utente AWS Command Line Interface .
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 utilizzando SNS AWS CloudFormation

Prima di iniziare questo tutorial, completare i prerequisiti descritti di seguito.

1. Copia il JSON codice seguente 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, AWS CloudFormation visualizza lo stato _. CREATE COMPLETE

Archiviazione, riproduzione e analisi dei SNS messaggi Amazon

Gli argomenti SNS standard di Amazon 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.

SNSFIFOGli argomenti di Amazon supportano un archivio di messaggi locale e senza codice che consente ai proprietari degli argomenti di archiviare (o 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 SNS messaggi su Amazon per argomenti FIFO.

Funzionalità	Argomenti standard	FIFOArgomenti
Archiviazione di messaggi	Flussi di distribuzione da Fanout a Firehose	Archiviazione dei SNS messaggi Amazon per i proprietari di FIFO argomenti
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 SNS messaggi Amazon per gli abbonati all'FIFOargomento

Gestione e ottimizzazione delle risorse in Amazon SNS

Questo argomento fornisce indicazioni su come sfruttare tutto il potenziale di Amazon SNS garantendo prestazioni ottimali, riducendo i costi non necessari e mantenendo risorse ben organizzate.

Argomenti

Etichettatura degli SNS argomenti di Amazon

Amazon SNS supporta l'etichettatura di SNS argomenti Amazon. Questo può aiutarti a tenere traccia e gestire i costi associati ai tuoi argomenti, a fornire una maggiore sicurezza nelle policy di AWS Identity and Access Management (IAM) e a consentirti di cercare o filtrare facilmente migliaia di argomenti. Il tagging ti consente di gestire i tuoi SNS argomenti Amazon 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

Per organizzare e identificare gli SNS argomenti di Amazon per l'allocazione dei costi, puoi aggiungere tag che identificano lo scopo di un argomento. Questo è particolarmente utile in presenza di molti argomenti. Puoi utilizzare i tag di allocazione dei costi per organizzare la AWS fattura in modo che rifletta la tua struttura dei costi. A tale scopo, registrati per ricevere nella fattura AWS dell'account le chiavi e i valori dei 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 dei tuoi SNS argomenti Amazon, come segue:

Risorsa	Chiave	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, fornisci informazioni sui tag delle risorse nell'elemento condizione di una IAM politica per la gestione dell'accesso basato sui tag. Per informazioni su come etichettare le tue risorse utilizzando la SNSconsole Amazon o il AWS SDK, consulta Configurazione dei tag.

Puoi limitare l'accesso a un'IAMidentità. Ad esempio, puoi limitare `Publish` e `PublishBatch` accedere a tutti gli SNS argomenti di Amazon che includono un tag con la chiave `environment` e il valore`production`, consentendo al contempo l'accesso a tutti gli altri SNS argomenti Amazon. 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 Controllare l'accesso tramite tag nella Guida IAM per l'utente.

> **Note**
>
> L'impostazione dell'IAMautorizzazione per `Publish` imposta l'autorizzazione per entrambi `Publish` e`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 AWS account può contenere decine di migliaia di SNS argomenti Amazon (consulta Amazon SNS Quotas 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());
    }
}
```

Configurazione dei tag SNS tematici di Amazon

Questa pagina mostra come utilizzare AWS Management Console AWS SDK, an e the per configurare i AWS CLI tag per un SNSargomento di Amazon.

> **Important**
>
> Non aggiungere informazioni di identificazione personale (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

- Pubblicazione, aggiunta e rimozione di tag per un SNS argomento di Amazon utilizzando il AWS Management Console

- Aggiungere tag a un argomento utilizzando un AWS SDK
- Gestione dei tag con Amazon SNS API actions
- APIazioni che supportano ABAC

Pubblicazione, aggiunta e rimozione di tag per un SNS argomento di Amazon utilizzando il AWS Management Console

1. Accedi alla SNSconsole Amazon.
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. Scegli Save changes (Salva modifiche).

Aggiungere tag a un argomento utilizzando un AWS SDK

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

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

CLI

AWS CLI

Aggiungere un tag a un argomento

L'`tag-resource`esempio seguente aggiunge un tag di metadati all'SNSargomento Amazon 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 API i dettagli, consulta TagResource AWS CLICommand Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi TagResource AWS SDK for Java 2.xAPIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento TagResource AWSSDKa Kotlin API.

Gestione dei tag con Amazon SNS API actions

Per gestire i tag utilizzando Amazon SNSAPI, utilizza le seguenti API azioni:

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

APIazioni che supportano ABAC

Di seguito è riportato un elenco di API azioni che supportano il controllo degli accessi basato sugli attributi (). ABAC Per maggiori dettagliABAC, vedi A cosa serve? ABAC AWS nella Guida IAM per l'utente.

- `AddPermission`
- `ConfirmSubscription`
- `DeleteTopic`
- `GetDataProtectionPolicy`
- `GetSubscriptionAttributes`
- `GetTopicAttributes`
- `ListSubscriptionsByTopic`
- `ListTagsForResource`
- `Publish`
- `PublishBatch`
- `PutDataProtectionPolicy`
- `RemovePermission`

- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

Fonti e destinazioni di SNS eventi Amazon

Amazon SNS connette Servizi AWS sistemi esterni instradando le notifiche basate sugli eventi. Amazon SNS riceve eventi da diverse fonti Servizi AWS, come aggiornamenti della pipeline di dati, azioni di EC2 scalabilità di Amazon o avvisi di sicurezza, e pubblica questi eventi su argomenti di Amazon. SNS Questi argomenti inviano quindi notifiche a destinazioni designate.

Amazon SNS supporta due tipi principali di destinazioni: Application-to-Application (A2A) e Application-to-Person (A2P). Nella messaggistica A2A, Amazon SNS può inviare eventi a Lambda per attivare una logica aziendale personalizzata, ad Amazon SQS per accodare i messaggi e ad Amazon Kinesis Data Firehose per lo streaming di dati verso servizi di storage e analisi. Per la messaggistica A2PSMS, Amazon SNS può inviare notifiche via e-mail e notifiche push ai dispositivi mobili, assicurando che gli utenti o i team ricevano avvisi tempestivi.

Agendo come hub centrale, Amazon SNS indirizza le notifiche nei posti giusti, aiutandoti ad automatizzare e gestire la tua AWS infrastruttura in modo più efficace. Questa configurazione consente una perfetta integrazione tra i servizi e una comunicazione affidabile con utenti e sistemi.

Argomenti

Fonti di SNS eventi Amazon

Amazon SNS si integra con un'ampia gamma di diverse Servizi AWS categorie, consentendo a questi servizi di pubblicare eventi su SNS argomenti Amazon. Questa integrazione fornisce notifiche in tempo reale di eventi chiave, come cambiamenti nell'infrastruttura, prestazioni delle applicazioni e gestione dei costi.

> **Note**
>
> Amazon SNS ha introdotto alcuni FIFOargomenti a ottobre 2020. Attualmente, la maggior parte AWS dei servizi supporta l'invio di eventi solo ad argomenti standard.

Argomenti

Servizi di analisi

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS analisi come Athena e Amazon Redshift per fornire notifiche in tempo reale per eventi chiave AWS Data Pipeline, tra cui violazioni dei limiti di controllo, aggiornamenti dello stato della pipeline e attività di data warehouse.

Puoi sfruttare queste integrazioni per automatizzare le risposte e mantenere una supervisione efficace delle tue operazioni sui dati.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon Athena: consente di analizzare i dati in Amazon S3 utilizzando lo standard. SQL	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 SnsAlarmla Guida per gli AWS Data Pipeline sviluppatori.
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

La tabella seguente descrive come Amazon SNS si integra con i servizi di integrazione delle applicazioni come EventBridge e AWS Step Functions, abilitando il routing dei dati e le notifiche in tempo reale per le applicazioni aziendali critiche.

Puoi sfruttare queste integrazioni per ricevere avvisi dagli EventBridge eventi e orchestrare i flussi di lavoro utilizzando Step Functions, migliorando l'automazione e la reattività delle tue applicazioni.

Servizio AWS	Vantaggi 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.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS Step Functions— Consente di combinare AWS Lambda funzioni e altri AWS servizi per creare applicazioni aziendali critiche.	Ricevere la notifica degli eventi Step Functions. Per ulteriori informazioni, consulta Call Amazon SNS with Step Functions nella AWS Step Functions Developer Guide.

Servizi di gestione di costi e fatturazione

La tabella seguente descrive come AWS Billing and Cost Management si integra con Amazon per SNS fornire notifiche per budget, variazioni di prezzo e anomalie dei costi.

Puoi sfruttare questa integrazione per configurare SNS gli argomenti di Amazon in modo da ricevere avvisi in tempo reale sulle tue AWS spese, aiutandoti a monitorare i costi e rispondere in modo efficiente agli addebiti imprevisti.

Servizio AWS	Vantaggi 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 SNS argomento Amazon per le notifiche sul budget • Configurazione delle notifiche • Rilevamento di spese insolite con AWS Cost Anomaly Detection

Servizi applicativi aziendali

La tabella seguente descrive come Amazon Chime si integra con Amazon SNS per inviare notifiche per eventi di riunioni importanti, consentendoti di rimanere informato sulle comunicazioni e sulla pianificazione.

Puoi sfruttare questa integrazione per utilizzare le notifiche degli eventi di Amazon SDK Chime per migliorare i tuoi strumenti di collaborazione all'interno e all'esterno dell'organizzazione.

Servizio AWS	Vantaggi 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 le notifiche degli SDK eventi di Amazon Chime nella Amazon Chime Developer Guide.

Servizi di elaborazione

La tabella seguente descrive come Amazon SNS si integra con vari servizi di AWS elaborazione, consentendoti di ricevere notifiche per eventi chiave come azioni di Auto Scaling, completamenti di Image EC2 Builder, modifiche all'ambiente Elastic Beanstalk, output delle funzioni Lambda e soglie metriche Lightsail.

Puoi sfruttare queste integrazioni per gestire in modo efficiente le tue applicazioni e risorse rimanendo informato sugli aggiornamenti e sulle azioni critiche in corso. Servizi AWS

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon EC2 Auto Scaling: ti aiuta a disporre del numero corretto di istanze Amazon Elastic Compute Cloud (AmazonEC2) disponibili per la gestione del carico dell'applicazione.	Ricevi notifiche quando Auto Scaling avvia o chiude EC2 istanze Amazon nel tuo gruppo Auto Scaling. Per ulteriori informazioni, consulta Ricevere SNS notifiche Amazon quando il tuo gruppo Auto Scaling aumenta nella Amazon EC2Auto Scaling User Guide.
EC2Image Builder: consente di automatizzare la creazione, la gestione e l'implementazione di immagini up-to-date server personalizzate, sicure e preinstallate con software e impostazioni per soddisfare specifici standard IT.	Ricevi notifiche quando le build sono completate. Per ulteriori informazioni, consulta Tracciamento delle immagini più recenti del server nelle pipeline di EC2 Image Builder sul blog di Compute AWS .
AWS Elastic Beanstalk – Gestisce automaticamente i dettagli di provisioning della capacità,	Ricevi notifiche di eventi importanti che interessano la tua applicazione. Per ulteriori

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
bilanciamento del carico, dimensionamento e monitoraggio dello stato dell'applicazione.	informazioni, consulta Notifiche dell'ambiente Elastic Beanstalk SNS con Amazon AWS Elastic Beanstalk nella Developer Guide.
AWS Lambda - Consente di eseguire il codice senza effettuare il provisioning dei server o senza gestirli.	Ricevi i dati di output della funzione impostando un SNS argomento come coda di lettere morte Lambda o destinazione Lambda. Per ulteriori informazioni, consulta Chiamata asincrona nella Guida per gli sviluppatori AWS Lambda .
Amazon Lightsail: aiuta gli sviluppatori a iniziare AWS a 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

La tabella seguente descrive come Amazon SNS si integra con servizi AWS container come Amazon EKS Distro e AmazonECS, consentendoti di tenere traccia degli aggiornamenti e delle patch di sicurezza per EKS i cluster Amazon e ricevere notifiche per nuove versioni ottimizzate. ECS AMI

Puoi sfruttare queste integrazioni per mantenere la sicurezza e l'efficienza delle implementazioni dei container rimanendo informato su aggiornamenti e modifiche importanti.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon EKS Distro: 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 Introduzione ad Amazon EKS Distro, una distribuzione Kubernetes open source utilizzata da Amazon. EKS

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon Elastic Container Service (AmazonECS): consente di eseguire, arrestare e gestire contenitori su un cluster.	Ricevi notifiche quando è disponibile una nuova versione ECS ottimizzata per AMI Amazon. Per ulteriori informazioni, consulta la sezione Abbonamento alle notifiche di AMI aggiornamento ECS ottimizzate per Amazon nella Amazon Elastic Container Service Developer Guide.

Servizi di coinvolgimento dei clienti

La tabella seguente descrive come Amazon SNS migliora i servizi di coinvolgimento dei clienti integrandosi con Amazon Connect e Amazon Simple Email Service (SES) AWS End User Messaging SMS, consentendoti di ricevere avvisi e convalide, configurare la SMS messaggistica bidirezionale e monitorare le notifiche e-mail per rimbalzi, reclami e consegne.

Queste integrazioni ti aiutano a gestire le comunicazioni con i clienti su più canali.

Servizio AWS	Vantaggi 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 Amazon Connect nella Amazon Connect Administrator Guide.
AWS End User Messaging SMS— Ti aiuta a coinvolgere i tuoi clienti inviando loro e-mail, messaggi vocali SMS e notifiche push.	Configurazione bidirezionaleSMS, che consente di ricevere messaggi dai clienti. Per ulteriori informazioni, consulta SMSMessaggistica bidirezionale nella Guida per l'AWS End User Messaging SMS utente.
Amazon Simple Email Service (AmazonSES): offre un modo conveniente per inviare e ricevere e-mail utilizzando i propri indirizzi e-mail e domini.	Ricevi notifiche di mancati recapiti (bounce), reclami e consegne. Per ulteriori informazioni, consulta Configurazione delle SNS notifiche Amazon per Amazon SES nella Amazon Simple Email Service Developer Guide.

Servizi di database

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS database come AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon ElastiCache, Amazon Neptune, Amazon Redshift e Amazon Relational Database RDS Service () per inviare notifiche su eventi importanti come migrazioni di dati, attività di manutenzione, aggiornamenti della cache e modifiche al database.

Queste integrazioni ti aiutano a monitorare e gestire gli ambienti di database in modo più efficace fornendo avvisi tempestivi sugli eventi operativi chiave.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS Database Migration Service— Migra i dati dai database locali al. Cloud AWS	Ricevi notifiche quando si verificano AWS DMS eventi, ad esempio quando viene creata o eliminata un'istanza di replica. Per ulteriori informazioni, vedi Utilizzo degli eventi e delle notifiche AWS Database Migration Service nella AWS Database Migration Service Guida per l'utente.
Amazon DynamoDB: offre prestazioni veloci e prevedibili con una scalabilità perfetta in questo servizio senza database completamente gestito. SQL	Ricevi notifiche quando si verificano eventi di manutenzione. Per ulteriori informazioni, consulta Personalizzazione delle impostazioni del DAX cluster nella Amazon DynamoDB Developer Guide.
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 distribuito.	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 (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.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
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, il funzionamento e la scalabilità di un database relazionale nel AWS cloud.	Ricevi notifiche RDS sugli eventi di Amazon. Per ulteriori informazioni, consulta Using Amazon RDS Event Notification nella Amazon RDS User Guide.

Servizi e strumenti per sviluppatori

La tabella seguente descrive come Amazon SNS si integra con i servizi di strumenti per AWS sviluppatori AWS CodeBuild, come Amazon e AWS CodeCommit AWS CodeDeploy CodeGuru AWS CodePipeline AWS CodeStar, per fornire notifiche per eventi critici come modifiche allo stato delle build, aggiornamenti del repository, avanzamento della distribuzione, anomalie delle prestazioni e azioni della pipeline.

Queste integrazioni ti aiutano a monitorare e gestire in modo efficiente i flussi di lavoro di sviluppo software ricevendo avvisi tempestivi su eventi importanti.

Servizio AWS	Vantaggi 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 CodeBuild utente.
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 AWS CodeCommit trigger per un SNS argomento Amazon nella Guida per l'AWS CodeCommit utente.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS CodeDeploy— Automatizza le distribuzioni di applicazioni su istanze Amazon, EC2 istanze locali, funzioni Lambda serverless o servizi Amazon. ECS	Ricevi notifiche per distribuzioni o eventi di istanze. CodeDeploy Per ulteriori informazioni, consulta Creare un trigger per un CodeDeploy evento nella Guida per l'AWS CodeDeploy utente.
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 CodePipeline utente.
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 SNSgli argomenti Configure Amazon per le notifiche nella Developer Tools Console User Guide.

Servizi web e mobili front-end

La tabella seguente descrive come Amazon SNS si integra AWS End User Messaging SMS per migliorare il coinvolgimento dei clienti inviando e-mailSMS, messaggi vocali e notifiche push, inclusa la possibilità di configurare la ricezione bidirezionale SMS dei messaggi dei clienti.

Questa integrazione ti consente di interagire in modo più efficace con i tuoi clienti attraverso vari canali di comunicazione.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS End User Messaging SMS— Ti aiuta a coinvolgere i tuoi clienti inviando loro e-mail, messaggi vocali SMS e notifiche push.	Configurazione bidirezionaleSMS, che consente di ricevere messaggi dai clienti. Per ulteriori informazioni, consulta SMSMessaggistica bidirezionale nella Guida per l'AWS End User Messaging SMS utente.

Servizi per lo sviluppo dei giochi

La tabella seguente descrive come Amazon SNS si integra con Amazon GameLift per fornire notifiche per matchmaking ed eventi di coda nei server di gioco multiplayer basati su sessioni.

Questa integrazione aiuta gli sviluppatori di giochi ad automatizzare e monitorare la distribuzione, il funzionamento e la scalabilità dei loro server di gioco, garantendo un'esperienza di gioco senza interruzioni.

Servizio AWS	Vantaggi 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 posizionamento delle sessioni di gioco nella Amazon GameLift Developer Guide.

Servizi Internet of Things

La tabella seguente descrive in che modo Amazon SNS si integra con AWS IoT servizi come, AWS IoT Core, e AWS IoT Device Defender AWS IoT Events AWS IoT Greengrass, per fornire notifiche per eventi e avvisi IoT.

Queste integrazioni consentono di monitorare efficacemente il comportamento dei dispositivi, ricevere avvisi per attività anomale e gestire i dispositivi IoT con aggiornamenti e azioni in tempo reale.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS IoT Core— Fornisce i servizi cloud che collegano i dispositivi IoT ad altri dispositivi e Cloud AWS servizi.	Ricevi notifiche di AWS IoT Core eventi. Per ulteriori informazioni, consulta Creating an Amazon SNS rule nella AWS IoT Developer Guide.
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 informazioni, consulta How to use AWS IoT Device Defender detect nella AWS IoT Developer Guide.
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.	Ricevi notifiche di AWS IoT Events eventi. Per ulteriori informazioni, consultare Amazon Simple Notification Service nella AWS IoT Events Guida per gli sviluppatori
AWS IoT Greengrass— Si estende AWS ai dispositivi fisici in modo che possano agire localmente sui dati generati, continuando a utilizzare il cloud per la gestione, l'analisi e l'archiviazione durevole.	Ricevi notifiche di AWS IoT Greengrass eventi. Per ulteriori informazioni, consulta il SNSconnet tore nella Guida per AWS IoT Greengrass Version 1 gli sviluppatori.

Servizi Machine Learning

La tabella seguente descrive come Amazon SNS si integra con i servizi di apprendimento AWS automatico, come Amazon, Amazon DevOps Guru CodeGuru, Amazon Lookout for Metrics, Amazon

Rekognition e Amazon, per fornire notifiche per anomalie, approfondimenti operativi SageMaker e attività di etichettatura dei dati.

Queste integrazioni consentono di monitorare le prestazioni delle applicazioni, ricevere avvisi in caso di irregolarità dei dati e semplificare l'implementazione di modelli di apprendimento automatico con aggiornamenti in tempo reale.

Servizio AWS	Vantaggi 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 automatic o per aiutarti a migliorare le prestazioni delle tue applicazioni operative.	Proseguire approfondimenti e conferme. Per ulteriori informazioni, consulta Offri approfond imenti operativi basati su ML ai tuoi team di chiamata utilizzando PagerDuty Amazon DevOps Guru sul blog AWS Management & Governance.
Amazon Lookout for Metrics - Individua le anomalie nei dati, ne determina le cause principali e consente di intervenire rapidamen te.	Ricevi notifiche di anomalie. Per ulteriori informazioni, consulta Using Amazon SNS with Lookout for Metrics nella Amazon Lookout for Metrics Developer Guide.
Amazon Rekognition - Consente di aggiunger e 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 Rekogniti on.
Amazon SageMaker: consente ai data scientist e agli sviluppatori di creare e addestrare modelli di machine learning e quindi di distribui rli 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

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS gestione e governance come AWS Chatbot, AWS CloudFormation, CloudTrail, CloudWatch, AWS Config,, AWS Control Tower AWS License Manager AWS Service Catalog AWS Systems Manager, e fornisce notifiche per eventi chiave come modifiche all'infrastruttura, avvisi di conformità e approfondimenti operativi.

Queste integrazioni ti aiutano a monitorare e gestire i tuoi AWS ambienti in modo efficiente fornendo avvisi e aggiornamenti tempestivi ai team e ai sistemi pertinenti.

Servizio AWS	Vantaggi 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 in. 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 e fornire implementazioni di AWS infrastrutture in modo prevedibile e ripetuto.	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 la sezione Configurazione SNS delle notifiche di Amazon CloudTrail nella Guida per l'AWS CloudTrail 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.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS Config— Fornisce una visualizzazione dettagliata della configurazione delle AWS risorse del tuo 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 AWS Config inviate a un SNS argomento e Esempi di notifiche di modifica degli elementi di configurazione nella Guida per gli AWS Config sviluppatori.
AWS Control Tower— Consente di configurare e gestire un ambiente sicuro, conforme e con più account. AWS	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 di software in modo centralizzato negli ambienti locali. AWS	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 visualizzare e controllare l'infrastruttura su AWS.	Ricevi notifiche sullo stato dei comandi. Per ulteriori informazioni, consulta la sezione Monitoring Systems Manager delle modifiche allo stato utilizzando SNS le notifiche di Amazon nella Guida per l'AWS Systems Manager utente.

Servizi multimediali

La tabella seguente descrive come Amazon SNS si integra con Amazon Elastic Transcoder per inviare notifiche quando i lavori di transcodifica multimediale cambiano lo stato, consentendoti di monitorare e gestire in modo efficiente la conversione dei file multimediali archiviati in Amazon S3 in formati adatti ai dispositivi di riproduzione consumer.

Questa integrazione ti aiuta a semplificare i flussi di lavoro di elaborazione multimediale fornendo avvisi in tempo reale sullo stato del lavoro.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon Elastic Transcoder - Consente di convertire i file multimediali memorizzati in Amazon S3 in file multimediali nei formati richiesti dai dispositivi di riproduzione dei consumatori.	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.

Servizi di trasferimento e migrazione

La tabella seguente descrive come Amazon SNS si integra con i servizi di AWS migrazione e trasferimento, come, AWS Database Migration Service (DMS) e AWS Application Discovery Service AWS Snowball, per fornire notifiche per eventi come la raccolta di dati sui server, le attività di migrazione dei database e i lavori di trasferimento dei dati.

Queste integrazioni ti aiutano a gestire e monitorare in modo efficace i processi di migrazione al cloud offrendo avvisi e aggiornamenti in tempo reale sulle attività di migrazione critiche.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS Application Discovery Service— Ti aiuta a pianificare la migrazione verso il Cloud AWS raccogliendo dati di utilizzo e configurazione sui tuoi server locali.	Ricevi notifiche di eventi tramite AWS CloudTrail. Per ulteriori informazioni, vedere Logging Application Discovery Service API con AWS CloudTrail nella Application Discovery Service User Guide.
AWS Database Migration Service— Migra i dati dai database locali a. Cloud AWS	Ricevi notifiche quando si verificano AWS DMS eventi, ad esempio quando viene creata

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
	o eliminata un'istanza di replica. Per ulteriori informazioni, vedi Utilizzo degli eventi e delle notifiche AWS Database Migration Service nella AWS Database Migration Service Guida 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 Notifiche per i dispositivi Snow Family nella Guida per l'AWS Snowcone utente.

Reti e servizi di distribuzione di contenuti

La tabella seguente descrive come Amazon SNS si integra con i servizi AWS di rete e distribuzione di contenuti, come Amazon API Gateway, Amazon CloudFront, Elastic Load Balancing AWS Direct Connect, Amazon Route 53 e VPC Amazon, per inviare notifiche per eventi API come messaggi, allarmi metrici, modifiche dello stato di connessione CloudFront , eventi di bilanciamento del carico, stati di controllo dello stato di salute e attività degli endpoint. VPC

Queste integrazioni ti aiutano a monitorare e gestire la rete e le operazioni di distribuzione dei contenuti fornendo avvisi e aggiornamenti tempestivi.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon API Gateway: consente di creare e implementare soluzioni personalizzate REST e WebSocket APIs su qualsiasi scala.	Ricevi messaggi pubblicati su un endpoint API Gateway. Per ulteriori informazioni, consulta Tutorial: Build an API Gateway REST API with AWS integration nella APIGateway Developer Guide.
Amazon CloudFront: velocizza la distribuzione di contenuti Web statici e dinamici, come .html, .css, .php, immagini e file multimediali.	Ricevi notifiche quando si verificano allarmi basati su metriche specificate. CloudFront Per ulteriori informazioni, consulta la sezione

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
	Impostazione degli allarmi per ricevere notifiche nell'Amazon CloudFront Developer Guide.
AWS Direct Connect— Collega la rete interna a una AWS Direct Connect posizione 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, vedere Creazione di CloudWatch allarmi per monitorare le AWS Direct Connect connessioni nella Guida per l'AWS Direct Connect utente.
Elastic Load Balancing: distribuisce automatic amente il traffico in entrata su più destinazi oni, come EC2 istanze Amazon, 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 CloudWatc h allarmi per il sistema di bilanciamento del carico nella Guida per l'utente di Classic Load Balancers.
Amazon Route 53: fornisce la registrazione del dominio, il DNS routing e il controllo dello stato.	Ricevere una notifica quando un controllo dello stato non è integro. Per ulteriori informazioni, consulta Ricevere una SNS notifica Amazon quando lo stato di un controllo sanitario non è integro (console) nella Amazon Route 53 Developer Guide.
Amazon Virtual Private Cloud (AmazonVPC): ti consente di avviare AWS risorse in una rete virtuale che hai definito.	Ricevere notifiche per eventi specifici che si verificano nell'endpoint di interfaccia. Per ulteriori informazioni, consulta Creare e gestire una notifica per un servizio endpoint nella Amazon VPC User Guide.

Servizi per sicurezza, identità e conformità

La tabella seguente descrive come Amazon SNS si integra con i servizi di AWS sicurezza, identità e conformità, come AWS Directory Service Amazon, Amazon Inspector e GuardDuty, per fornire

notifiche per le modifiche allo stato delle directory, i risultati di sicurezza, gli eventi di Inspector AWS Security Hub e gli annunci degli hub di sicurezza.

Queste integrazioni ti aiutano a mantenere solide pratiche di sicurezza offrendo avvisi e aggiornamenti tempestivi sugli eventi di sicurezza e conformità.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS Directory Service— Fornisce diversi modi per utilizzare Microsoft Active Directory (AD) con altri Servizi AWS.	Ricevi e-mail o messaggi di testo (SMS) quando lo stato della tua directory cambia. Per ulteriori informazioni, consulta Configurare le notifiche sullo stato della directory nella AWS Directory Service Guida 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'SNSargome nto Abbonamento agli GuardDuty annunci nella Amazon GuardDuty User Guide.
Amazon Inspector: verifica l'accessibilità di rete EC2 delle istanze Amazon e lo stato di sicurezza delle applicazioni eseguite su tali istanze.	Ricevi notifiche per gli eventi Amazon Inspector . Per ulteriori informazioni, consulta Configura zione di un SNS argomento per le notifiche di Amazon Inspector nella Amazon Inspector User Guide.
AWS Security Hub: automatizza i controlli di sicurezza di AWS l e centralizza gli avvisi di sicurezza.	Ricevi notifiche sugli AWS Security Hub annunci, incluse notifiche su AWS Security Hub controlli o standard che sono stati aggiunti, modificati o ritirati. Per ulteriori informazioni, consulta Abbonarsi agli AWS Security Hub annunci con Amazon. SNS

Servizi serverless

La tabella seguente descrive come Amazon SNS si integra con servizi come Amazon DynamoDB EventBridge, Amazon e Lambda per inviare notifiche per eventi chiave come aggiornamenti di manutenzione, flussi di dati in tempo reale e output delle funzioni Lambda.

Queste integrazioni ti aiutano a monitorare e gestire in modo efficiente le tue applicazioni ricevendo avvisi tempestivi sulle operazioni critiche e automatizzando le risposte a questi eventi.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
Amazon DynamoDB: offre prestazioni veloci e prevedibili con una scalabilità perfetta in questo servizio senza database completamente gestito. SQL	Ricevi notifiche quando si verificano eventi di manutenzione. Per ulteriori informazioni, consulta Personalizzazione delle impostazioni del DAX cluster nella Amazon DynamoDB Developer Guide.
Amazon EventBridge: fornisce un flusso di dati in tempo reale dalle tue applicazioni software-as-a-service (SaaS) Servizi AWS 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.	Ricevi i dati di output della funzione impostando un SNS argomento come coda di lettere morte Lambda o destinazione Lambda. Per ulteriori informazioni, consulta Chiamata asincrona nella AWS Lambda Guida per gli sviluppatori.

Servizi di storage

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS storage come AWS Backup Amazon Elastic File System (EFS), Amazon S3 Glacier, Amazon S3 e fornisce notifiche per vari eventi come attività di backup AWS Snowball , allarmi del file system, processi di recupero dati, modifiche ai bucket e operazioni di trasferimento dei dati.

Queste integrazioni ti aiutano a monitorare e gestire in modo efficiente le tue soluzioni di storage ricevendo avvisi tempestivi sugli eventi di storage critici.

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
AWS Backup— Ti aiuta a centralizzare e automatizzare il backup dei dati Servizi AWS nel cloud e in locale	Ricevi notifiche di eventi. AWS Backup Per ulteriori informazioni, consulta SNSUsare Amazon per tenere traccia AWS Backup degli eventi nella AWS Backup Developer Guide.
Amazon Elastic File System: fornisce lo storage di file per le tue EC2 istanze Amazon.	Ricevi notifiche sugli allarmi che hai creato per EFS gli eventi 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.	Imposta una configurazione di notifica su un vault in modo che quando un lavoro viene completato, venga inviato un messaggio a un argomento. SNS 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 Procedura dettagliata: Configurazione di un bucket per le notifiche (SNSargomento o SQS coda) e Monitoraggio dei progressi con i parametri di replica e le notifiche degli eventi di 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 Notifiche per i

Servizio AWS	Vantaggi dell'utilizzo con Amazon SNS
dei dati in sede a velocità elevate. faster-than-internet	dispositivi Snow Family nella Guida per l'AWS Snowcone utente.

Origini eventi aggiuntivi

La tabella seguente descrive in che modo Amazon SNS può essere utilizzato per ricevere notifiche tempestive sugli aggiornamenti AWS giornalieri delle funzionalità e sulle modifiche agli intervalli di indirizzi AWS IP, assicurandoti di essere informato sulle ultime versioni del AWS servizio, sui tipi di istanze, sugli VPC endpoint e sulle modifiche degli indirizzi IP pubblici.

Queste integrazioni ti aiutano a rimanere al passo up-to-date con i cambiamenti AWS dell'infrastruttura e a gestire le tue risorse in modo efficace.

Origine	Vantaggi dell'utilizzo con Amazon SNS
AWS Aggiornamenti giornalieri delle funzionalità	Ricevi informazioni tempestive e dettagliate su versioni e aggiornamenti AWS tramite un SNS argomento di Amazon. Queste versioni includono VPC endpoint Amazon Regioni AWS Servizi AWS, Servizi AWS integrati con AWS Service Quotas, tipi di istanze Amazon, tipi di istanze EC2 Amazon, tipi di istanze SageMaker Amazon Nimble Studio, versioni del motore di database Amazon e versioni di RDS Amazon Apache MSK Kafka. Per ulteriori informazioni, consulta Abbonarsi agli aggiornamenti AWS giornalieri delle funzionalità tramite Amazon SNS nel blog di AWS notizie.
AWS Intervalli di indirizzi IP	Ricevi notifiche sulle modifiche agli intervalli AWS IP tramite un SNS argomento Amazon. Per ulteriori informazioni, consulta la sezione Notifiche degli Riferimenti generali di Amazon Web Servicesintervalli di indirizzi AWS IP in e Iscriviti alle modifiche degli indirizzi IP AWS

Origine	Vantaggi dell'utilizzo con Amazon SNS
	pubblici tramite Amazon SNS nel AWS News Blog.

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 e servizi di AWS elaborazione, storage, database e rete sul blog Compute AWS
- Arricchimento delle architetture basate sugli eventi con le pipeline Event Fork sul blog di Compute AWS AWS

Destinazioni per SNS eventi Amazon

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

> **Note**
>
> Amazon SNS ha introdotto alcuni FIFOargomenti a ottobre 2020. Attualmente, la maggior parte AWS dei servizi supporta la ricezione di eventi solo su argomenti SNS standard. Amazon SQS supporta la ricezione di eventi sia SNS standard che FIFO tematici.

Argomenti

- A2A destinazioni
- Destinazioni A2P

A2A destinazioni

La tabella seguente descrive come Amazon SNS può fornire eventi a varie destinazioni application-to-application (A2A) come Amazon Data Firehose, Lambda, SQS AWS Amazon, Event Fork Pipelines ed endpoint /S. HTTP

Queste integrazioni consentono di archiviare e analizzare i dati, attivare logiche di business personalizzate, facilitare l'integrazione delle applicazioni e indirizzare gli eventi a webhook esterni, migliorando l'efficienza e la flessibilità delle architetture basate sugli eventi.

Destinazione di evento	Vantaggi dell'utilizzo con 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 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 Fanout SNS delle notifiche Amazon alle funzioni Lambda per l'elaborazione automatizzata.
Amazon SQS	Distribuire gli eventi alle code per scopi di integrazione delle applicazioni. Per ulteriori informazioni, consulta Fanout SNS delle notifiche Amazon alle SQS code Amazon per l'elaborazione asincrona.
AWS Event Fork Pipelines	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 Fanout SNS degli eventi Amazon su AWS Event Fork Pipelines.

Destinazione di evento	Vantaggi dell'utilizzo con Amazon SNS
HTTP/S	Distribuire eventi a webhook esterni. Per ulteriori informazioni, consulta Fanout SNS delle notifiche Amazon agli endpoint HTTPS.

Destinazioni A2P

La tabella seguente descrive in che modo Amazon SNS invia notifiche application-to-person (A2P) a varie destinazioni, tra cui telefoni cellulari tramite SMS notifiche push native, caselle di posta elettronica, chat room di Amazon Chime, canali Slack e informazioni operative ai team in servizio tramite. PagerDuty

Queste integrazioni migliorano la comunicazione e l'efficienza operativa abilitando avvisi e aggiornamenti in tempo reale su più piattaforme e canali di comunicazione.

Destinazione di evento	Vantaggi dell'utilizzo con Amazon SNS
SMS	Consegnare eventi ai telefoni cellulari come messaggi di testo. Per ulteriori informazioni, consulta Messaggi di testo mobili con Amazon SNS.
E-mail	Consegna degli eventi nelle caselle di posta in arrivo come messaggi di posta. Per ulteriori informazioni, consulta Configurazione e gestione dell'abbonamento SNS e-mail Amazon.
Endpoint platform	Distribuisci eventi ai telefoni cellulari come notifiche push native. Per ulteriori informazioni, consulta Invio di notifiche push per dispositivi mobili con Amazon SNS.
AWS Chatbot	Consegna eventi alle chat room Amazon Chime o ai canali Slack. Per ulteriori informazioni,

Destinazione di evento	Vantaggi dell'utilizzo con Amazon SNS
	consulta le pagine seguenti nella AWS Chatbot Guida per l'amministratore: • Configurazione AWS Chatbot con Amazon Chime • Configurazione AWS Chatbot con Slack • Utilizzo 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 AWS Management & Governance.

> **Note**
>
> Puoi offrire sia AWS eventi nativi che eventi personalizzati alle app di chat:
>
> - AWS Eventi nativi: puoi utilizzarli AWS Chatbot per inviare AWS eventi nativi, tramite SNS argomenti Amazon, ad Amazon Chime e Slack. Il set di AWS eventi nativi supportato include eventi di AWS Billing and Cost Management AWS Health, AWS CloudFormation, CloudWatch, Amazon e altri. Per ulteriori informazioni, consulta Utilizzo AWS Chatbot con altri servizi nella Guida per l'AWS Chatbot amministratore.
> - Eventi personalizzati: puoi anche inviare i tuoi eventi personalizzati, tramite SNS argomenti Amazon, ad Amazon Chime, Slack e Microsoft Teams. A tale scopo, si pubblicano eventi personalizzati su un SNS argomento, che invia 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 si usano i webhook per pubblicare SNS messaggi Amazon su Amazon Chime, Slack o Microsoft Teams?

Utilizzo di Amazon SNS per la application-to-application messaggistica

Amazon SNS semplifica la messaggistica application-to-application (A2A) separando gli editori dagli abbonati, il che supporta microservizi, sistemi distribuiti e applicazioni serverless. I messaggi vengono inviati agli SNS argomenti di Amazon, dove possono essere filtrati e recapitati a sottoscrittori come Lambda, SQS Amazon o endpoint. HTTP Se la consegna non riesce, i messaggi vengono archiviati in una coda di lettere non scritte per ulteriori analisi o rielaborazioni.

Per informazioni sull'uso di Amazon SNS per la application-to-application messaggistica con gli abbonati, consulta quanto segue:

Argomenti

- Flussi di distribuzione da Fanout a Firehose
- Fanout SNS delle notifiche Amazon alle funzioni Lambda per l'elaborazione automatizzata
- Fanout SNS delle notifiche Amazon alle SQS code Amazon per l'elaborazione asincrona
- Fanout SNS delle notifiche Amazon agli endpoint HTTPS
- Fanout SNS degli eventi Amazon su AWS Event Fork Pipelines
- Utilizzo di Amazon EventBridge Scheduler con Amazon SNS

Flussi di distribuzione da Fanout a Firehose

Puoi abbonare i flussi di distribuzione di Amazon Data Firehose agli SNS argomenti di Amazon, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi aggiuntivi. I messaggi pubblicati su un SNS argomento Amazon 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 notifiche Amazon SNS ad Amazon Simple Storage Service (Amazon S3), OpenSearch Amazon Redshift, Amazon Service (Service) e a fornitori di OpenSearch servizi di terze parti come Datadog, New Relic, MongoDB e Splunk.

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 SNS argomento Amazon, crea un flusso di consegna con una destinazione dell'indice di OpenSearch servizio. Puoi quindi abbonare lo stream di distribuzione di Firehose all'argomento AmazonSNS.

Amazon supporta SNS anche la registrazione dello stato di consegna dei messaggi per le notifiche inviate agli endpoint Firehose. Per ulteriori informazioni, consulta Stato di consegna dei SNS messaggi Amazon.

Argomenti

- Prerequisiti per sottoscrivere i flussi di distribuzione di Firehose agli argomenti di Amazon SNS
- Sottoscrizione di uno stream di distribuzione di Firehose a un argomento Amazon SNS
- Gestione dei SNS messaggi Amazon su più destinazioni di flussi di consegna
- Archiviazione e analisi dei SNS messaggi di Amazon: un esempio di utilizzo per le piattaforme di biglietteria aerea

Prerequisiti per sottoscrivere i flussi di distribuzione di Firehose agli argomenti di Amazon SNS

Per abbonare uno stream di distribuzione di Amazon Data Firehose a un SNS argomento, Account AWS devi avere:

- Un SNS argomento standard. Per ulteriori informazioni, consulta Creare un SNS argomento Amazon.
- Un flusso di distribuzione Firehose. Per ulteriori informazioni, consulta Creating an Amazon Data Firehose Delivery Stream e Concedi all'applicazione l'accesso alle risorse Firehose nella Amazon Data Firehose Developer Guide.
- Un ruolo AWS Identity and Access Management (IAM) che si fida del responsabile del SNS servizio Amazon e dispone dell'autorizzazione a scrivere nel flusso di consegna. Immetterai Amazon Resource Name (ARN) di questo ruolo come `SubscriptionRoleARN` quando crei l'abbonamento. Amazon SNS assume questo ruolo, che consente SNS ad Amazon di inserire record nel flusso di distribuzione di 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, devi anche modificare la relazione di trust per includere il responsabile del SNS servizio Amazon. Per esempio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Per ulteriori informazioni sulla creazione di ruoli, consulta Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella Guida per l'IAMutente.

Dopo aver completato questi requisiti, puoi iscrivere lo stream di distribuzione all'SNSargomento.

Sottoscrizione di uno stream di distribuzione di Firehose a un argomento Amazon SNS

Per inviare SNS notifiche Amazon 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 sottoscrivere uno stream di distribuzione di Firehose a un argomento

1. Accedi alla SNSconsole Amazon.

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 Argomento ARN, scegli l'Amazon Resource Name (ARN) di un argomento standard.
   b. Per Protocollo, scegliete Firehose.
   c. Per Endpoint, scegli uno stream ARN di distribuzione Firehose in grado di ricevere notifiche da Amazon. SNS
   d. Per il ruolo Subscription ARN, specificate il ARN ruolo AWS Identity and Access Management (IAM) che avete creato per la scrittura nei flussi di distribuzione Firehose. Per ulteriori informazioni, consulta Prerequisiti per sottoscrivere i flussi di distribuzione di Firehose agli argomenti di Amazon SNS.
   e. (Facoltativo) Per rimuovere tutti SNS i metadati Amazon dai messaggi pubblicati, scegli Abilita il recapito dei messaggi non elaborati. Per ulteriori informazioni, consulta Consegna di messaggi non SNS elaborati su Amazon.
5. (Facoltativo) Per configurare un criterio di filtro, espandere la sezione policy di filtro della sottoscrizione. Per ulteriori informazioni, consulta Politiche di filtro degli SNS abbonamenti Amazon.
6. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta Code di lettere non SNS ricevute su Amazon.
7. Scegli Create Subscription (Crea sottoscrizione).

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

Gestione dei SNS messaggi Amazon su più destinazioni di flussi di consegna

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.

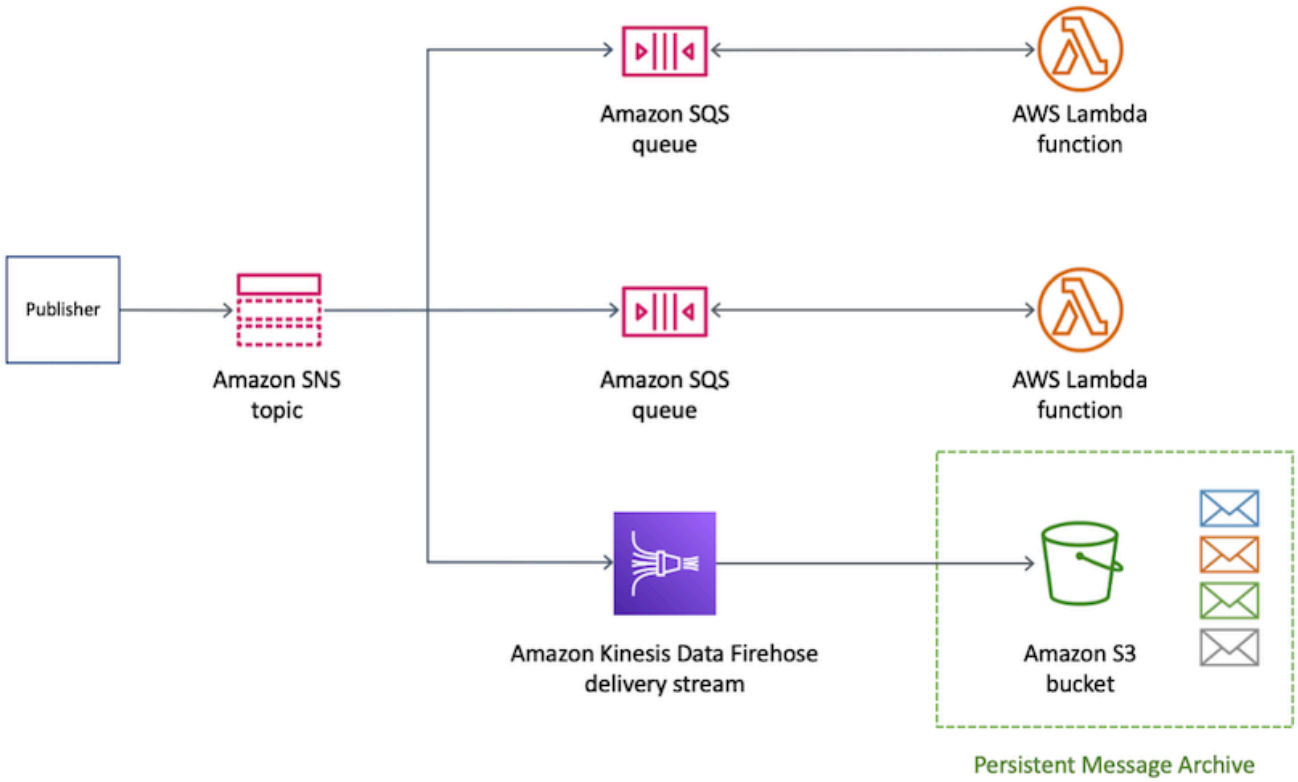
Argomenti

- Archiviazione e analisi dei SNS messaggi Amazon nelle destinazioni Amazon S3
- Integrazione dei SNS messaggi Amazon con le destinazioni di Amazon OpenSearch Service

- Configurazione del recapito e dell'analisi dei SNS messaggi Amazon nelle destinazioni Amazon Redshift
- Configurazione del recapito di SNS messaggi Amazon alle HTTP destinazioni utilizzando Amazon Data Firehose

Archiviazione e analisi dei SNS messaggi Amazon nelle destinazioni 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

- Formattazione delle SNS notifiche Amazon per lo storage nelle destinazioni Amazon S3
- Analisi dei SNS messaggi Amazon archiviati in Amazon S3 utilizzando Athena

Formattazione delle SNS notifiche Amazon per lo storage nelle destinazioni Amazon S3

L'esempio seguente mostra una SNS notifica Amazon inviata a un bucket Amazon Simple Storage Service (Amazon S3), utilizzando rientri per la leggibilità.

Note

In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito di messaggi non elaborati è disabilitato, Amazon SNS aggiunge JSON metadati 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 SNS elaborati su Amazon.

```
{
    "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 SNS messaggi 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 SNS messaggi Amazon archiviati in Amazon S3 utilizzando Athena

Questa pagina descrive come analizzare i SNS messaggi Amazon inviati tramite i flussi di consegna di Amazon Data Firehose verso destinazioni Amazon Simple Storage Service (Amazon S3).

Per analizzare i SNS messaggi inviati tramite i flussi di consegna 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. Usa Amazon Athena per interrogare gli oggetti Amazon S3 utilizzando standard. SQL 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:

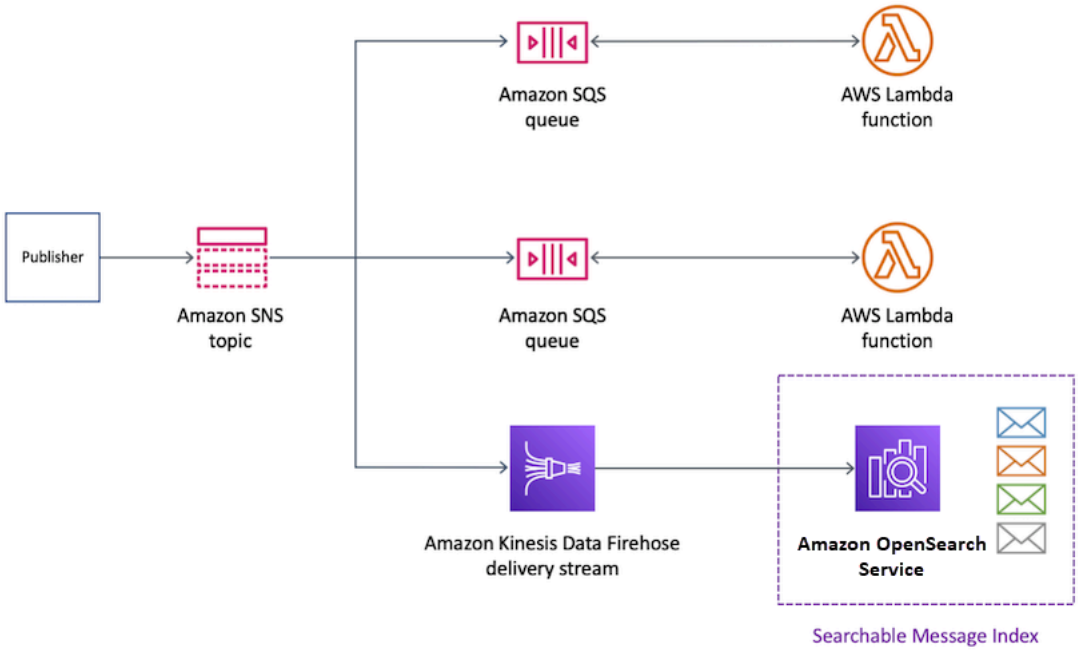
- I messaggi vengono archiviati nella tabella `notifications` nello schema `default`.
- La tabella `notifications` include una colonna `timestamp` con un tipo di `string`.

La seguente query restituisce tutti i SNS messaggi 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';
```

Integrazione dei SNS messaggi Amazon con le destinazioni di Amazon OpenSearch Service

Questa sezione fornisce informazioni sui flussi di distribuzione di Amazon Data Firehose che pubblicano dati su Amazon OpenSearch Service (Service)OpenSearch .



Argomenti

Archiviazione e formattazione di Amazon SNS Notifications negli indici OpenSearch di servizio

L'esempio seguente mostra una SNS notifica Amazon inviata a un indice Amazon OpenSearch OpenSearch Service (Service) denominato`my-index`. Questo indice ha un campo filtro tempo nella scheda `Timestamp`. La SNS notifica viene inserita nella `_source` proprietà del payload.

> **Note**
>
> In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando la consegna di messaggi non elaborati è disabilitata, Amazon SNS aggiunge JSON metadati 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 SNS elaborati su Amazon.


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "4961310096311132320325040540219328379477388655098593280 2.0",
  "_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 SNS messaggi Amazon per le destinazioni dei OpenSearch servizi

Questa pagina descrive come analizzare i SNS messaggi Amazon inviati tramite i flussi di consegna di Amazon Data Firehose verso destinazioni Amazon OpenSearch Service (OpenSearch Service).

Per analizzare i SNS messaggi inviati tramite i flussi di consegna di Firehose verso le destinazioni del Servizio OpenSearch

1. Configura le risorse del tuo OpenSearch Servizio. 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.

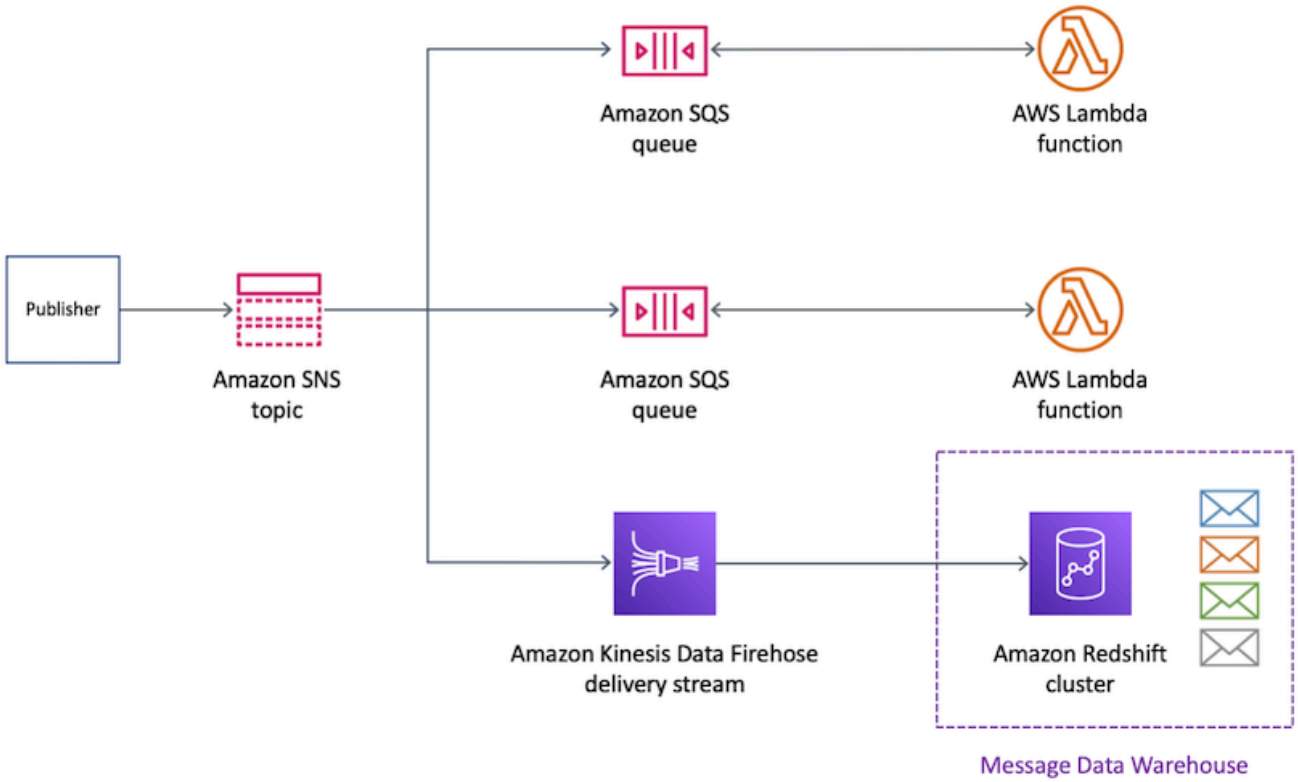
Query di esempio

L'esempio seguente esegue una query sull'`my-index`indice di tutti i SNS messaggi 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"
            }
          }
        }
      ]
    }
  }
}
```

Configurazione del recapito e dell'analisi dei SNS messaggi Amazon nelle destinazioni Amazon Redshift

Questa sezione descrive come estendere SNS le notifiche di Amazon 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 SQL query per interrogare il database alla ricerca di SNS messaggi Amazon che soddisfano determinati criteri.



Argomenti

- Strutturazione degli archivi di SNS messaggi Amazon nelle tabelle di Amazon Redshift
- Analisi dei SNS messaggi Amazon archiviati nelle destinazioni Amazon Redshift

Strutturazione degli archivi di SNS messaggi Amazon nelle tabelle di Amazon Redshift

Per gli endpoint Amazon Redshift, i SNS messaggi Amazon 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 la consegna di messaggi non elaborati è disabilitata, Amazon SNS aggiunge JSON metadati 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 SNS elaborati su Amazon.
Sebbene Amazon SNS aggiunga proprietà al messaggio utilizzando le lettere maiuscole mostrate in questo elenco, i nomi delle colonne nelle tabelle di Amazon Redshift vengono visualizzati solo in caratteri minuscoli. Per trasformare i JSON metadati per l'endpoint Amazon Redshift, puoi usare il comando. SQL `COPY` Per ulteriori informazioni, consulta Copy from JSON examples e Load from JSON data using the 'auto ignorecase' nella Amazon Redshift Database Developer Guide.

tipo	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=Annulla l'iscrizione&Subscript	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

tipo	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageattributes
						ionArn=arn:aws:sns:us-east-1:1111:my-topic:326deeeb-cb f4-45da-b92b-ca77a247813b	

tipo	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageattributes
Notification	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	Oggetto di esempio 2	Messaggio di esempio 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?action=Annulla l'iscrizione&SubscriptionArn=arn:aws:sns:us-east-1:1111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

tipo	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=Annulla l'iscrizione&SubscriptionArn=arn:aws:sns:us-east-1:1111: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 Configurazione del recapito e dell'analisi dei SNS messaggi Amazon nelle destinazioni Amazon Redshift.

Analisi dei SNS messaggi Amazon archiviati nelle destinazioni Amazon Redshift

Questa pagina descrive come analizzare i SNS messaggi Amazon inviati tramite i flussi di consegna di Amazon Data Firehose verso destinazioni Amazon Redshift.

Per analizzare i SNS messaggi 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 `Timestamp` proprietà del SNS messaggio viene memorizzata nella `timestamp` colonna della tabella con un tipo di dati di colonna di. `timestamptz`

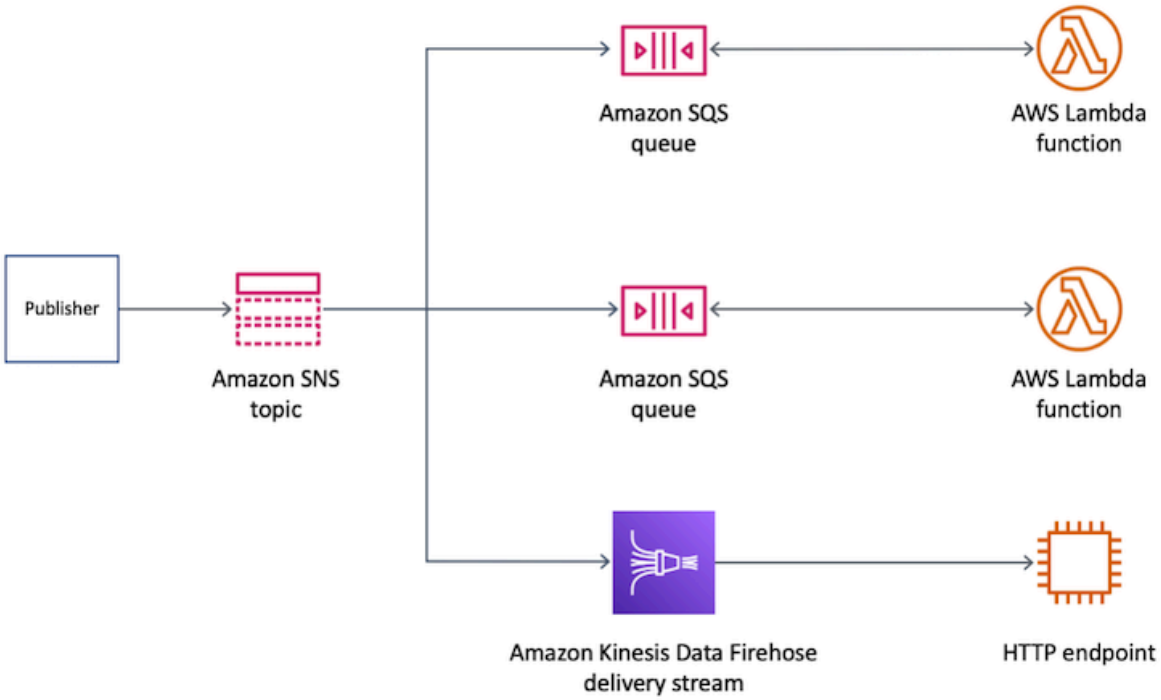
> **Note**
>
> Per trasformare i JSON metadati per l'endpoint Amazon Redshift, puoi usare il comando. SQL `COPY` Per ulteriori informazioni, consulta Copy from JSON examples e Load from JSON data using the 'auto ignorecase' nella Amazon Redshift Database Developer Guide.

La seguente query restituisce tutti i SNS messaggi 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';
```

Configurazione del recapito di SNS messaggi Amazon alle HTTP destinazioni utilizzando Amazon Data Firehose

Questa sezione fornisce informazioni sui flussi di distribuzione di Amazon Data Firehose che pubblicano dati sugli endpoint. HTTP



Argomenti

- Formato SNS di notifica Amazon per la consegna a HTTP destinazioni

Formato SNS di notifica Amazon per la consegna a HTTP destinazioni

Di seguito è riportato un esempio di corpo di HTTP POST richiesta proveniente da Amazon SNS che un flusso di distribuzione di Amazon Data Firehose può inviare all'HTTPendpoint. La SNS notifica è codificata come payload base64 nella proprietà. `records`

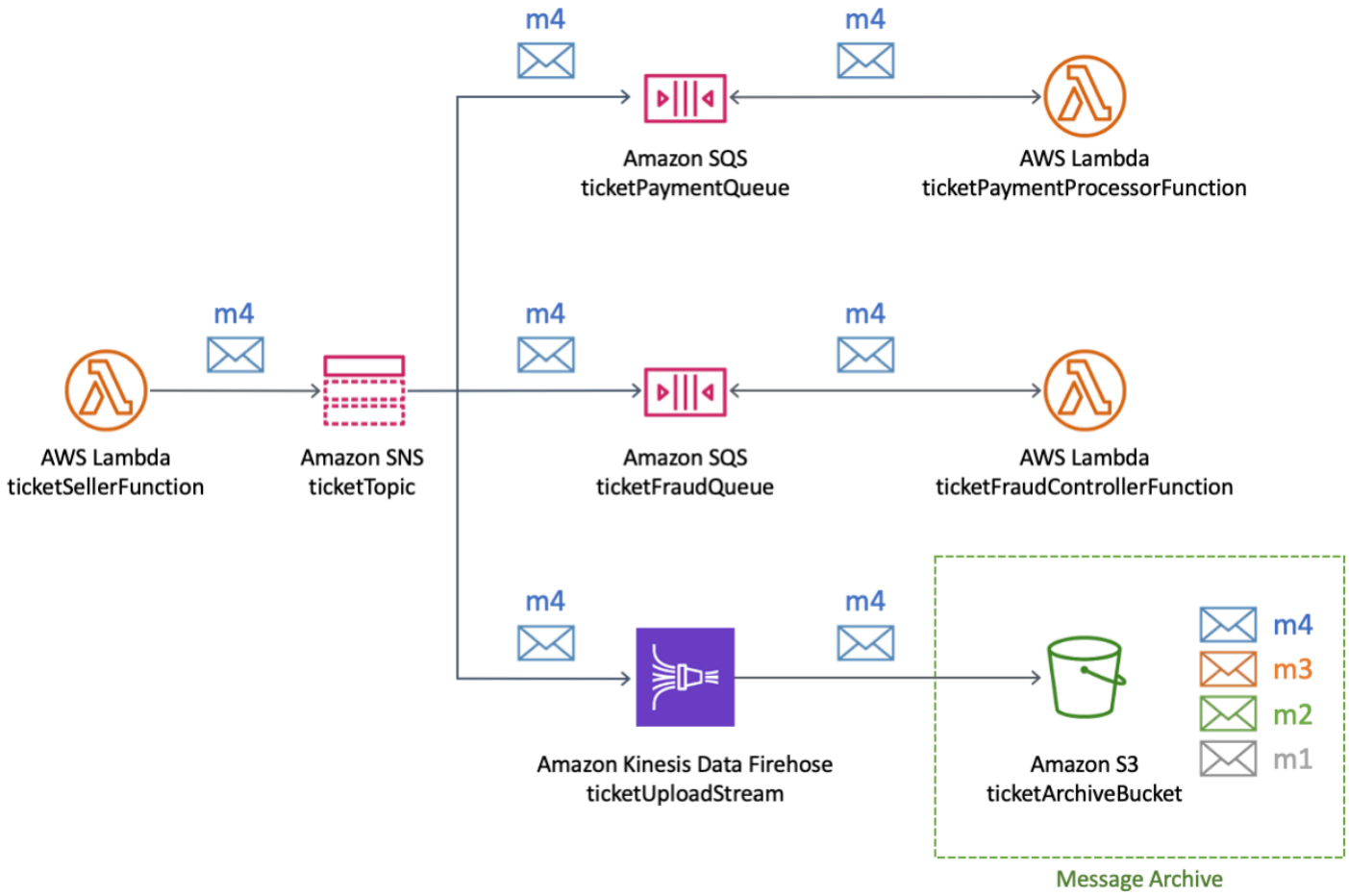
> **Note**
>
> In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Per ulteriori informazioni sulla distribuzione non elaborata, consulta Consegna di messaggi non SNS elaborati su Amazon.


```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
        "data":
 "eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```


Archiviazione e analisi dei SNS messaggi di Amazon: un esempio di utilizzo per le piattaforme di biglietteria aerea

Questa sezione fornisce un tutorial su un caso d'uso comune per l'archiviazione e l'analisi dei messaggi AmazonSNS.

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 SNS argomento esistente. La destinazione per il flusso di distribuzione è un bucket Amazon Simple Storage Service (Amazon S3). Con questa configurazione, tutti gli eventi pubblicati sull'SNSargomento vengono archiviati nel bucket Amazon S3. Nel diagramma seguente viene illustrata l'architettura di questa configurazione.



Per eseguire analisi e ottenere informazioni sulla vendita dei biglietti, l'azienda esegue SQL query utilizzando Amazon Athena. Ad esempio, l'azienda può eseguire query per conoscere le destinazioni più popolari e i volantini più frequenti.

Per creare le AWS risorse per questo caso d'uso, puoi utilizzare AWS Management Console o un AWS CloudFormation modello.

Argomenti

- Configurazione AWS delle risorse iniziali per l'archiviazione e l'analisi dei SNS messaggi di Amazon
- Configurazione di un flusso di distribuzione Firehose per l'archiviazione dei messaggi Amazon SNS
- Iscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS
- Test e interrogazione di una SNS configurazione Amazon per una gestione efficace dei dati
- Automatizzazione dell'archiviazione dei SNS messaggi di Amazon con un modello AWS CloudFormation

Configurazione AWS delle risorse iniziali per l'archiviazione e l'analisi dei SNS messaggi di Amazon

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 Amazon Simple Queue Service (AmazonSQS)
- Un SNS argomento di Amazon
- Due SQS abbonamenti Amazon 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 SQS code Amazon:

   a. Apri la SQSconsole Amazon.

   b. Scegliere Crea coda.

   c. Per Tipo, scegliere Standard.

   d. Per Nome, immetti **`ticketPaymentQueue`**.

   e. Su Policy di accesso, per Scegli il metodo, scegliere Avanzato.

   f. Nella casella della JSON politica, incolla la seguente politica:

```
{
  "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 politica di accesso, sostituisci il Account AWS numero (*123456789012*) con il tuo e cambia la AWS regione (*us-east-1*) di conseguenza.

g. Scegliere Crea coda.

h. Ripetere questi passaggi per creare una seconda SQS coda denominata**ticketFraudQueue**.

Per ulteriori informazioni sulla creazione di SQS code, consulta Creating an Amazon SQS queue (console) nella Amazon Simple Queue Service Developer Guide.

3. Crea l'argomento: SNS

   a. Apri la pagina Argomenti della SNS console Amazon.

   b. Scegliere Create topic (Crea argomento).

   c. Su Details (Dettagli), per Type (Tipo), scegliere Standard.

   d. Per Nome, immetti **ticketTopic**.

   e. Scegliere Create topic (Crea argomento).

   Per ulteriori informazioni sulla creazione di SNS argomenti, consultaCreare un SNS argomento Amazon.

4. Sottoscrivi entrambe le SQS code all'SNSargomento:

a. Nella SNSconsole Amazon, nella pagina dei dettagli dell'ticketTopicargomento, scegli Crea abbonamento.

b. In Dettagli, per Protocollo, scegli Amazon SQS.

c. Per Endpoint, scegli Amazon Resource Name (ARN) della ticketPaymentQueuecoda.

d. Scegli Crea sottoscrizione.

e. Ripeti questi passaggi per creare un secondo abbonamento utilizzando ARN la ticketFraudQueuecoda.

   Per ulteriori informazioni sulla sottoscrizione agli SNS argomenti, vedere. Creare un abbonamento a un SNS argomento di Amazon Puoi anche iscriverti alle SQS code agli SNS argomenti dalla SQS console Amazon. Per ulteriori informazioni, consulta la sezione Abbonamento di una SQS coda Amazon a un SNS argomento Amazon (console) nella Amazon Simple Queue Service Developer Guide.

Sono state create le risorse iniziali per questo caso d'uso di esempio. Per continuare, consulta Configurazione di un flusso di distribuzione Firehose per l'archiviazione dei messaggi Amazon SNS.

Configurazione di un flusso di distribuzione Firehose per l'archiviazione dei messaggi Amazon SNS

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 Configura impostazioni, per Autorizzazioni, scegli di creare un ruolo AWS Identity and Access Management (IAM) con le autorizzazioni richieste 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 scheda Dettagli, annota l'Amazon Resource Name (ARN) dello stream per dopo.

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 IAM ruoli, consulta Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella Guida per l'utente. IAM

Hai creato il flusso di distribuzione di Firehose con le autorizzazioni richieste. Per continuare, consulta Iscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS.

Iscrizione 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'SNSabbonamento Amazon di inserire record nel flusso di distribuzione di Amazon Data Firehose
- L'abbonamento Firehose Delivery Stream all'argomento SNS

Per creare il IAM ruolo per l'SNSabbonamento Amazon

1. Apri la pagina Ruoli della IAM console.

2. Scegliere Create role (Crea ruolo).
3. In Select type of trusted entity (Seleziona tipo di entità attendibile), scegli AWS service (Servizio).
4. Per Scegli un caso d'uso, scegli 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, sceglietene il nome (ticketUploadStreamSubscriptionRole).
9. Nrlla pagina Riepilogo del ruolo, scegliere Aggiunta di policy inline.
10. Nella pagina Crea politica, scegli la JSONscheda, quindi incolla la seguente politica nella casella:

```
{
    "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 politica, sostituisci il Account AWS numero (*`123456789012`*) con il tuo e cambia la AWS regione (*`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, annota il ruolo ARN per dopo.

Per ulteriori informazioni sulla creazione di IAM ruoli, vedere Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella Guida per l'IAMutente.

Per abbonare lo stream di distribuzione di Firehose all'argomento SNS

1. Apri la pagina Argomenti della SNS console Amazon.
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) del flusso di ticketUploadStreamdistribuzione creato in precedenza. Ad esempio, specifica **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Per il ruolo Subscription ARN, inserisci il ARN ticketUploadStreamSubscriptionRoleIAMruolo che hai creato in precedenza. Ad esempio, specifica **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Selezionare l'Abilitazione del recapito di messaggi.
7. Scegli Crea sottoscrizione.

Hai creato la sottoscrizione al IAM ruolo e all'SNSargomento. Per continuare, consulta Test e interrogazione di una SNS configurazione Amazon per una gestione efficace dei dati.

Test e interrogazione di una SNS configurazione Amazon per una gestione efficace dei dati

Questa pagina descrive come testare il caso d'uso di esempio di archiviazione e analisi dei messaggi pubblicando un messaggio SNS sull'argomento Amazon. Le istruzioni includono una query di esempio che è possibile eseguire e adattare alle proprie esigenze.

Per testare la configurazione

1. Apri la pagina Argomenti della SNS console Amazon.
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 di un SNS messaggio Amazon.

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:

- SNSAbbonamenti Amazon
- SNSArgomento Amazon
- Code di Amazon Simple Queue Service (AmazonSQS)
- Bucket Amazon S3
- Flusso di distribuzione di Amazon Data Firehose
- AWS Identity and Access Management (IAM) ruoli e politiche

Automatizzazione dell'archiviazione dei SNS messaggi di Amazon con un modello AWS CloudFormation

Per automatizzare la distribuzione del caso d'uso di esempio di Amazon SNS Message Archiving and Analytics, puoi 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
```

Fanout SNS delle notifiche Amazon alle funzioni Lambda per l'elaborazione automatizzata

Amazon SNS e AWS Lambda sono integrati in modo da poter richiamare le funzioni Lambda con le notifiche AmazonSNS. Quando un messaggio viene pubblicato su un SNS argomento a cui è stata sottoscritta una funzione Lambda, 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, pubblicarlo SNS su altri argomenti o inviarlo ad altri servizi. AWS

Amazon supporta SNS anche gli attributi dello stato di consegna dei messaggi per le notifiche dei messaggi inviate agli endpoint Lambda. Per ulteriori informazioni, consulta Stato di consegna dei SNS messaggi Amazon.

Argomenti

Prerequisiti per l'integrazione di Amazon con le funzioni SNS Lambda in tutte le regioni

Per richiamare le funzioni Lambda utilizzando le notifiche di SNS Amazon, è necessario quanto segue:

- Funzione Lambda
- SNSArgomento Amazon

Per informazioni sulla creazione di una funzione Lambda da utilizzare con AmazonSNS, consulta Using Lambda with Amazon. SNS Per informazioni sulla creazione di un SNS argomento Amazon, consulta Creare un argomento.

Quando utilizzi Amazon SNS per recapitare messaggi da regioni con consenso esplicito a regioni abilitate per impostazione predefinita, devi modificare la politica creata nella funzione AWS Lambda sostituendo la `sns.amazonaws.com` principale con. `sns.<opt-in-region>.amazonaws.com`

Ad esempio, se desideri sottoscrivere una funzione Lambda negli Stati Uniti orientali (Virginia settentrionale) a un SNS argomento in Asia Pacifico (Hong Kong), modifica il principale nella politica della 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 interregionale a Lambda da una regione abilitata per impostazione predefinita a una regione opt-in. Inoltre, l'inoltro interregionale di SNS messaggi da regioni opt-in ad altre aree opt-in non è supportato.

Sottoscrizione di una funzione Lambda a un argomento Amazon SNS

1. Accedi alla SNSconsole Amazon.

2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti) scegliere un argomento.
4. Nella sezione Subscriptions (Sottoscrizioni) scegliere Create subscription (Crea sottoscrizione).
5. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:

    a. Verifica l'argomento sceltoARN.
    b. Per Protocollo scegli AWS Lambda.
    c. Per Endpoint, inserisci il nome ARN di una funzione.
    d. Scegli Crea sottoscrizione.

Quando un messaggio viene pubblicato su un SNS argomento a cui è stata sottoscritta una funzione Lambda, la funzione Lambda viene richiamata con il payload del messaggio pubblicato. Per informazioni sull'utilizzo AWS Lambda con AmazonSNS, incluso un tutorial, consulta Using AWS Lambda with Amazon SNS.

Fanout SNS delle notifiche Amazon alle SQS code Amazon per l'elaborazione asincrona

Amazon SNS collabora a stretto contatto con Amazon Simple Queue Service (AmazonSQS). Questi servizi offrono numerosi vantaggi agli sviluppatori. Amazon SNS consente alle applicazioni di inviare messaggi urgenti a più abbonati tramite un meccanismo «push», eliminando la necessità di controllare o «verificare» periodicamente gli aggiornamenti. Amazon SQS è un servizio di coda di messaggi utilizzato da applicazioni distribuite per scambiare messaggi tramite un modello di polling e può essere utilizzato per disaccoppiare i componenti di invio e ricezione, senza richiedere che ogni componente sia disponibile contemporaneamente. Utilizzando Amazon SNS e Amazon SQS insieme, i messaggi possono essere recapitati ad applicazioni che richiedono una notifica immediata di un evento e possono anche essere mantenuti in una SQS coda Amazon per l'elaborazione successiva da parte di altre applicazioni.

Quando sottoscrivi una SQS coda Amazon a un SNS argomento Amazon, puoi pubblicare un messaggio sull'argomento e Amazon SNS invia un SQS messaggio Amazon alla coda degli abbonati. Il SQS messaggio Amazon contiene l'oggetto e il messaggio pubblicati sull'argomento insieme ai metadati relativi al messaggio in un JSON documento. Il SQS messaggio di Amazon sarà simile al seguente JSON documento.

```
{
```

```
   "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"
}
```

Sottoscrizione di una SQS coda Amazon a un argomento Amazon SNS

Per consentire a un SNS argomento Amazon di inviare messaggi a una SQS coda Amazon, esegui una delle seguenti operazioni:

- Usa la SQSconsole Amazon, che semplifica il processo. Per ulteriori informazioni, consulta l'SNSargomento Sottoscrizione di Amazon SQS queue to an Amazon nella Amazon Simple Queue Service Developer Guide.
- Completare la procedura riportata di seguito.
  1. Ottieni l'Amazon Resource Name (ARN) della coda a cui desideri inviare messaggi e l'argomento a cui desideri iscriverti alla coda.
  2. `sqs:SendMessage`Autorizza l'SNSargomento Amazon in modo che possa inviare messaggi alla coda.
  3. Iscriviti alla coda all'SNSargomento Amazon.
  4. Concedi IAM agli utenti o Account AWS le autorizzazioni appropriate per pubblicare SNS sull'argomento Amazon e leggere i messaggi dalla SQS coda Amazon.
  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 SNS di messaggi Amazon a una SQS coda Amazon in un altro account.

Per vedere un AWS CloudFormation modello che crea un argomento che invia messaggi a due code, consulta. Automatizza la SQS messaggistica SNS da Amazon ad Amazon con AWS CloudFormation

Fase 1: Eliminare ARN la coda e l'argomento

Quando ti iscrivi a una coda al tuo argomento, avrai bisogno di una copia del file ARN per la coda. Allo stesso modo, quando autorizzi l'argomento a inviare messaggi alla coda, avrai bisogno di una copia dell'ARNargomento.

Per ottenere la codaARN, puoi utilizzare la SQS console Amazon o l'GetQueueAttributesAPIazione.

Per ottenere la coda ARN dalla console Amazon SQS

1. Accedi a AWS Management Console e apri la SQS console Amazon all'indirizzo https://console.aws.amazon.com/sqs/.
2. Seleziona la casella per la coda a cui ARN vuoi accedere.
3. Dalla sezione Dettagli, copia il ARN valore in modo da poterlo utilizzare per abbonarti all'SNSargomento Amazon.

Per approfondire l'argomentoARN, puoi utilizzare la SNS console Amazon, il `sns-get-topic-attributes` comando o l'`GetQueueAttributes`APIazione.

Per scaricare l'argomento ARN dalla SNS console Amazon

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegli l'argomento di cui ARN desideri accedere.
3. Dalla sezione Dettagli, copia il ARNvalore in modo da poterlo utilizzare per autorizzare l'SNSargomento Amazon a inviare messaggi alla coda.

Passaggio 2: autorizza l'SNSargomento Amazon a inviare messaggi alla SQS coda Amazon

Affinché un SNS argomento Amazon sia in grado di inviare messaggi a una coda, devi impostare una politica sulla coda che consenta all'SNSargomento Amazon di eseguire l'`sqs:SendMessage`azione.

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 politica su una coda, puoi utilizzare la SQS console Amazon o l'SetQueueAttributesAPIazione. Prima di iniziare, assicurati di avere ARN l'argomento a cui desideri consentire l'invio di messaggi alla coda. Se stai sottoscrivendo una coda a più argomenti, la policy deve contenere un elemento `Statement` per ogni argomento.

Per impostare una SendMessage policy su una coda utilizzando la console Amazon SQS

1. Accedi a AWS Management Console e apri la SQS console Amazon 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.
   - Impostato `Principal` per essere il SNS servizio Amazon, 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 condizionali, imposta il ARN valore sul tuo argomento. Se la coda è sottoscritta a più argomenti, è possibile usare invece `aws:SourceAccount`.

   Ad esempio, la seguente politica consente MyTopic di inviare messaggi a MyQueue.

```
{
  "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"
        }
      }
    }
  ]
```

```
}
```

Passaggio 3: iscriviti alla coda all'argomento Amazon SNS

Per inviare messaggi a una coda tramite un argomento, devi iscrivere la coda all'argomento AmazonSNS. Specifichi la coda in base alla sua. ARN Per iscriverti a un argomento, puoi utilizzare la SNS console Amazon, il `sns-subscribe` CLI comando o l'`Subscribe`APIazione. Prima di iniziare, assicurati di avere ARN la coda a cui desideri iscriverti.

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti) scegliere un argomento.
4. Sul ***`MyTopic`*** nella pagina Abbonamenti, scegli Crea abbonamento.
5. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:

   a. Verifica l'argomento ARN.

   b. Per Protocol, scegli Amazon SQS.

   c. Per Endpoint, inserisci il nome ARN di una SQS coda Amazon.

   d. Selezionare Create Subscription (Crea abbonamento).

   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 SNS di messaggi Amazon a una SQS coda Amazon in un altro account.

Fase 4: concedere agli utenti le autorizzazioni per le operazioni appropriate su argomenti e code

Dovresti usare AWS Identity and Access Management (IAM) per consentire solo agli utenti appropriati di pubblicare sull'SNSargomento Amazon e di leggere/eliminare i messaggi dalla coda AmazonSQS. Per ulteriori informazioni sul controllo delle azioni su argomenti e code per IAM gli utentiUtilizzo di politiche basate sull'identità con Amazon SNS, consulta la pagina Gestione delle identità e degli accessi in Amazon SQS nella Amazon Simple Queue Service Developer Guide.

Esistono due modi di controllare l'accesso a un argomento o a una coda:

- Aggiungi una policy a un IAM utente o a un gruppo. 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 desideri concedere le autorizzazioni a un argomento o aggiungere una coda a un altro AWS account, l'unico modo per farlo è aggiungere una politica che abbia come principale la politica a Account AWS cui desideri 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.

Aggiungere una politica a un utente o a un gruppo IAM

Se aggiungessi la seguente politica a un IAM utente o a un gruppo, concederesti all'utente o ai membri di quel gruppo l'autorizzazione a eseguire l'`sns:Publish`azione sull'argomento MyTopic.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Se aggiungessi la seguente politica a un IAM utente o a un gruppo, concederesti all'utente o ai membri di quel gruppo l'autorizzazione a eseguire `sqs:DeleteMessage` le azioni `sqs:ReceiveMessage` and sulle code MyQueue 1 e MyQueue 2.

```
{
  "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'altra persona Account AWS l'accesso a una risorsa del tuo account, concedi anche agli IAM utenti che dispongono di autorizzazioni di accesso a livello di amministratore (accesso con wildcard) a quella risorsa. A tutti IAM gli altri utenti dell'altro account viene automaticamente negato l'accesso alla tua risorsa. Se desideri IAM consentire a utenti specifici di Account AWS accedere alla tua risorsa, l'account o un IAM utente con accesso a livello di amministratore deve delegare le autorizzazioni per la risorsa a tali utenti. IAM Per ulteriori informazioni sulla delega tra account, vedere Abilitazione dell'accesso su più account nella Guida all'uso. IAM

Se hai aggiunto la seguente politica a un argomento MyTopic nell'account 123456789012, daresti all'account 111122223333 il permesso di eseguire l'azione su quell'argomento. `sns:Publish`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Se hai aggiunto la seguente politica a una coda MyQueue nell'account 123456789012, daresti all'account 111122223333 l'autorizzazione a eseguire le azioni and su quella coda. `sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "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 su un argomento utilizzando la SNS console Amazon

1. Utilizzando le credenziali dell'IAMutente Account AWS o con autorizzazione alla pubblicazione sull'argomento, accedi AWS Management Console e apri la SNS console Amazon all'indirizzo 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).

Per visualizzare il messaggio dell'argomento utilizzando la SQS console Amazon

1. Utilizzando le credenziali dell'IAMutente Account AWS o autorizzato a visualizzare i messaggi in coda, accedi AWS Management Console e apri la SQS console Amazon all'indirizzo. 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 Dettagli del messaggio contiene un JSON documento che contiene l'oggetto e il messaggio che hai pubblicato sull'argomento. Il messaggio è simile al seguente JSON documento.

```
{
  "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.

Automatizza la SQS messaggistica SNS da Amazon ad Amazon con AWS CloudFormation

AWS CloudFormation consente di utilizzare un file modello per creare e configurare una raccolta di AWS risorse insieme 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 si occupano automaticamente dei passaggi di configurazione creando due code, creando un argomento con sottoscrizioni alle code, aggiungendo una politica alle code in modo che l'argomento possa inviare messaggi alle code e creando IAM utenti e gruppi per controllare l'accesso a tali risorse.

Per ulteriori informazioni sulla distribuzione AWS delle risorse utilizzando un AWS CloudFormation modello, consulta la Guida introduttiva nella Guida per l'utente.AWS CloudFormation

Utilizzo di un AWS CloudFormation modello per configurare argomenti e code all'interno di un Account AWS

Il modello di esempio crea un SNS argomento Amazon in grado di inviare messaggi a due SQS code Amazon con le autorizzazioni appropriate per consentire ai membri di un IAM gruppo di pubblicare sull'argomento e un altro di leggere i messaggi dalle code. Il modello crea anche IAM utenti che vengono aggiunti a ciascun gruppo.

Copiare il contenuto del modello in un file. Puoi anche scaricare il modello dalla pagina AWS CloudFormation Modelli. Nella pagina dei modelli, scegli Sfoglia modelli di esempio per AWS servizio, quindi scegli Amazon Simple Queue Service.

M ySNSTopic è configurato per pubblicare su due endpoint sottoscritti, che sono due SQS code Amazon (MyQueue1 e MyQueue 2). MyPublishTopicGroup è un IAM gruppo i cui membri sono autorizzati a pubblicare su M ySNSTopic utilizzando l'APIazione Publish o il comando sns-publish. Il modello crea gli IAM utenti MyPublishUser MyQueueUser e fornisce loro profili di accesso 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 IAM utenti con MyPublishUserKey e MyQueueUserKey. AddUserToMyPublishTopicGroup si aggiunge

MyPublishUser a MyPublishTopicGroup in modo che all'utente vengano assegnate le autorizzazioni al gruppo.

M yRDMessage QueueGroup è un IAM gruppo i cui membri sono autorizzati a leggere ed eliminare messaggi dalle due SQS code di Amazon utilizzando le DeleteMessageAPIazioni ReceiveMessageand. AddUserToMyQueueGroup si aggiunge MyQueueUser alla M yRDMessage QueueGroup in modo che all'utente vengano assegnate le autorizzazioni al gruppo. MyQueuePolicy assegna il permesso ySNSTopic a M di pubblicare le sue notifiche nelle due code.

L'elenco seguente mostra il contenuto del AWS CloudFormation modello.

```
{
  "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 SNS delle notifiche Amazon agli endpoint HTTPS

Puoi utilizzare Amazon SNS per inviare messaggi di notifica a uno HTTP o più HTTPS endpoint. Quando sottoscrivi un endpoint a un argomento, puoi pubblicare una notifica sull'argomento e Amazon SNS invia una HTTP POST richiesta per recapitare il contenuto della notifica all'endpoint sottoscritto. Quando sottoscrivi l'endpoint, scegli se Amazon SNS utilizzare HTTP o HTTPS inviare la POST richiesta all'endpoint. Se lo utilizziHTTPS, puoi usufruire del supporto di Amazon SNS per quanto segue:

- Indicazione del nome del server (SNI): consente SNS ad Amazon di supportare HTTPS endpoint che richiedonoSNI, ad esempio, un server che richiede più certificati per ospitare più domini. Per ulteriori informazioni suSNI, consulta Server Name Indication.

- Autenticazione di base e Digest Access: consente di specificare un nome utente e una password nella HTTP POST richiesta, ad esempio `https://user:password@domain.com` o Il nome utente e `https://user@domain.com` la password vengono crittografati sulla SSL connessione stabilita durante l'utilizzo. HTTPS URL HTTPS Solo il nome di dominio viene inviato come testo normale. Per ulteriori informazioni sull'autenticazione Basic e Digest Access, vedere -2617. RFC

> **Important**
>
> Amazon attualmente SNS non supporta gli endpoint privati HTTP (S). HTTPSURLssono recuperabili solo dall'`SNSGetSubscriptionAttributes`APIazione Amazon, per i principali a cui hai concesso l'accesso. 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 che sono stati pubblicati sull'argomento insieme ai metadati relativi alla notifica in un documento. JSON La richiesta sarà simile alla HTTP POST richiesta seguente. Per dettagli sull'HTTPintestazione e sul JSON formato del corpo della richiesta, consulta HTTP/HTTPSintestazioni eHTTP/formato HTTPS di notifica 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

- Sottoscrizione di un HTTPS endpoint a un argomento Amazon SNS
- Verifica delle firme dei messaggi Amazon SNS
- Analisi dei formati dei SNS messaggi Amazon

Sottoscrizione di un HTTPS endpoint a un argomento Amazon SNS

Le pagine di questa sezione descrivono come sottoscrivere gli endpoint HTTP /S agli SNS argomenti di Amazon.

Argomenti

- Passaggio 1: assicurati che l'endpoint sia pronto per elaborare i messaggi Amazon SNS
- Passaggio 2: sottoscrivi l'HTTPSendpointHTTP/all'argomento Amazon SNS
- Passaggio 3: conferma l'SNSabbonamento Amazon
- Passaggio 4: Facoltativo: imposta la politica di spedizione per l'SNSabbonamento Amazon
- Passaggio 5: Facoltativo: concedi agli utenti le autorizzazioni per la pubblicazione sull'argomento Amazon SNS
- Passaggio 6: invio di SNS messaggi Amazon all'HTTPSendpointHTTP/

Passaggio 1: assicurati che l'endpoint sia pronto per elaborare i messaggi Amazon SNS

Prima di sottoscrivere il tuo HTTP o il tuo HTTPS endpoint a un argomento, devi assicurarti che l'HTTPSendpoint HTTP o sia in grado di gestire le HTTP POST richieste SNS utilizzate da Amazon per inviare i messaggi di conferma e notifica dell'abbonamento. Di solito, ciò significa creare e distribuire un'applicazione Web (ad esempio, un servlet Java se l'host dell'endpoint esegue Linux con Apache e Tomcat) che elabora le richieste provenienti da Amazon. HTTP SNS Quando sottoscrivi un HTTP endpoint, Amazon SNS invia una richiesta di conferma dell'abbonamento. Il tuo endpoint deve essere pronto a ricevere ed elaborare questa richiesta al momento della creazione dell'abbonamento, poiché Amazon SNS invia la richiesta in quel momento. Amazon non SNS invierà notifiche all'endpoint finché non confermerai l'abbonamento. Una volta confermato l'abbonamento, Amazon SNS invierà notifiche all'endpoint quando viene eseguita un'azione di pubblicazione sull'argomento sottoscritto.

Per impostare l'endpoint in modo che elabori i messaggi di conferma della sottoscrizione e di notifica

1. Il codice dovrebbe leggere le HTTP intestazioni delle HTTP POST richieste che Amazon SNS invia al tuo endpoint. Il codice dovrebbe cercare il campo dell'intestazione`x-amz-sns-message-type`, che indica il tipo di messaggio che Amazon ti SNS ha inviato. Osservando l'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 informazioni dettagliate sull'HTTPintestazione, consulta. HTTP/HTTPSintestazioni La seguente HTTP POST richiesta è un esempio di messaggio di conferma dell'iscrizione.

```
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 JSON documento nel corpo della HTTP POST richiesta e nel tipo di contenuto text/plain per leggere le coppie nome-valore che compongono il messaggio Amazon. SNS Usa un JSON parser che gestisca la conversione della rappresentazione in escape dei caratteri di controllo nei rispettivi valori di carattere (ad esempio, convertendo \nin ASCII un carattere di nuova riga). È possibile utilizzare un JSON parser esistente come Jackson Processor o scriverne uno personalizzato. JSON Per inviare il testo nei campi oggetto e messaggio come validoJSON, Amazon SNS deve convertire alcuni caratteri di controllo in rappresentazioni escape che possono essere incluse nel JSON documento. Quando ricevi il JSON documento nel corpo della POST richiesta inviata al tuo dispositivo, devi riconvertire i caratteri sfuggiti ai valori dei caratteri originali se desideri una rappresentazione esatta dell'oggetto e dei messaggi originali pubblicati sull'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 una notifica, di un messaggio di conferma dell'abbonamento o dell'annullamento dell'iscrizione inviato da Amazon. SNS Utilizzando le informazioni contenute nel SNS messaggio Amazon, il tuo endpoint può ricreare la firma in modo da poter verificare il contenuto del messaggio abbinando la tua firma alla 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 dovrebbe leggere il JSON documento contenuto nel corpo della HTTP richiesta ed elaborare il messaggio. Di seguito sono riportate le linee guida per gestire i due tipi di messaggi principali:

SubscriptionConfirmation

Leggete il valore `SubscribeURL` e visitateloURL. Per confermare l'abbonamento e iniziare a ricevere notifiche sull'endpoint, devi visitare il `SubscribeURL` URL (ad esempio, inviando una HTTP GET richiesta aURL). Vedi la HTTP richiesta di esempio nel passaggio precedente per vedere che `SubscribeURL` aspetto ha. Per ulteriori informazioni sul formato del messaggio `SubscriptionConfirmation`, consulta HTTP/formato HTTPS di conferma dell'iscrizione JSON. Quando visiti ilURL, riceverai una risposta simile al seguente XML documento. Il documento restituisce l'abbonamento ARN per l'endpoint all'interno dell'`ConfirmSubscriptionResult`elemento.

```
<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>
```

In alternativa alla visita di`SubscribeURL`, è possibile confermare l'iscrizione utilizzando l'ConfirmSubscriptionazione con il valore corrispondente `Token` impostato nel `SubscriptionConfirmation` messaggio. 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 HTTP/HTTPSintestazioni. La HTTP POST richiesta 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 il tuo endpoint risponda al HTTP POST messaggio di Amazon SNS con il codice di stato appropriato. La connessione scadrà tra circa 15 secondi. Se l'endpoint non risponde prima del timeout della connessione o se restituisce un codice di stato compreso tra 200 e 4 xx, Amazon SNS considererà la consegna del messaggio come un tentativo fallito.
6. Assicurati che il tuo codice sia in grado di gestire i tentativi di recapito dei messaggi da AmazonSNS. Se Amazon SNS non riceve una risposta corretta dal tuo endpoint, tenta di recapitare nuovamente il messaggio. Ciò vale per tutti i messaggi, incluso il messaggio di conferma della sottoscrizione. Per impostazione predefinita, se il recapito iniziale del messaggio fallisce, Amazon SNS tenta fino a tre tentativi con un ritardo tra i tentativi falliti impostato su 20 secondi.

> **Note**
>
> La richiesta del messaggio scade dopo circa 15 secondi. Ciò significa che, se l'errore di recapito del messaggio è causato da un timeout, Amazon SNS riprova per circa 35

secondi dopo il precedente tentativo di consegna. È possibile impostare una policy di consegna diversa per l'endpoint.

Amazon SNS utilizza il campo di `x-amz-sns-message-id` intestazione per identificare in modo univoco ogni messaggio pubblicato su un argomento AmazonSNS. Confrontando i IDs messaggi che hai elaborato con i messaggi in arrivo, puoi determinare se si tratta di un nuovo tentativo.

7. Se stai sottoscrivendo un HTTPS endpoint, assicurati che l'endpoint disponga di un certificato server rilasciato da un'autorità di certificazione (CA) affidabile. Amazon SNS invierà messaggi solo agli HTTPS endpoint che dispongono di un certificato server firmato da una CA considerata affidabile da AmazonSNS.
8. Implementa il codice che hai creato per ricevere SNS messaggi Amazon. Quando effettui la sottoscrizione dell'endpoint, questo deve essere pronto a ricevere almeno il messaggio di conferma della sottoscrizione.

Passaggio 2: sottoscrivi l'HTTPSendpointHTTP/all'argomento Amazon SNS

Per inviare messaggi a un HTTPS endpoint HTTP o tramite un argomento, devi sottoscrivere l'endpoint all'argomento AmazonSNS. Specifichi l'endpoint utilizzando il suo. URL Per iscriverti a un argomento, puoi utilizzare la SNS console Amazon, il comando sns-subscribe o l'azione APISubscribe. Prima di iniziare, assicurati di avere l'URLendpoint a cui desideri sottoscrivere e che il tuo endpoint sia pronto a ricevere i messaggi di conferma e notifica come descritto nel passaggio 1.

Per sottoscrivere un argomento HTTP o un HTTPS endpoint utilizzando la console Amazon SNS

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Scegli l'opzione Create Subscription (Crea sottoscrizione).
4. Nell'elenco a discesa Protocollo, seleziona HTTPo. HTTPS
5. Nella casella Endpoint, incolla il messaggio URL relativo all'endpoint a cui desideri inviare l'argomento, quindi scegli Crea abbonamento.
6. Viene visualizzato il messaggio di conferma. Scegli Chiudi.

   Viene visualizzato l'ID di abbonamento del tuo nuovo abbonamento. PendingConfirmation Dopo la conferma della sottoscrizione, il campo Subscription ID (ID sottoscrizione) visualizzerà il relativo ID.

Passaggio 3: conferma l'SNSabbonamento Amazon

Per confermare un SNS abbonamento AWS Amazon, segui questi passaggi per assicurarti che il tuo endpoint possa ricevere correttamente i messaggi. Questo processo prevede la configurazione dell'endpoint per gestire i messaggi di conferma in arrivo, il recupero della conferma necessaria e la conferma dell'URLabbonamento tramite mezzi automatici o manuali.

1. Messaggio di conferma dell'iscrizione. Dopo aver sottoscritto il tuo endpoint a un SNS argomento Amazon, Amazon SNS invia un messaggio di conferma a quell'endpoint. Questo messaggio contiene un messaggio`SubscribeURL`, che ti serve per confermare l'abbonamento.
2. Recupera il. **`SubscribeURL`** Il tuo endpoint dovrebbe avere un codice che ascolta ed elabora i messaggi in arrivo. Questo codice deve estrarre il codice `SubscribeURL` dal messaggio di conferma. Il messaggio di conferma in genere arriva come JSON payload con la `SubscribeURL` chiave.
3. Conferma l'abbonamento. Esistono due modi per confermare l'iscrizione:
   - Conferma automatica. Il codice dell'endpoint può accedere automaticamente `SubscribeURL` a per confermare l'abbonamento. Questo approccio richiede che l'endpoint effettui una HTTP GET richiesta al URL provider.
   - Conferma manuale. Se la conferma automatica non è impostata, è possibile visitarla manualmente `SubscribeURL` utilizzando un browser Web. Questo passaggio prevede di copiare il messaggio URL dal messaggio e incollarlo nella barra degli indirizzi del browser.
4. Verifica lo stato dell'abbonamento. Dopo aver confermato l'abbonamento visitando il`SubscribeURL`, Amazon SNS invia una risposta che include un XML documento con un elemento chiamato`SubscriptionArn`. Questo elemento contiene l'Amazon Resource Name (ARN) per l'abbonamento, che indica che l'abbonamento è attivo.
5. Usa la SNS console Amazon. Puoi anche verificare lo stato dell'abbonamento utilizzando il AWS Management Console. Vai alla SNS dashboard di Amazon e, nella sezione Abbonamenti, trova il tuo abbonamento. Un abbonamento confermato mostrerà il suoARN, mentre un abbonamento non confermato verrà visualizzato. `PendingConfirmation`

Passaggio 4: Facoltativo: imposta la politica di spedizione per l'SNSabbonamento Amazon

Per impostazione predefinita, se il recapito iniziale del messaggio fallisce, Amazon SNS tenta fino a tre tentativi con un ritardo tra i tentativi falliti impostato su 20 secondi. Come descritto nella fase

1, l'endpoint deve disporre di codice in grado di gestire i tentativi ripetuti di consegna dei messaggi. Impostando la politica di consegna su un argomento o un abbonamento, puoi controllare la frequenza e l'intervallo in cui Amazon riprova i SNS messaggi non riusciti. Puoi anche specificare il tipo di contenuto per le tue notifiche HTTP /S in. `DeliveryPolicy` Per ulteriori informazioni, consulta Creazione di una politica di consegna HTTP /S.

Passaggio 5: Facoltativo: concedi agli utenti le autorizzazioni per la pubblicazione sull'argomento Amazon SNS

Per default il proprietario dell'argomento dispone delle autorizzazioni per pubblicare nell'argomento. Per consentire ad altri utenti o applicazioni di pubblicare sull'argomento, devi utilizzare AWS Identity and Access Management (IAM) per autorizzare la pubblicazione sull'argomento. Per ulteriori informazioni sulla concessione delle autorizzazioni per le SNS azioni Amazon agli IAM utenti, consultaUtilizzo di politiche basate sull'identità con Amazon SNS.

Vi sono due modi di controllare l'accesso a un argomento:

- Aggiungi una policy a un IAM utente o a un gruppo. 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 seguente politica a un IAM utente o a un gruppo, concederesti all'utente o ai membri di quel gruppo l'autorizzazione a eseguire l'`sns:Publish`azione 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'altra persona Account AWS l'accesso a una risorsa del tuo account, concedi anche agli IAM utenti che dispongono di autorizzazioni di accesso a livello di amministratore (accesso con wildcard) a quella risorsa. A tutti IAM gli altri utenti dell'altro account viene automaticamente negato l'accesso alla tua risorsa. Se desideri IAM consentire a utenti specifici di Account AWS accedere alla tua risorsa, l'account o un IAM utente con accesso a livello di amministratore deve delegare le autorizzazioni per la risorsa a tali utenti. IAM Per ulteriori informazioni sulla delega tra account, vedere Abilitazione dell'accesso su più account nella Guida all'uso. IAM

Se hai aggiunto la seguente politica a un argomento MyTopic nell'account 123456789012, daresti all'account 111122223333 il permesso di eseguire l'azione su quell'argomento. `sns:Publish`

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Passaggio 6: invio di SNS messaggi Amazon all'HTTPSendpointHTTP/

Puoi inviare un messaggio alle sottoscrizioni di un argomento pubblicando nell'argomento. Per pubblicare su un argomento, puoi utilizzare la SNS console Amazon, il `sns-publish` CLI comando o il `Publish`API.

Se hai seguito le istruzioni nella fase 1, il codice distribuito all'endpoint è in grado di elaborare la notifica.

Per pubblicare su un argomento utilizzando la SNS console Amazon

1. Utilizzando le credenziali dell'IAMutente Account AWS o con l'autorizzazione a pubblicare sull'argomento, accedi AWS Management Console e apri la SNS console Amazon 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 al tuo HTTP endpoint da AmazonSNS, 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 al tuo HTTP endpoint indicando che ti sei iscritto a un argomento. In secondo luogo, quando Amazon ti SNS invia un messaggio di conferma al tuo HTTP endpoint dopo l'`Subscribe`esecuzione delle `Unsubscribe` API azioni.

Quando verifichi i messaggi inviati da AmazonSNS, devi fare quanto segue:

- Utilizzalo sempre HTTPS quando ricevi il certificato da AmazonSNS.
- Convalida l'autenticità del certificato.
- Verifica che il certificato sia stato ricevuto da AmazonSNS.
- Se possibile, utilizza uno dei servizi supportati da Amazon AWS SDKs per SNS convalidare e verificare i messaggi.
- Verifica che i SNS messaggi Amazon vengano ricevuti dal destinatario desideratoTopicArn.

Amazon SNS supporta due versioni di firma dei messaggi:

- `SignatureVersion1`: Amazon SNS crea la firma in base all'SHA1hash del messaggio.
- `SignatureVersion2`: Amazon SNS crea la firma in base all'SHA256hash del messaggio.

Per configurare la versione della firma dei messaggi sugli SNS argomenti di Amazon

Per impostazione predefinita, SNS gli argomenti di Amazon utilizzano `SignatureVersion` 1. Per scegliere l'algoritmo di hashing sul tuo SNS argomento Amazon, `SignatureVersion` 1 (SHA1) o `SignatureVersion` 2 (SHA256), puoi utilizzare l'`SetTopicAttributes`APIazione.

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 SNS messaggio Amazon quando si utilizzano richieste basate su HTTP query

1. Estrai le coppie nome-valore dal JSON documento nel corpo della HTTP POST richiesta che Amazon SNS ha inviato al tuo endpoint. Utilizzerai i valori di alcune delle coppie nome/valore per creare la stringa di firma. Quando verifichi la firma di un SNS messaggio Amazon, è fondamentale convertire i caratteri di controllo sfuggiti nelle loro rappresentazioni originali dei caratteri nei valori and. `Message` `Subject` Questi valori devono avere il formato originale quando li utilizzi come parte della stringa di firma. Per informazioni su come analizzare il documento, consultaJSON. Passaggio 1: assicurati che l'endpoint sia pronto per elaborare i 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 le versioni di firma 1 e 2. Questa sezione contiene la procedura per verificare una firma usando qieste versioni.
2. Ottieni il certificato X509 SNS utilizzato da Amazon 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 SNS messaggio Amazon. Invia il SNS messaggio Amazon, in formato canonico, allo stesso algoritmo hash utilizzato per generare la firma.
   a. Se `SignatureVersion` è 1, usa SHA1come algoritmo di hash.
   b. Se `SignatureVersion` è 2, usalo SHA256come algoritmo di hash.
8. Genera il valore hash dichiarato del messaggio AmazonSNS. Il valore hash dichiarato è il risultato dell'utilizzo del valore della chiave pubblica (dal passaggio 3) per decrittografare la firma fornita con il messaggio Amazon. SNS
9. Verifica l'autenticità e l'integrità del SNS messaggio Amazon. Confronta il valore hash derivato (fase 7) con il valore hash dichiarato (fase 8). Se i valori sono identici, al destinatario viene

garantito che il messaggio non è stato modificato durante il transito e che il messaggio deve provenire da Amazon. SNS Se i valori non sono identici, il messaggio non può essere considerato attendibile dal ricevitore.

Analisi dei formati dei SNS messaggi Amazon

Amazon SNS utilizza i seguenti formati.

Argomenti

HTTP/HTTPSintestazioni

Quando Amazon SNS invia un messaggio di conferma dell'abbonamento, notifica o annullamento dell'iscrizione aHTTP/HTTPSendpoints, invia un POST messaggio con una serie di valori di intestazione SNS specifici di Amazon. Puoi utilizzare i valori di intestazione per attività come l'identificazione del tipo di messaggio senza dover analizzare il corpo del JSON messaggio per leggere il valore. `Type` Per impostazione predefinita, Amazon SNS invia tutte le notifiche agli endpoint HTTP /S con l'`Content-Type`impostazione impostata su. `text/plain; charset=UTF-8` Per scegliere un `Content-Type` diverso da text/plain (impostazione predefinita), consultare `headerContentType` in Creazione di una politica 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 univoco universale (UUID), unico per ogni messaggio pubblicato. Per una notifica che Amazon SNS invia nuovamente durante un nuovo tentativo, viene utilizzato l'ID del messaggio originale.

`x-amz-sns-topic-arn`

L'Amazon Resource Name (ARN) per l'argomento su cui è stato pubblicato questo messaggio.

`x-amz-sns-subscription-arn`

Il ARN per l'abbonamento a questo endpoint.

L'HTTPPOSTintestazione seguente è un esempio di intestazione per un `Notification` messaggio 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
```

HTTP/formato HTTPS di conferma dell'iscrizione JSON

Dopo aver sottoscritto un HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS endpoint. 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 invia notifiche a questo endpoint fino alla conferma dell'abbonamento

Il messaggio di conferma dell'abbonamento è un POST messaggio con un corpo del messaggio che contiene un JSON documento con le seguenti coppie nome-valore.

`Type`

Il tipo di messaggio. Per confermare la sottoscrizione, il tipo è `SubscriptionConfirmation`.

`MessageId`

Un identificatore univoco universale (UUID), unico per ogni messaggio pubblicato. Per un messaggio che Amazon SNS invia nuovamente durante un nuovo tentativo, viene utilizzato l'ID del 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 è iscritto questo endpoint.

`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`

Il URL che devi visitare per confermare l'abbonamento. 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 dell'abbonamento.

`SignatureVersion`

Versione della SNS firma Amazon 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`

Al URL certificato utilizzato per firmare il messaggio.

Il HTTP POST messaggio seguente è un esempio di `SubscriptionConfirmation` messaggio a un HTTP endpoint.

```
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"
}
```

HTTP/formato HTTPS di notifica JSON

Quando Amazon SNS invia una notifica a un abbonato HTTP o a un HTTPS endpoint, il POST messaggio inviato all'endpoint ha un corpo del messaggio che contiene un JSON documento con le seguenti coppie nome-valore.

`Type`

Il tipo di messaggio. Per una notifica, il tipo è `Notification`.

`MessageId`

Un identificatore univoco universale (UUID), unico per ogni messaggio pubblicato. Per una notifica che Amazon SNS invia nuovamente durante un nuovo tentativo, viene utilizzato l'ID del messaggio originale.

`TopicArn`

L'Amazon Resource Name (ARN) per l'argomento su cui è stato pubblicato questo messaggio.

`Subject`

Il parametro `Subject` specificato quando la notifica è stata pubblicata nell'argomento.

> **Note**
>
> Si tratta di un parametro facoltativo. Se non `Subject` è stato specificato alcun valore, questa coppia nome-valore non viene visualizzata in questo JSON documento.

`Message`

Il valore `Message` specificato nel momento in cui la notifica è stata pubblicata nell'argomento.

`Timestamp`

L'ora (GMT) in cui è stata pubblicata la notifica.

`SignatureVersion`

Versione della SNS firma Amazon 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`

Al URL certificato utilizzato per firmare il messaggio.

`UnsubscribeURL`

Un URL che puoi usare per annullare l'iscrizione all'endpoint da questo argomento. Se visiti questa paginaURL, Amazon annulla l'SNSiscrizione all'endpoint e interrompe l'invio di notifiche a questo endpoint.

Il seguente HTTP POST messaggio è un esempio di `Notification` messaggio 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"
  }
}
```

HTTP/formato di conferma HTTPS dell'annullamento dell'iscrizione JSON

Dopo l'annullamento dell'iscrizione a un HTTPS endpointHTTP/da un argomento, Amazon SNS invia un messaggio di conferma dell'annullamento dell'iscrizione all'endpoint.

Il messaggio di conferma dell'annullamento dell'iscrizione è un POST messaggio con un corpo del messaggio che contiene un JSON documento con le seguenti coppie nome-valore.

`Type`

Il tipo di messaggio. Per confermare l'annullamento della sottoscrizione, il tipo è `UnsubscribeConfirmation`.

`MessageId`

Un identificatore univoco universale (UUID), unico per ogni messaggio pubblicato. Per un messaggio che Amazon SNS invia nuovamente durante un nuovo tentativo, viene utilizzato l'ID del 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 stata annullata la sottoscrizione a 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`

Il URL che devi visitare per confermare nuovamente l'abbonamento. 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 dell'iscrizione.

`SignatureVersion`

Versione della SNS firma Amazon 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`

Al URL certificato utilizzato per firmare il messaggio.

Il HTTP POST messaggio seguente è un esempio di `UnsubscribeConfirmation` messaggio a un HTTP endpoint.

```
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"
}
```

SetSubscriptionAttributesformato della politica di JSON consegna

Se si invia una richiesta all'`SetSubscriptionAttributes`azione e si imposta il `AttributeName` parametro su un valore di`DeliveryPolicy`, il valore del `AttributeValue` parametro deve essere un oggetto validoJSON. 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}}
...
```

Utilizzate il seguente JSON formato per il valore del `AttributeValue` parametro.

```
{
    "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'`SetSubscriptionAttribute`azione, consulta Amazon Simple Notification Service API Reference. SetSubscriptionAttributes Per ulteriori informazioni sulle intestazioni dei HTTP tipi di contenuto supportate, consulta. Creazione di una politica di consegna HTTP /S

SetTopicAttributes formato della politica di consegna JSON

Se si invia una richiesta all'`SetTopicAttributes`azione e si imposta il `AttributeName` parametro su un valore di`DeliveryPolicy`, il valore del `AttributeValue` parametro deve essere un oggetto validoJSON. 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}}}
...
```

Utilizzate il seguente JSON formato per il valore del `AttributeValue` parametro.

```
{
    "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 sull'`SetTopicAttribute`azione, consulta Amazon Simple Notification Service API Reference. SetTopicAttributes Per ulteriori informazioni sulle intestazioni dei HTTP tipi di contenuto supportate, consulta. Creazione di una politica di consegna HTTP /S

Fanout SNS degli eventi Amazon su AWS Event Fork Pipelines

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 SNS argomenti, 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'utilizzo 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.

Puoi usare Amazon SNS per creare applicazioni basate sugli eventi che utilizzano i servizi per abbonati per eseguire il lavoro automaticamente in risposta a eventi attivati dai servizi di pubblicazione. 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 tue applicazioni basate sugli eventi, puoi sottoscrivere pipeline di gestione degli eventi, basate su Event Fork Pipelines, agli argomenti di Amazon. AWS SNS AWS Event Fork Pipelines è una suite di applicazioni annidate open source, basata sul AWS Serverless Application Model (AWS SAM), che puoi distribuire direttamente dalla suite AWS Event Fork Pipelines (scegli Mostra app che creano ruoli o politiche di risorse personalizzati) nel tuo account. IAM AWS

Per un AWS caso d'uso di Event Fork Pipelines, vedi. Distribuzione e test dell'applicazione di esempio Amazon SNS Event Fork Pipeline

Argomenti

- Come funziona AWS Event Fork Pipelines
- Implementazione di Event Fork Pipelines AWS
- Distribuzione e test dell'applicazione di esempio Amazon SNS Event Fork Pipeline

- Iscrizione a AWS Event Fork Pipelines a un argomento Amazon SNS

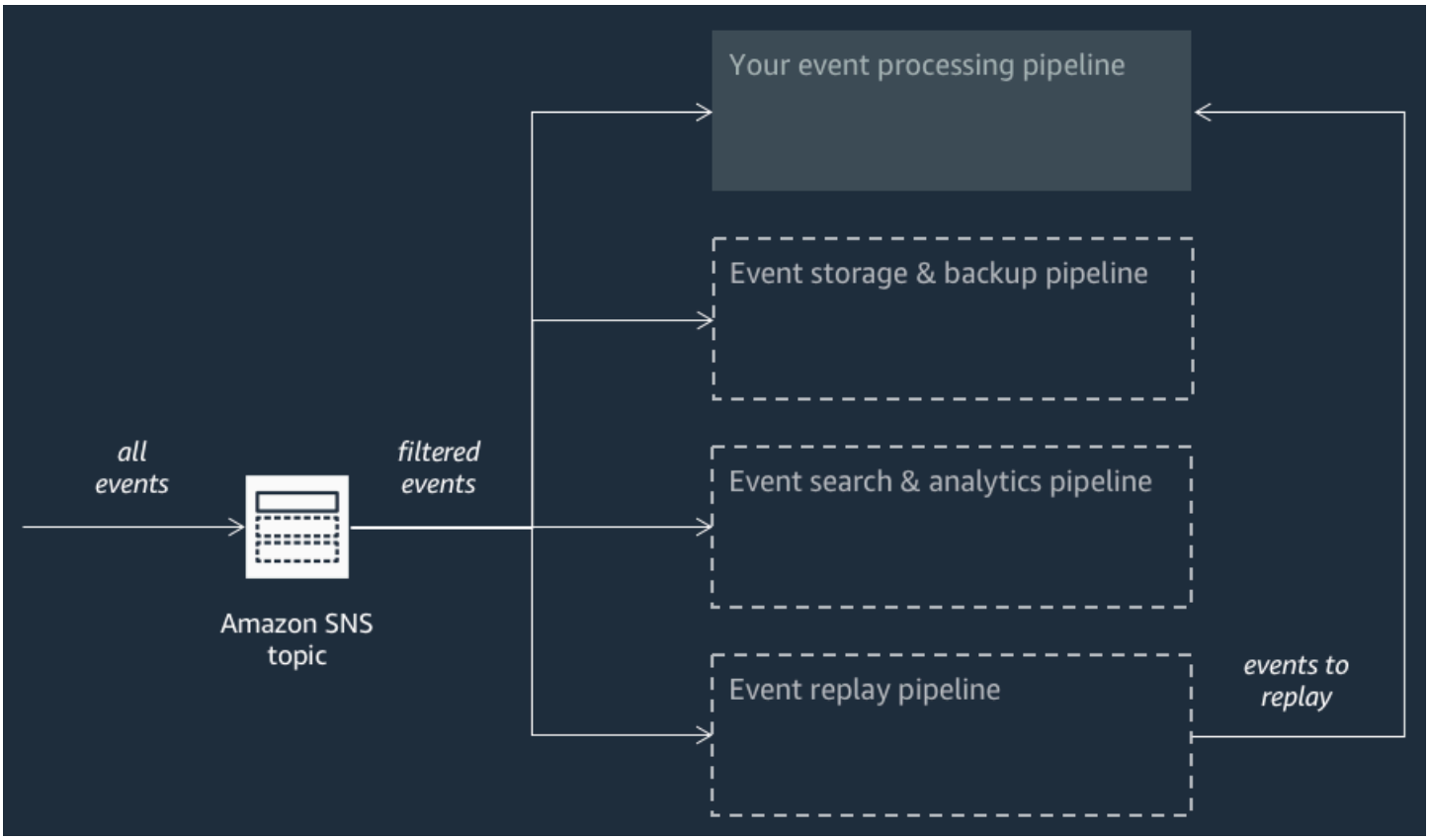
Come funziona AWS Event Fork Pipelines

AWS Event Fork Pipelines è un modello di progettazione senza server. Tuttavia, è anche una suite di applicazioni serverless annidate basate su AWS SAM (che potete distribuire direttamente da AWS Serverless Application Repository () alle vostre per arricchire le vostre piattaforme basate sugli Account AWS eventi AWS SAR). È possibile distribuire queste applicazioni nidificate individualmente, come richiesto dall'architettura.

Argomenti

- La pipeline di archiviazione di eventi e di backup
- La pipeline di ricerca di eventi e di analisi dei dati
- Pipeline di riproduzione eventi

Il diagramma seguente mostra un'applicazione AWS Event Fork Pipelines integrata da tre applicazioni annidate. È possibile distribuire qualsiasi pipeline della suite AWS Event Fork Pipelines in modo indipendente, come richiesto dall'architettura. AWS SAR



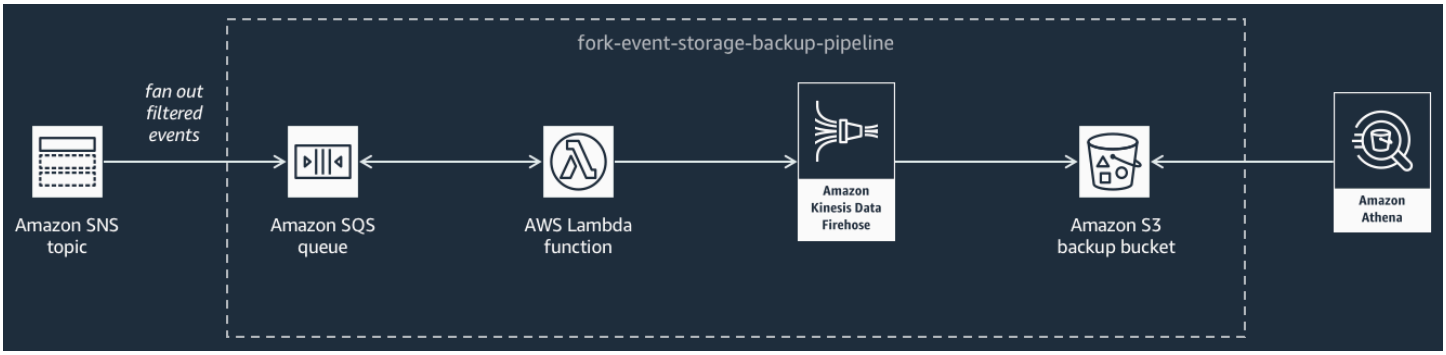
Ogni pipeline è sottoscritta allo stesso SNS argomento Amazon, permettendo a se stessa di elaborare gli eventi in parallelo man mano che questi eventi vengono pubblicati sull'argomento. Ogni pipeline è indipendente ed è in grado di impostare la propria Policy di filtro per le sottoscrizioni. In questo modo una pipeline può elaborare solo un sottoinsieme di eventi a cui è interessata (invece che tutti gli eventi pubblicati nell'argomento).

> **Note**
>
> Poiché colloca le tre AWS Event Fork Pipeline insieme alle normali pipeline di elaborazione degli eventi (probabilmente già abbonata al tuo SNS argomento Amazon), non è necessario modificare alcuna parte del tuo attuale editor di messaggi per sfruttare AWS Event Fork Pipelines nei tuoi carichi 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 abbonare questa pipeline al tuo SNS argomento Amazon per eseguire automaticamente il backup degli eventi che fluiscono nel tuo sistema.

Questa pipeline comprende una SQS coda Amazon che memorizza nel buffer gli eventi forniti dall'SNSargomento Amazon, una AWS Lambda 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.

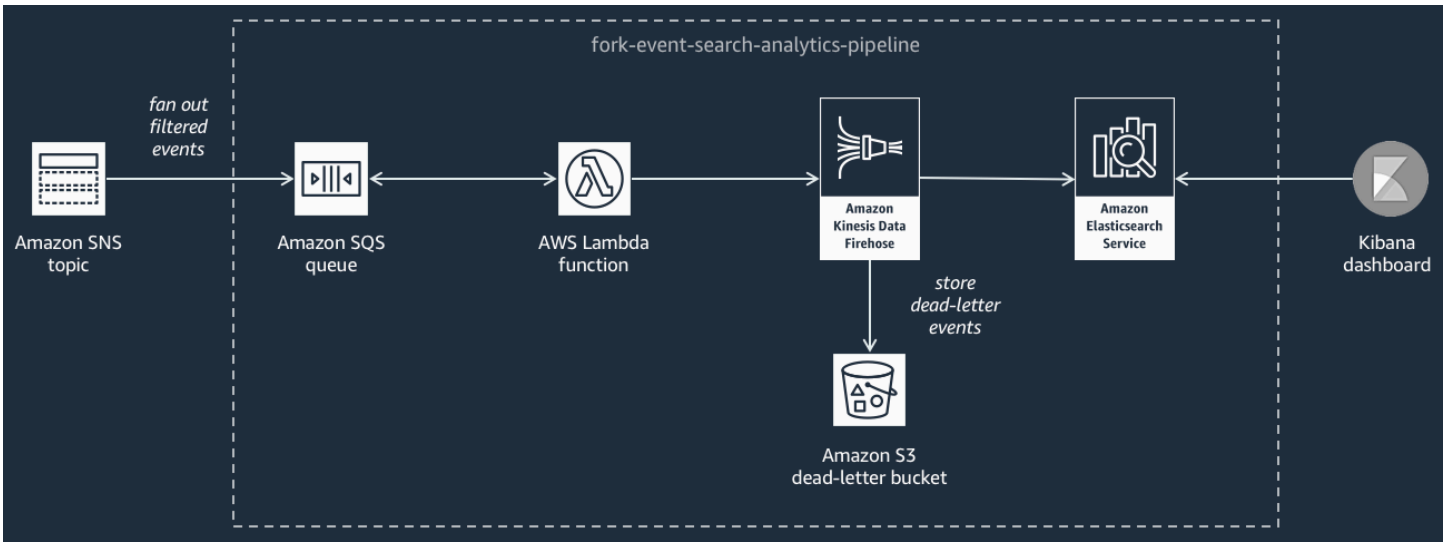


Per affinare il comportamento del flusso Firehose, puoi configurarlo in modo che trasformi, comprima ed esegua il buffering degli eventi prima di caricarli nel bucket. Man mano che gli eventi vengono caricati, puoi utilizzare Amazon Athena per interrogare il bucket utilizzando query standard. SQL Puoi 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 iscrivere questa pipeline al tuo SNS argomento Amazon per indicizzare gli eventi che fluiscono nel tuo sistema in un dominio di ricerca e quindi eseguire analisi su di essi.

Questa pipeline è composta da una SQS coda Amazon che memorizza nel buffer gli eventi forniti dall'SNSargomento Amazon, una AWS Lambda 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 possono essere indicizzati nel dominio di ricerca.



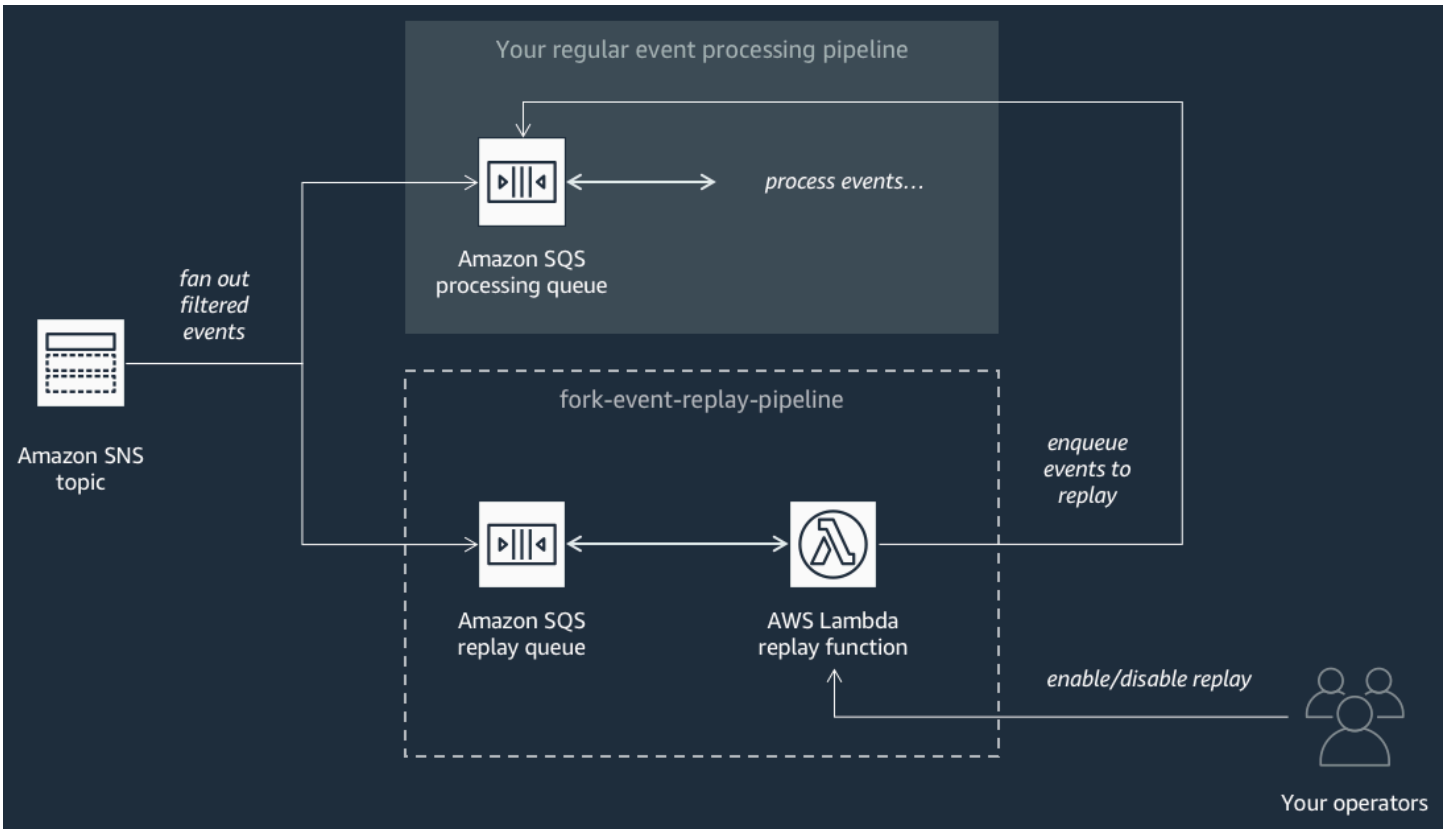
Per affinare il flusso Firehose in termini di buffering, trasformazione e compressione degli eventi, puoi configurare questa pipeline.

Puoi anche configurare se la pipeline deve riutilizzare un OpenSearch dominio esistente nel tuo dominio 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 che sono stati elaborati dal sistema negli ultimi 14 giorni (ad esempio quando la piattaforma deve essere ripristinata in seguito a un guasto), puoi iscrivere questa pipeline al tuo SNS argomento Amazon e quindi rielaborare gli eventi.

Questa pipeline è composta da una SQS coda Amazon che memorizza nel buffer gli eventi forniti dall'SNSargomento Amazon e da una AWS Lambda funzione che analizza gli eventi dalla coda e li reindirizza nella normale pipeline di elaborazione degli eventi, anch'essa sottoscritta al tuo 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 SQS replay di Amazon come fonte di eventi per la AWS Lambda funzione di replay.

Implementazione di Event Fork Pipelines AWS

La suite AWS Event Fork Pipelines (scegli Mostra app che creano IAM ruoli o politiche di risorse personalizzati) è disponibile come gruppo di applicazioni pubbliche in AWS Serverless Application Repository, da dove puoi distribuirle e testarle manualmente utilizzando la console.AWS Lambda Per informazioni sulla distribuzione delle pipeline tramite la console, consulta. AWS Lambda Iscrizione a AWS Event Fork Pipelines a un argomento Amazon SNS

In uno scenario di produzione, consigliamo di incorporare AWS Event Fork Pipelines nel modello generale dell'applicazione. AWS SAM La funzionalità di applicazione annidata consente di eseguire questa operazione aggiungendo la risorsa al AWS SAM modello, facendo riferimento `AWS::Serverless::Application` alla AWS SAR `ApplicationId` e all'applicazione nidificata. `SemanticVersion`

Ad esempio, è possibile utilizzare Event Storage and Backup Pipeline come applicazione annidata aggiungendo il seguente YAML frammento alla `Resources` sezione 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 specificate i valori dei parametri, potete utilizzare le funzioni AWS CloudFormation intrinseche per fare riferimento ad altre risorse del modello. Ad esempio, nel YAML frammento precedente, il `TopicArn` parametro fa riferimento alla `AWS::SNS::Topic` risorsa`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 AWS Lambda console AWS SAR dell'applicazione include il pulsante Copia come SAM risorsa, che copia negli appunti quanto YAML necessario per annidare un' AWS SARapplicazione.

Distribuzione e test dell'applicazione di esempio Amazon SNS Event Fork Pipeline

Per accelerare lo sviluppo delle tue applicazioni basate sugli eventi, puoi sottoscrivere pipeline di gestione degli eventi, basate su Event Fork Pipelines, agli argomenti di Amazon. AWS SNS

AWS Event Fork Pipelines è una suite di applicazioni annidate open source, basata sul AWS Serverless Application Model (AWS SAM), che puoi distribuire direttamente dalla suite AWS Event Fork Pipelines (scegli Mostra app che creano ruoli o politiche di risorse personalizzati) nel tuo account. IAM AWS Per ulteriori informazioni, consulta Come funziona AWS Event Fork Pipelines.

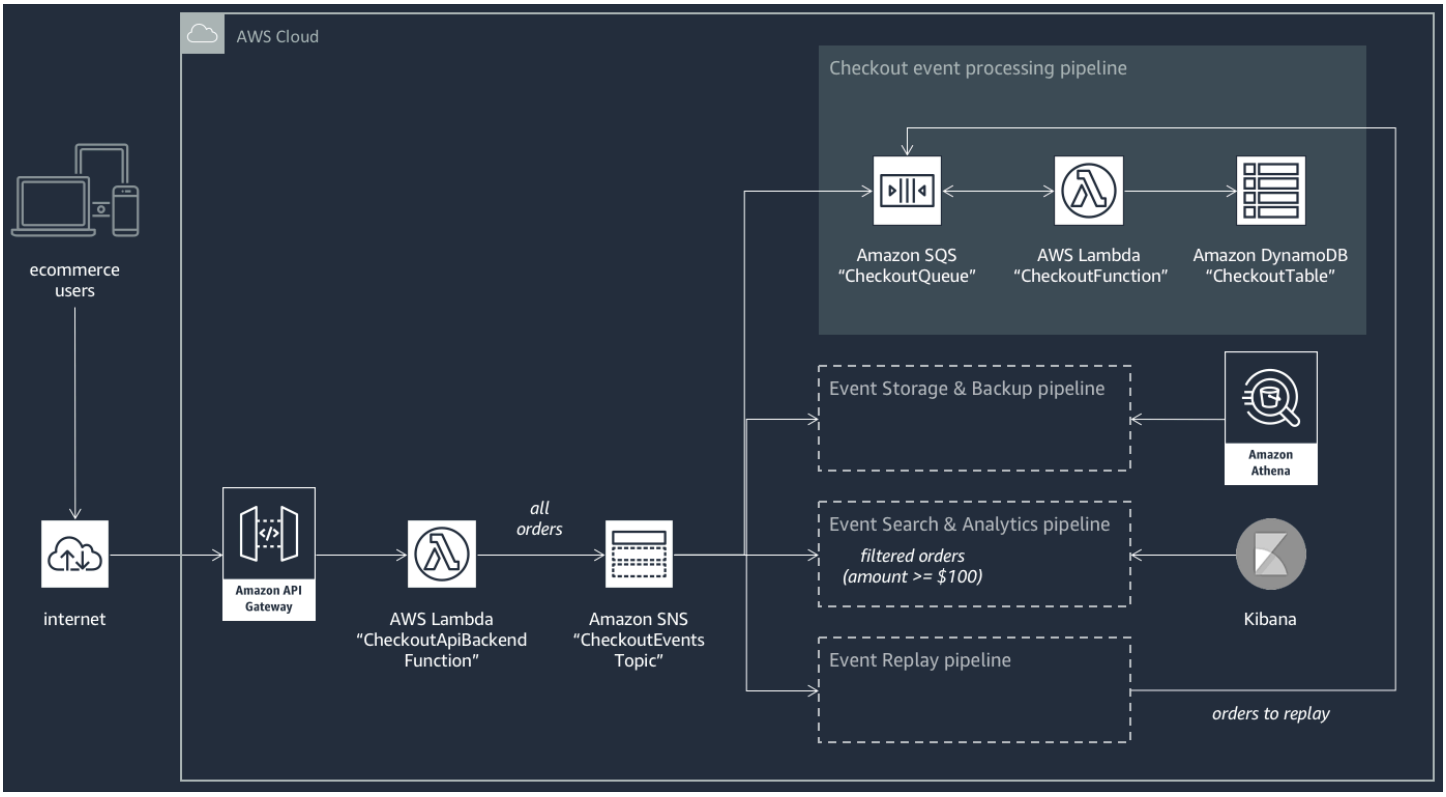
Questa pagina mostra come utilizzare l'applicazione di esempio Event Fork Pipelines AWS Management Console per distribuire e testare l'applicazione di esempio Event Fork Pipelines. AWS

> **Important**
>
> Per evitare di incorrere in costi indesiderati dopo aver completato la distribuzione dell'applicazione di esempio AWS Event Fork Pipelines, eliminate il relativo stack. AWS CloudFormation Per ulteriori informazioni, vedi Eliminazione di uno stack sulla console AWS CloudFormation nella Guida per l'utente di AWS CloudFormation .

Argomenti

AWS Esempio di caso d'uso di Event Fork Pipelines

Lo scenario seguente descrive un'applicazione di e-commerce senza server e basata sugli eventi che utilizza AWS Event Fork Pipelines. 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 riceve gli ordini dagli acquirenti tramite un servizio ospitato da Gateway e supportato dalla funzione. RESTful API API AWS Lambda `CheckoutApiBackendFunction` Questa funzione pubblica tutti gli ordini ricevuti su un SNS argomento Amazon denominato `CheckoutEventsTopic` che, a sua volta, distribuisce gli ordini a quattro diverse pipeline.

La prima è la normale pipeline di elaborazione del checkout progettata e implementata dal proprietario dell'applicazione di E-Commerce. Questa pipeline include la SQS coda Amazon `CheckoutQueue` che memorizza nel buffer tutti gli ordini ricevuti, una AWS Lambda funzione denominata `CheckoutFunction` che esegue il polling della coda per elaborare questi ordini e la tabella DynamoDB che salva in modo sicuro tutti gli ordini effettuati. `CheckoutTable`

Applicazione di Event Fork AWS Pipelines

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 sottoscrivere AWS Event Fork Pipelines all'argomento Amazon `CheckoutEventsTopic` SNS

- La pipeline di archiviazione di eventi e di backupè configurato per trasformare i dati per rimuovere i dati della carta di credito, memorizzarli nel buffer per 60 secondi, comprimerli e crittografarli utilizzando la chiave gestita dal cliente predefinita per Amazon S3. GZIP Questa chiave è gestita AWS e alimentata da (). 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 eventiè configurato con la parte Amazon SQS queue della normale pipeline di elaborazione degli ordini progettata e implementata dal proprietario dell'applicazione di e-commerce.

  Per ulteriori informazioni, consulta Queue Name e URL nella Amazon Simple Queue Service Developer Guide.

La seguente politica di JSON filtro è impostata nella configurazione per Event Search and Analytics Pipeline. che trova solo ordini in entrata il cui importo totale sia di 100 dollari o più. Per ulteriori informazioni, consulta Filtraggio SNS dei messaggi Amazon.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

Utilizzando il pattern AWS Event Fork Pipelines, il proprietario dell'applicazione di e-commerce può evitare il sovraccarico di sviluppo che spesso deriva dalla logica di codifica indifferenziata per la gestione degli eventi. Invece, può implementare AWS Event Fork Pipelines direttamente dal suo interno. AWS Serverless Application Repository Account AWS

Fase 1: distribuzione dell'applicazione Amazon SNS 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. Scegli Sfoglia l'archivio di app serverless, Applicazioni pubbliche, Mostra app che creano IAM ruoli o politiche 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 riutilizzi il nome di un'applicazione, la distribuzione aggiornerà solo lo AWS CloudFormation stack precedentemente distribuito (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. Scegli Riconosco che questa app crea IAM ruoli personalizzati, politiche delle risorse e distribuisce applicazioni annidate. quindi, nella parte inferiore della pagina, scegli Deploy.

Sullo stato di distribuzione per fork-example-ecommerce -***my-app***pagina, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, AWS CloudFormation inizia a creare lo stack e visualizza lo stato CREATE_IN_ PROGRESS per ogni risorsa. Quando il processo è completo, AWS CloudFormation visualizza 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 SNS di esempio collegata

1. Nella AWS Lambda console, nel pannello di navigazione, scegli 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. Copia il link Prod API endpoint e aggiungilo, `/checkout` ad esempio:

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. Copia quanto segue in un file JSON denominato. `test_event.json`

   ```
   {
      "id": 15311,
      "date": "2019-03-25T23:41:11-08:00",
      "status": "confirmed",
      "customer": {
         "id": 65144,
     "quantity": 2,
         "price": 25.00,
         "subtotal": 50.00
      }]
   ```

```
}
```

5. Per inviare una HTTPS richiesta al tuo API endpoint, trasmetti il payload dell'evento di esempio come input eseguendo un `curl` comando, ad esempio:

   ```
   curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
   east-2.amazonaws.com/Prod/checkout
   ```

   APIrestituisce la seguente risposta vuota, che indica un'esecuzione riuscita:

   ```
   { }
   ```

Fase 3: Verifica delle prestazioni SNS dell'applicazione e della pipeline Amazon

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 archiviazione e backup degli eventi

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 JSON payload memorizzato non contenga informazioni sulla carta di credito, decomprimi il file.

Fase 3: Verifica dell'esecuzione della pipeline di ricerca e analisi degli eventi

1. Accedi alla console OpenSearch di servizio.
2. Nel riquadro di navigazione, in My domains (I miei domini), scegliere il dominio con prefisso `serverl-analyt`.
3. La pipeline è configurata con una politica di filtro degli SNS abbonamenti Amazon 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 USD $100, sul serverl-analyt- ***abcdefgh1ijk***pagina, scegli Indices, checkout_events.

Fase 4: Verifica dell'esecuzione della pipeline di riproduzione degli eventi

1. Accedi alla SQSconsole Amazon.
2. Nell'elenco delle code, cercare `serverlessrepo-fork-example` e scegliere `ReplayQueue`.
3. Scegli Invia e ricevi messaggi.
4. Nella sezione Invia e ricevi messaggi in fork-example-ecommerce -***my-app***... ReplayP- - ReplayQueue***123ABCD4E5F6***finestra di dialogo, 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 degli eventi per il ripristino

Fase 1: Attivare il problema simulato e inviare una seconda API richiesta

1. Accedi alla console AWS Lambda.
2. Nel riquadro di navigazione, scegliere Functions (Funzioni).
3. Cercare `serverlessrepo-fork-example` e selezionare `CheckoutFunction`.
4. Sul fork-example-ecommerce -***my-app***-CheckoutFunction-***ABCDEF***... pagina, nella sezione Variabili d'ambiente, imposta la ENABLED variabile BUG_ su true e poi scegli Salva.
5. Copia quanto segue JSON in un file denominato`test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
```

```
    "customer": {
       "id": 56999,
"quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Per inviare una HTTPS richiesta al tuo API endpoint, trasmetti il payload dell'evento di esempio come input eseguendo un `curl` comando, ad esempio:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

APIrestituisce la seguente risposta vuota, che indica un'esecuzione riuscita:

```
{ }
```

Fase 2: Verificare il 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 memorizzati, alcuni contrassegnati come! CORRUPTED

Fase 3: Disattiva il 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. Sul - fork-example-ecommerce***my-app***-CheckoutFunction-***ABCDEF***... pagina, nella sezione Variabili d'ambiente, imposta la ENABLED variabile BUG_ su false e poi scegli Salva.

Passaggio 4: abilita il replay per risolvere il problema

1. Nella AWS Lambda console, nel pannello di navigazione, scegli Funzioni.
2. Cercare `serverlessrepo-fork-example` e selezionare `ReplayFunction`.
3. Espandi la sezione Designer, scegli il SQSriquadro e quindi, nella SQSsezione, scegli Abilitato.

> **Note**
>
> L'attivazione del trigger Amazon SQS Event Source richiede circa 1 minuto.

4. Seleziona Salva.
5. Per visualizzare gli attributi ripristinati, tornare alla console Amazon DynamoDB.
6. Per disabilitare la riproduzione, torna alla AWS Lambda console e disabilita il trigger della fonte di SQS eventi di Amazon per`ReplayFunction`.

Iscrizione a AWS Event Fork Pipelines a un argomento Amazon SNS

Per accelerare lo sviluppo delle tue applicazioni basate sugli eventi, puoi sottoscrivere pipeline di gestione degli eventi, basate su Event Fork Pipelines, agli argomenti di Amazon. AWS SNS AWS Event Fork Pipelines è una suite di applicazioni annidate open source, basata sul AWS Serverless Application Model (AWS SAM), che puoi distribuire direttamente dalla suite AWS Event Fork Pipelines (scegli Mostra app che creano ruoli o politiche di risorse personalizzati) nel tuo account. IAM AWS Per ulteriori informazioni, consulta Come funziona AWS Event Fork Pipelines.

Questa sezione mostra come utilizzare il AWS Management Console per distribuire una pipeline e quindi sottoscrivere AWS Event Fork Pipelines a un argomento Amazon. SNS Prima di iniziare, crea un SNS argomento Amazon.

Per eliminare le risorse che compongono una pipeline, individua la pipeline nella pagina Applicazioni della AWS Lambda console, espandi la sezione dei SAM modelli, scegli CloudFormationstack, quindi scegli Altre azioni, Elimina stack.

Argomenti

- Implementazione e sottoscrizione della Event Storage and Backup Pipeline su Amazon SNS
- Implementazione e sottoscrizione della Event Search and Analytics Pipeline su Amazon SNS
- Implementazione della Event Replay Pipeline con l'integrazione di Amazon SNS

Implementazione e sottoscrizione della Event Storage and Backup Pipeline su Amazon SNS

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 SNS argomenti, 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'utilizzo 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.

Questa pagina mostra come distribuire la Event Storage and Backup Pipeline e iscriverla a un argomento di AmazonSNS. Questo processo trasforma automaticamente il AWS SAM modello associato alla pipeline in uno AWS CloudFormation stack, quindi distribuisce lo stack 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:

- SQSCoda Amazon
- 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 Amazon Data *`S3DestinationConfiguration`* Firehose Reference. API

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 Politiche di filtro degli SNS abbonamenti Amazon 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. Scegli Sfoglia l'archivio di app serverless, Applicazioni pubbliche, Mostra app che creano ruoli o politiche di risorse personalizzati. IAM

   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 riutilizzi il nome di un'applicazione, la distribuzione aggiornerà solo lo AWS CloudFormation stack precedentemente distribuito (anziché crearne uno nuovo).

   b. (Facoltativo) Per BucketArn, inserisci il ARN bucket S3 in cui vengono caricati gli eventi in entrata. Se non inserisci un valore, viene creato un nuovo bucket S3 nel tuo account. AWS

   c. (Facoltativo) Per DataTransformationFunctionArn, inserisci la ARN 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 TopicArn, inserisci ARN l'SNSargomento Amazon a cui deve essere sottoscritta questa istanza della pipeline fork.

   f. (Facoltativo) Per StreamBufferingIntervalInSecondse StreamBufferingSizeInMBs, inserisci i valori per configurare il buffering degli eventi in entrata. 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 SNS abbonamenti Amazon, nel JSON formato, 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. Scegli Riconosco che questa app crea IAM ruoli personalizzati, politiche delle risorse e distribuisce applicazioni annidate. quindi scegli Deploy.

Nello stato di distribuzione per ***`my-app`***pagina, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, AWS CloudFormation inizia a creare lo stack e visualizza lo stato CREATE_IN_ PROGRESS per ogni risorsa. Quando il processo è completo, AWS CloudFormation visualizza 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 SNS argomento Amazon vengono archiviati automaticamente nel bucket di backup S3 fornito automaticamente dalla pipeline Event Storage and Backup.

Implementazione e sottoscrizione della Event Search and Analytics Pipeline su Amazon SNS

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 SNS argomenti, 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'utilizzo 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.

Questa pagina mostra come distribuire Event Search and Analytics Pipeline e iscriverla a un argomento di AmazonSNS. Questo processo trasforma automaticamente il AWS SAM modello associato alla pipeline in uno AWS CloudFormation stack, quindi distribuisce lo stack 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:

- SQSCoda Amazon
- 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 API Firehose 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 Politiche di filtro degli SNS abbonamenti Amazon 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. Scegli Sfoglia l'archivio di app serverless, Applicazioni pubbliche, Mostra app che creano ruoli o politiche di risorse personalizzati. IAM
   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 riutilizzi il nome di un'applicazione, la distribuzione aggiornerà solo lo AWS CloudFormation stack precedentemente distribuito (anziché crearne uno nuovo).

b. (Facoltativo) Per DataTransformationFunctionArn, inserisci ARN la 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 il dominio ARN 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 TopicArn, inserisci ARN l'SNSargomento Amazon 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 (tranne 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 StreamBufferingSizeInMBs, 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 SNS abbonamenti Amazon, nel JSON formato, 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. Scegli Riconosco che questa app crea IAM ruoli personalizzati, politiche delle risorse e distribuisce applicazioni annidate. quindi scegli Deploy.

Nello stato di distribuzione per ***`my-app-search`***pagina, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, AWS CloudFormation inizia a creare lo stack e visualizza lo stato CREATE_IN_ PROGRESS per ogni risorsa. Quando il processo è completo, AWS CloudFormation visualizza 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 SNS argomento Amazon 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.

Implementazione della Event Replay Pipeline con l'integrazione di Amazon SNS

Questa pagina mostra come distribuire Event Replay Pipeline e iscriverla a un argomento di Amazon. SNS Questo processo trasforma automaticamente il AWS SAM modello associato alla pipeline in uno AWS CloudFormation stack, quindi distribuisce lo stack nel tuo. Account AWS Questo processo crea e configura anche l'insieme di risorse che compongono la Event Replay Pipeline, tra cui una SQS coda Amazon e una funzione Lambda.

Per ulteriori informazioni su come filtrare gli eventi, consulta Politiche di filtro degli SNS abbonamenti Amazon 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. Scegli Browse serverless app repository, Public applications, Show apps che creano ruoli o policy di risorse personalizzati. IAM
   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 riutilizzate il nome di un'applicazione, la distribuzione aggiornerà solo lo AWS CloudFormation stack precedentemente distribuito (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 SQS replay di Amazon conserva il messaggio. In assenza di valori immessi, viene utilizzato un valore di 1.209.600 secondi (14 giorni).
   d. Per TopicArn, inserisci ARN l'SNSargomento Amazon a cui deve essere sottoscritta questa istanza della pipeline fork.
   e. Per DestinationQueueName, inserisci il nome della SQS coda Amazon a cui la funzione Lambda Replay inoltra i messaggi.

f. (Facoltativo) Per SubscriptionFilterPolicy, inserisci la politica di filtro degli SNS abbonamenti Amazon, nel JSON formato, 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. Scegli Riconosco che questa app crea IAM ruoli personalizzati, politiche di risorse e distribuisce applicazioni annidate. quindi scegli Deploy.

Nello stato di distribuzione per ***my-app-replay***pagina, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, AWS CloudFormation inizia a creare lo stack e visualizza lo stato CREATE_IN_ PROGRESS per ogni risorsa. Quando il processo è completo, AWS CloudFormation visualizza 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 SNS argomento Amazon vengono inseriti nel buffer per essere riprodotti nella SQS coda Amazon fornita automaticamente dalla Event Replay Pipeline.

> **Note**
>
> La riproduzione è disabilitata per impostazione predefinita. Per abilitare la riproduzione, vai alla pagina della funzione sulla console Lambda, espandi la sezione Designer, scegli SQSil riquadro e quindi, SQSnella sezione, scegli Enabled.

Utilizzo di Amazon EventBridge Scheduler con Amazon SNS

Amazon EventBridge Scheduler è uno strumento di pianificazione senza server che consente di creare, eseguire e gestire attività da un unico servizio gestito centralizzato. Con EventBridge Scheduler, puoi creare pianificazioni utilizzando Cron e rate expression per schemi ricorrenti o configurare chiamate una tantum. È possibile impostare finestre temporali flessibili per la consegna, definire limiti di nuovi tentativi e impostare il tempo massimo di conservazione per le chiamate non riuscite. API

Questa pagina spiega come utilizzare EventBridge Scheduler per pubblicare un messaggio da un SNS argomento di Amazon in base a una pianificazione.

Argomenti

Impostazione del ruolo di esecuzione

Quando crei una nuova EventBridge pianificazione, Scheduler deve essere autorizzato a richiamare l'APIoperazione di destinazione per tuo conto. Concedi queste autorizzazioni a EventBridge Scheduler utilizzando un ruolo di esecuzione. La policy di autorizzazione collegata al ruolo di esecuzione della pianificazione definisce le autorizzazioni necessarie. Queste autorizzazioni dipendono dall'obiettivo che EventBridge Scheduler API deve richiamare.

Quando si utilizza la console EventBridge Scheduler per creare una pianificazione, come nella procedura seguente, EventBridge Scheduler imposta automaticamente un ruolo di esecuzione in base all'obiettivo selezionato. Se si desidera creare una pianificazione utilizzando uno degli EventBridge Scheduler SDKs AWS CLI AWS CloudFormation, oppure è necessario disporre di un ruolo di esecuzione esistente che conceda le autorizzazioni richieste da EventBridge Scheduler per richiamare una destinazione. Per ulteriori informazioni sull'impostazione manuale di un ruolo di esecuzione per la pianificazione, consulta Impostazione di un ruolo di esecuzione nella Guida per l'utente di Scheduler. EventBridge

Creare una pianificazione.

Per creare una pianificazione utilizzando la console

1. Apri la console Amazon EventBridge Scheduler a https://console.aws.amazon.com/scheduler/casa.
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 immetti 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 Schedule types on EventBridge Scheduler nella Amazon EventBridge Scheduler User Guide. 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. Scegli Next (Successivo).
7. Nella pagina Seleziona destinazione, scegli l' AWS APIoperazione richiamata da Scheduler: EventBridge

   a. Scegli Amazon SNS Publish.

   b. Nella sezione Pubblica, seleziona un SNS argomento o scegli Crea nuovo SNS argomento.

   c. (Facoltativo) Inserisci un JSON carico utile. Se non si inserisce un payload, EventBridge Scheduler utilizza un evento vuoto per richiamare la funzione.

8. Scegli Next (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 politica di riprova per la tua pianificazione, in Politica di riprova e dead-letter queue (), procedi come segue: DLQ

      - Attiva/disattiva Riprova.
      - Per Età massima dell'evento, inserisci il numero massimo di ore e minuti in cui EventBridge Scheduler deve conservare un evento non elaborato.
      - La durata massima è 24 ore.
      - Per Numero massimo di tentativi, inserisci il numero massimo di volte in cui EventBridge Scheduler riprova la pianificazione se la destinazione restituisce un errore.

        Il valore massimo è 185 tentativi.

      Con le politiche di ripetizione dei tentativi, se una pianificazione non riesce a richiamare l'obiettivo, EventBridge Scheduler esegue nuovamente la pianificazione. Se configurato, è necessario impostare il tempo di conservazione massimo e i nuovi tentativi per la pianificazione.

   c. Scegli dove EventBridge Scheduler archivia gli eventi non consegnati.

Opzione Dead-letter queue () DLQ	Esegui questa operazione…	
Non conservare	Scegliere None (Nessuno).	
Memorizza l'evento nello stesso spazio in Account AWS cui stai creando il programma	a. Scegli Seleziona una SQS coda Amazon nel mio Account AWS account. DLQ b. Scegli l'Amazon Resource Name (ARN) della SQS coda Amazon.	
Archivia l'evento in un luogo diverso Account AWS da quello in cui stai creando il programma	a. Scegli Specificare una SQS coda Amazon in altro Account AWS formato. DLQ b. Inserisci l'Amazon Resource Name (ARN) della SQS coda Amazon.	

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 una KMS chiave esistente ARN o scegli Crea un codice AWS KMS key per accedere alla AWS KMS console. Per ulteriori informazioni su come EventBridge Scheduler crittografa i dati inattivi, consulta Encryption at rest nella Amazon EventBridge Scheduler User Guide.

e. Per fare in modo che EventBridge Scheduler crei un nuovo ruolo di esecuzione per te, scegli Crea nuovo ruolo per questa pianificazione. Inserisci, quindi, un nome per Nome ruolo. Se scegli questa opzione, EventBridge Scheduler assegna al ruolo le autorizzazioni necessarie per la destinazione basata sul modello.

10. Scegli Next (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.

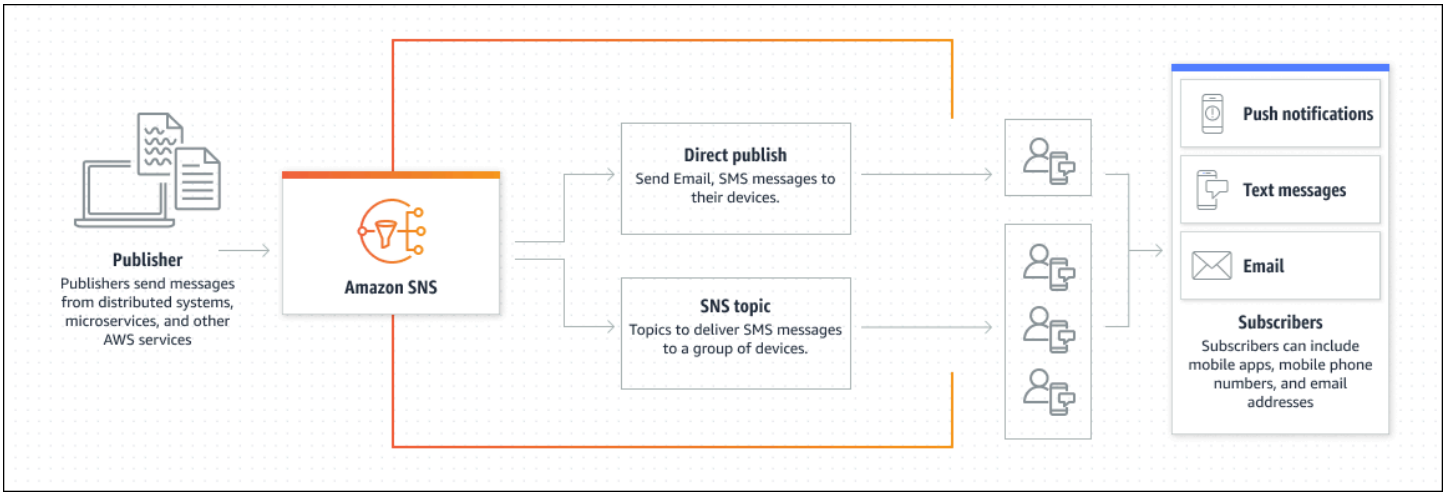
Risorse correlate

Per ulteriori informazioni su EventBridge Scheduler, consulta quanto segue:

- EventBridge Guida per l'utente di Scheduler
- EventBridge Riferimento allo Scheduler API
- EventBridge Prezzi Scheduler

Utilizzo di Amazon SNS per la application-to-person messaggistica

La messaggistica Amazon SNS application-to-person (A2P) ti consente di inviare notifiche e avvisi direttamente ai dispositivi mobili dei tuoi clienti tramite SMS (Short Message Service). Utilizzando questa funzione, puoi inviare notifiche push alle app mobili, messaggi di testo ai numeri di cellulare ed e-mail in testo semplice agli indirizzi e-mail. Inoltre, hai la flessibilità di distribuire messaggi utilizzando argomenti per raggiungere più destinatari o di pubblicare messaggi direttamente su singoli endpoint mobili per comunicazioni personalizzate.



I seguenti argomenti spiegano come utilizzare Amazon SNS per le notifiche utente con gli abbonati, ad esempio applicazioni mobili, numeri di cellulare e indirizzi e-mail:

Argomenti

- Messaggi di testo mobili con Amazon SNS
- Invio di notifiche push per dispositivi mobili con Amazon SNS
- Configurazione e gestione dell'abbonamento SNS e-mail Amazon

Messaggi di testo mobili con Amazon SNS

Important

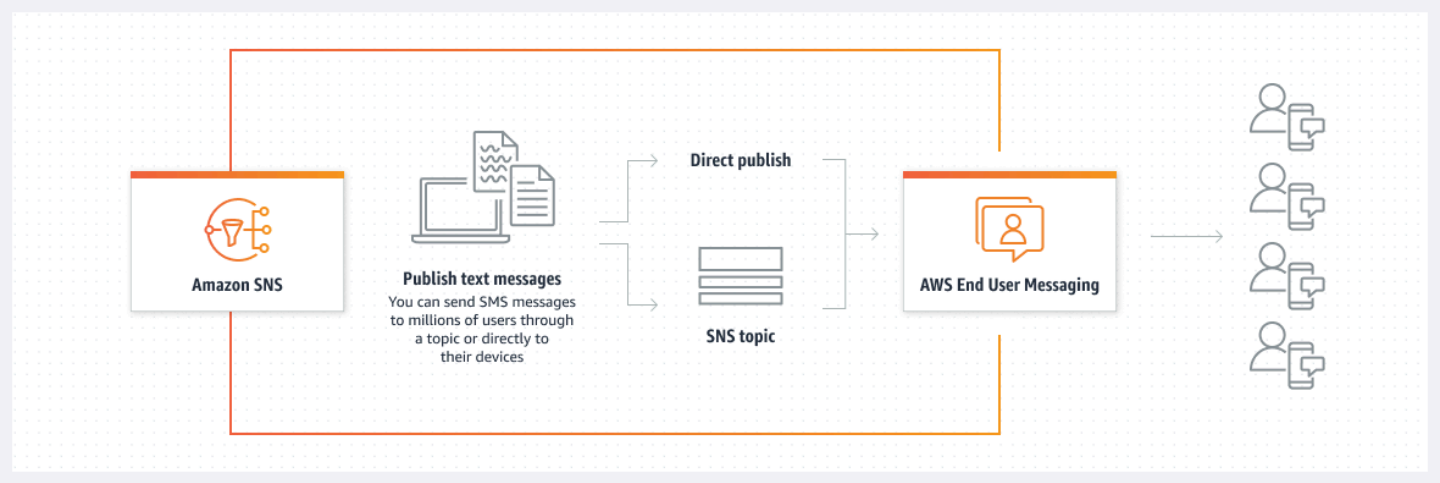
L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida

contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

La messaggistica di testo SNS mobile di Amazon (SMS) è progettata per facilitare la consegna dei messaggi su varie piattaforme, come applicazioni Web, mobili e aziendali che supportanoSMS. Gli utenti possono inviare messaggi a uno o più numeri di telefono iscrivendoli a un argomento, semplificando il processo di distribuzione.

SNSI messaggi Amazon vengono recapitati da AWS End User Messaging SMS, il che garantisce una trasmissione affidabile dei messaggi. All'interno di Amazon SNSAPIs, puoi impostare varie proprietà come i tipi di messaggi (promozionali o transazionali), i limiti di spesa mensili, gli elenchi di opt-out e l'ottimizzazione della consegna dei messaggi.

AWS End User Messaging SMS gestisce la trasmissione di messaggi al numero di telefono di destinazione attraverso la sua rete di fornitura globaleSMS. Gestisce il routing, lo stato di consegna e qualsiasi richiesta di conformità alle normative regionali. Per accedere a SMS funzionalità aggiuntive come autorizzazioni granulari, pool di telefoni, set di configurazioni, SMS simulatore e regole nazionali, consulta la Guida per l'utente.AWS End User Messaging SMS



Le seguenti funzionalità chiave ti aiutano a inviare SNS SMS messaggi Amazon scalabili e facilmente estensibili:

Personalizza le preferenze dei messaggi

Personalizza SMS le consegne per te Account AWS impostando SMS le preferenze in base al budget e al caso d'uso. Ad esempio, puoi scegliere se dare priorità ai tuoi messaggi all'efficienza dei costi o alla consegna affidabile.

Imposta le quote di spesa

Personalizza le tue SMS consegne specificando le quote di spesa o per il recapito di singoli messaggi e le quote di spesa mensili per le tue. Account AWS Laddove richiesto dalle leggi e dai regolamenti locali (come Stati Uniti e Canada), SMS i destinatari possono rinunciare, il che significa che scelgono di non ricevere più messaggi dall'utente. SMS Account AWS Dopo che un destinatario ha scelto di non ricevere messaggi, puoi, con limitazioni, inserire nuovamente il numero di telefono in modo da poter riprendere a inviare messaggi.

Invia messaggi a livello globale SMS

Amazon SNS supporta la SMS messaggistica in più regioni, consentendoti di inviare messaggi in oltre 240 paesi e regioni.

In che modo Amazon SNS recapita SMS i miei messaggi?

Quando richiedi SNS ad Amazon di inviare per tuo SMS conto, i messaggi vengono inviati tramite. AWS End User Messaging SMS L'integrazione tra Amazon SNS e AWS End User Messaging SMS offre i seguenti vantaggi:

IAMe politiche relative alle risorse

È possibile sfruttare IAM le politiche relative alle risorse per controllare e distribuire l'accesso alle SMS risorse tra altri AWS servizi e aree geografiche.

Configurazioni AWS End User Messaging SMS

Tutte le configurazioni relative all'ID di origine (creazione, aggiornamento della configurazione, fornitura di una nuova origineIDs, modifica dei modelli di registrazione) utilizzano. AWS End User Messaging SMS

AWS End User Messaging SMS fatturazione

Tuttavia, tutta la SMS fatturazione viene effettuata. AWS End User Messaging SMS Puoi consolidare la AWS spesa per i SMS carichi di lavoro, procurando e gestendo le SMS risorse in un'unica posizione centrale.

Guida introduttiva ad Amazon SNS SMS

> **Important**
>
> L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

Questo argomento ti guida nella gestione della SMS sandbox e nella configurazione di politiche basate sulle risorse per concedere ad Amazon le autorizzazioni necessarie per accedere IAM e utilizzare SNS il. AWS End User Messaging SMS APIs

Argomenti

Guida introduttiva alla gestione degli SNS SMS accessi di Amazon

> **Important**
>
> L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

Per abilitare la SMS messaggistica in AmazonSNS, devi concedere ad Amazon SNS le autorizzazioni necessarie per accedere alle tue SMS risorse e chiamarle per tuo AWS End User Messaging SMS APIs conto. Esistono due meccanismi principali che controllano questo accesso:

1. Una IAMpolitica che garantisce l'accesso a AWS End User Messaging SMS APIs
2. Politiche basate sulle risorse per concedere l'autorizzazione alle risorse AWS End User Messaging SMS

Per impostazione predefinita, SMS risorse come gli elenchi di origine IDs e di opt-out hanno politiche delle risorse che concedono l'autorizzazione di AmazonSNS.

Argomenti

- SMSIAMpolitiche
- Gestione delle SNS IAM politiche Amazon personalizzate
- Policy basate su risorse

SMSIAMpolitiche

SMS AWS Identity and Access Management (IAM) le politiche si riferiscono alle politiche che concedono ad Amazon SNS le autorizzazioni necessarie per l'accesso e l'utilizzo AWS End User Messaging SMS APIs. Queste politiche definiscono le azioni che Amazon SNS è autorizzata a eseguire quando interagisce con AWS End User Messaging SMS le risorse, come l'invio di SMS messaggi.

1. Se non utilizzi un ruolo di amministratore, allega una IAM politica che includa il `sms-voice`APIs.

   Se ti trovi nella sandbox, puoi iniziare a inviare SMS messaggi a numeri di telefono di destinazione verificati senza impostare politiche aggiuntive in materia di risorse.
2. Se hai richiesto una nuova identità di origine, seleziona la politica appropriata nella console. Ciò garantisce ad Amazon SNS l' AWS End User Messaging SMS accesso alla risorsa.
3. Se desideri utilizzare gli opt-out, l'elenco di opt-out predefinito non ha una politica delle risorse predefinita. È necessario configurare manualmente una politica delle risorse in AWS End User Messaging SMS per utilizzare Amazon SNSAPIs.

Utilizza la seguente IAM politica per accedere a tutti i SMS contenuti correlati APIsSNS:

> **Note**
>
> `sms-voice:SendTextMessage`ee l'opt-out non APIs sono presenti nell'esempio seguente.

```
{
    "Effect": "Allow",
```

```
    "Principal": { "Service": "sns.amazonaws.com" },
    "Action": [
        "sns:*",
        "sms-voice:CreateVerifiedDestinationPhoneNumber",
        "sms-voice:DeleteVerifiedDestinationPhoneNumber",
        "sms-voice:GetAccountTier",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeDestinationPhoneNumbers",
        "sms-voice:VerifyDestinationPhoneNumber",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeSpendLimits",
        "sms-voice:DescribeConfigurationSets",
        "sms-voice:SetTextMessageSpendLimitOverride",
        "sms-voice:UpdateRouteType",
        "sms-voice:UpdateSenderId"
     ]
    "Resource": "*",
    "Condition": { "StringEquals": { "aws:SourceAccount": "<owner account>" } }
}
```

Utilizza la seguente IAM politica per accedere a tutte le SMS funzionalità relative alla pubblicazione diretta inSNS:

```
{
    "Effect": "Allow",
    "Principal": { "Service": "sns.amazonaws.com" },
    "Action": [
        "sns:CreateSMSSandboxPhoneNumber",
        "sns:DeleteSMSSandboxPhoneNumber",
        "sns:GetSMSSandboxPhoneNumber",
        "sns:ListSMSSandboxPhoneNumber",
        "sns:VerifySMSSandboxPhoneNumber",
        "sns:ListOriginationNumbers",
        "sns:CheckIfPhoneNumberIsOptedOut",
        "sns:GetSMSAttributes",
        "sns:SetSMSAttributes",
        "sns:Publish",
        "sms-voice:CreateVerifiedDestinationPhoneNumber",
        "sms-voice:DeleteVerifiedDestinationPhoneNumber",
        "sms-voice:GetAccountTier",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeDestinationPhoneNumbers",
        "sms-voice:VerifyDestinationPhoneNumber",
```

```
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeSpendLimits",
        "sms-voice:DescribeConfigurationSets",
        "sms-voice:SetTextMessageSpendLimitOverride",
        "sms-voice:UpdateRouteType",
        "sms-voice:UpdateSenderId"
     ]
    "Resource": "*"
}
```

Gestione delle SNS IAM politiche Amazon personalizzate

Le IAM politiche personalizzate consentono di specificare le autorizzazioni per singoli IAM utenti, gruppi o ruoli, concedendo o limitando l'accesso a AWS risorse e azioni specifiche. Quando gestisci SNS le risorse Amazon, IAM le politiche personalizzate ti consentono di personalizzare le autorizzazioni di accesso in base ai requisiti operativi e di sicurezza della tua organizzazione.

Utilizza i seguenti passaggi per gestire IAM le politiche personalizzate per AmazonSNS:

1. Accedi a AWS Management Console e apri la IAM console all'indirizzo https://console.aws.amazon.com/iam/.
2. Dal riquadro di navigazione, scegli Politiche.
3. Per creare una nuova IAM politica personalizzata, scegli Crea politica e scegli SNS. Per modificare una politica esistente, seleziona la politica dall'elenco e scegli Modifica politica.
4. Nell'editor delle policy, definisci le autorizzazioni per l'accesso alle SNS risorse Amazon. Puoi specificare azioni, risorse e condizioni in base ai tuoi requisiti specifici.
5. Per concedere le autorizzazioni per SNS le azioni di Amazon, includi SNS azioni Amazon pertinenti come `sns:Publishsns:Subscribe`, e `sns:DeleteTopic` nella tua IAM politica. Definisci il ARN (Amazon Resource Name) degli SNS argomenti Amazon a cui si applicano le autorizzazioni.
6. Specificate gli IAM utenti, i gruppi o i ruoli a cui allegare la policy. È possibile allegare la policy direttamente a IAM utenti o gruppi o associarla ai IAM ruoli utilizzati dalle Servizi AWS nostre applicazioni.
7. Rivedi la configurazione della IAM policy per assicurarti che sia in linea con i requisiti di controllo degli accessi. Una volta verificate, salva le modifiche alla politica.
8. Allega la IAMpolitica personalizzata IAM agli utenti, ai gruppi o ai ruoli pertinenti all'interno del tuo Account AWS. Ciò concede loro le autorizzazioni definite nella politica per la gestione delle risorse AmazonSNS.

Policy basate su risorse

Le policy SNS basate sulle risorse di Amazon vengono utilizzate per controllare l'accesso alle risorse di SMS messaggistica e gestire le autorizzazioni per l'invio di messaggi per tuo conto. Queste politiche definiscono chi può eseguire azioni sulle risorse di SMS messaggistica, come l'invio di messaggi o la gestione delle identità di origine.

Configurando politiche basate sulle risorse, puoi specificare quali AWS identità o account dispongono delle autorizzazioni per accedere e interagire con la funzionalità di messaggistica di Amazon. SMS SNS Ciò contribuisce a garantire la sicurezza e la conformità limitando l'accesso agli utenti o ai sistemi autorizzati e consentendo loro di utilizzare le funzionalità SMS di messaggistica fornite da Amazon. SNS

Identità di origine

Quando SMS invii messaggi tramite AmazonSNS, puoi identificarti presso i destinatari utilizzando un'identità di origine. Utilizza la seguente politica basata sulle risorse per inviare SMS messaggi utilizzando un'identità di origine:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sms-voice:SendTextMessage",
            "Resource": "arn:aws:sms-voice:us-east-1:555555555555:phone-number/
phone-11aa2b3333c44444d55e6ffff77gggg8",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                }
            }
        }
    ]
}
```

Prerequisiti

Amazon SNS consiglia di aggiornare la tua IAM politica per includere le seguenti azioni per garantire il controllo e la visibilità completi sulle tue SNS risorse Amazon:

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

Utilizzo della SNS SMS sandbox di Amazon

SNSSMSGli account Amazon appena creati vengono inseriti automaticamente nella SMS sandbox per garantire la sicurezza sia dei AWS clienti che dei destinatari mitigando il rischio di frodi e abusi. Questo ambiente funge da spazio sicuro per scopi di test e sviluppo. Mentre operi all'interno della SMS sandbox, hai accesso a tutte le SNS funzionalità di Amazon ma sei soggetto a determinate limitazioni:

- Puoi inviare SMS messaggi 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 un minimo di 24 ore dalla verifica o dall'ultimo tentativo di verifica.

Una volta che l'account esce dalla sandbox, queste restrizioni vengono rimosse e puoi inviare SMS messaggi a qualsiasi destinatario.

Fase iniziale

I nuovi SNS SMS account Amazon vengono inseriti in una SMS sandbox. Utilizza i seguenti passaggi per creare e gestire i numeri di telefono nella sandbox, creare i numeri di origine e il mittente e registrare la tua IDs azienda.

1. Aggiungi un numero di telefono di destinazione alla sandbox. SMS Per dettagli sull'aggiunta, la gestione e lo spostamento dei numeri di telefono fuori dalla SNS SMS sandbox di Amazon, consultaAggiungere e verificare numeri di telefono nella sandbox di Amazon SNS SMS.
2. Crea un'identità di origine che i destinatari vedano sui loro dispositivi quando invii loro un messaggio. SMS Per ulteriori informazioni sulle identità di origine, compresi i diversi tipi che è possibile utilizzare, consulta la documentazione. Identità di origine per i messaggi Amazon SNS SMS

3. Registra la tua azienda. Alcuni Paesi richiedono la registrazione dell'identità dell'azienda per poter acquistare i numeri di telefono o il mittente IDs e rivedere i messaggi inviati ai destinatari nel loro paese. Per informazioni sui paesi che richiedono la registrazione, consulta Paesi e aree geografiche supportati per la SMS messaggistica AWS End User Messaging SMS nella Guida per l'AWS End User Messaging SMS utente.
4. Invia i tuoi messaggi a un argomento o a un telefono cellulare. Per ulteriori informazioni, consulta Invio SMS di messaggi tramite Amazon SNS.

Argomenti

- Aggiungere e verificare numeri di telefono nella sandbox di Amazon SNS SMS
- Eliminazione dei numeri di telefono dalla sandbox di Amazon SNS SMS
- Uscire dalla SNS SMS sandbox di Amazon

Aggiungere e verificare numeri di telefono nella sandbox di Amazon SNS SMS

Per iniziare a inviare SMS messaggi mentre il tuo AWS account è nella SMSsandbox, devi prima fare quanto segue:

1. Crea un ID di origine. Come per gli account che non si trovano nella SMS sandbox, è necessario un ID di origine prima di poter inviare SMS messaggi a destinatari in alcuni paesi o aree geografiche. Per ulteriori informazioni, consulta Scelta di un numero di telefono o di un ID mittente nella Guida per l'utente.AWS End User Messaging SMS
2. Aggiungi i numeri di telefono di destinazione alla SNS sandbox di Amazon.
3. Verifica i numeri di telefono.

Per aggiungere e verificare i numeri di telefono di destinazione

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, scegli una regione che supporta la SMS messaggistica.
3. Nel riquadro di navigazione, scegli Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo mobili (SMS), in Numeri di telefono di destinazione Sandbox, scegli Aggiungi numero di telefono.
5. In Dettagli di destinazione, inserisci il prefisso del paese e il numero di telefono, specifica la lingua da utilizzare per il messaggio di verifica, quindi scegli Aggiungi numero di telefono.

Amazon SNS invia una password monouso (OTP) al numero di telefono di destinazione. Se il numero di telefono di destinazione non lo riceve OTP entro 15 minuti, scegli Invia nuovamente il codice di verifica. Puoi inviarlo OTP allo stesso numero di telefono di destinazione fino a cinque volte ogni 24 ore.

6. Nella casella Codice di verifica, inserisci il numero di telefono OTP inviato al destinatario, quindi scegli Verifica il 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.

7. 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 messaggio OTP

Risolvi i problemi più comuni che possono impedire a un numero di telefono di ricevere SMS. OTP

- Limite di SNS SMS spesa Amazon: se hai Account AWS superato il limite di spesa per l'invio di SMS messaggi, altri messaggi, inclusi gli OTP SMS, potrebbero non essere recapitati finché il limite non viene aumentato o il problema di fatturazione non viene risolto.
- Numero di telefono non inserito nelle SMS notifiche: in alcuni paesi o aree geografiche, i destinatari devono scegliere di ricevere SMS messaggi utilizzando codici brevi, che vengono comunemente utilizzati per gli SMS. OTP Se il numero di telefono del destinatario non è abilitato, il destinatario non riceverà il testo. OTP
- Restrizioni o filtri dell'operatore: alcuni operatori di telefonia mobile possono disporre di restrizioni o meccanismi di filtraggio che impediscono la consegna di determinati tipi di SMS messaggi, inclusi gli SMS. 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 OTP testo non verrà recapitato.
- Problemi di rete: problemi o interruzioni temporanee della rete potrebbero impedire la consegna di SMS messaggi, inclusi OTP SMS, al telefono del destinatario.

- Ritardo nella consegna: in alcuni casi, SMS i messaggi possono subire ritardi nella consegna a causa della congestione della rete o di altri fattori. Alla fine il OTP testo potrebbe 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 SNS messaggistica di Amazon, inclusi gli OTP SMS, potrebbero essere sospese o interrotte.

Eliminazione dei numeri di telefono dalla sandbox di Amazon SNS SMS

Puoi eliminare dalla SMSsandbox sia i numeri di telefono di destinazione in sospeso che quelli verificati.

> **Important**
>
> Puoi eliminare un numero di telefono solo 24 ore dopo la verifica del numero di telefono o 24 ore dopo l'ultimo tentativo di verifica.

Per eliminare i numeri di telefono di destinazione dalla sandbox SMS

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, scegli una regione che supporti la SMS messaggistica in cui hai aggiunto un numero di telefono di destinazione.
3. Nel riquadro di navigazione, seleziona Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo mobili (SMS), accedi alla sezione Numeri di telefono di destinazione nella Sandbox.
5. Scegli il numero di telefono specifico che desideri eliminare, quindi scegli Elimina numero di telefono.
6. Per confermare che si desidera eliminare il numero di telefono, immettere **`delete me`** e quindi scegliere Delete (Elimina).

   Assicurati che siano trascorse almeno 24 ore dalla verifica o dal tentativo di verificare il numero di telefono di destinazione prima di procedere con l'eliminazione.
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.

Uscire dalla SNS SMS sandbox di Amazon

Account AWS Per uscire dalla SMSsandbox è necessario innanzitutto aggiungere, verificare e testare i numeri di telefono di destinazione. Dopo aver fatto ciò, crea un case con AWS Support.

Per richiedere che il tuo AWS account venga rimosso dalla SMS sandbox

1. Verifica i numeri di telefono

   a. Mentre sei nella Account AWS SMS sandbox, apri la SNSconsole Amazon.

   b. Nel pannello di navigazione, in Mobile, scegli Messaggi di testo (SMS).

   c. Nella sezione Numeri di telefono di destinazione nella Sandbox, aggiungi e verifica uno o più numeri di telefono di destinazione. Questa verifica garantisce che tu possa inviare e ricevere messaggi con successo.

2. SMSPubblicazione di prova

   - Conferma di essere in grado di inviare e ricevere messaggi su almeno un numero di telefono verificato. Per istruzioni più dettagliate su come pubblicare SMS messaggi, consultaPubblicazione di SMS messaggi su un telefono cellulare tramite Amazon SNS.

3. Avvia la modifica della sandbox

   - Nella pagina Messaggi di testo mobili (SMS) della SNS console Amazon, in Informazioni sull'account, scegli Esci dalla SMS sandbox. 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 «Messaggi di SNS testo» come servizio.

     ii. Fornisci il nome del sito Web URL o dell'app da cui intendi inviare SMS messaggi.

     iii. Specificate il tipo di messaggi che invierete: password monouso, promozionale o transazionale.

     iv. Scegli il tipo Regione AWSda cui inviare i messaggiSMS.

     v. Elenca i paesi o le aree geografiche in cui intendi inviare SMS messaggi.

     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, scegliete Aggiungi un'altra richiesta e specificate i dettagli necessari.

     v. In Nuovo valore di quota, inserisci il limite nella USD richiesta.

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 Amazon SNS SMS

> **Important**
>
> L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

Le identità di origine SMS dei messaggi sono identificatori utilizzati per rappresentare il mittente di un messaggio. SMS È possibile identificarsi presso i destinatari utilizzando i seguenti tipi di identità di origine:

- Numeri di origine: una stringa numerica che identifica il numero di telefono del mittente di un SMS messaggio. Esistono diversi tipi di numeri di origine, tra cui codici lunghi (numeri di telefono standard che in genere hanno 10 o più cifre), codici lunghi a 10 cifre (10DLC), numeri verdi () e codici brevi (numeri di telefono che contengono da quattro a sette cifreTFN). Il supporto per i numeri di origine non è disponibile nei paesi in cui le leggi locali richiedono l'uso del mittenteIDs. Quando invii un SMS messaggio 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 nel tuo AWS account, nel riquadro di navigazione della SNSconsole Amazon, scegli Origination numbers.

  Il supporto per i numeri di origine non è disponibile nei paesi in cui le leggi locali richiedono l'uso del mittente IDs anziché dei numeri di origine.

  Per ulteriori informazioni, consulta Numeri di telefono nella Guida per l'AWS End User Messaging SMS utente.

- Mittente IDs: nome alfabetico che identifica il mittente di un messaggio. SMS Quando invii un SMS messaggio utilizzando un ID mittente e il destinatario si trova in un'area in cui è supportata l'autenticazione dell'ID mittente, il tuo ID mittente viene visualizzato sul dispositivo del destinatario anziché il tuo numero di telefono. L'ID mittente fornisce SMS ai destinatari più informazioni sul mittente rispetto a un numero di telefono, un codice lungo o un codice breve.

  IDsI mittenti sono supportati in diversi paesi e aree geografiche in tutto il mondo. In alcuni paesi, se sei un'azienda che invia SMS messaggi a singoli clienti, devi utilizzare un ID mittente preregistrato presso un'agenzia di regolamentazione o un gruppo di settore. Per un elenco completo dei Paesi e delle aree geografiche che supportano o richiedono il mittenteIDs, consulta Paesi e aree geografiche supportati con cui SMS inviare messaggi AWS End User Messaging SMS nella Guida per l'AWS End User Messaging SMS utente.

Non sono previsti costi aggiuntivi per l'utilizzo del mittenteIDs. Tuttavia, il supporto e i requisiti per l'autenticazione dell'ID mittente variano in base al Paese. Diversi mercati importanti (tra cui Canada, Cina e Stati Uniti) non supportano l'utilizzo del mittenteIDs. Alcune aree richiedono che le aziende che inviano SMS messaggi a singoli clienti debbano utilizzare un ID mittente preregistrato presso un'agenzia di regolamentazione o un gruppo di settore.

Per ulteriori informazioni, consulta Sender IDs nella Guida per l'AWS End User Messaging SMS utente.

Configurazione della SMS messaggistica in Amazon SNS

> **Important**
>
> L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

Puoi utilizzare le configurazioni di Amazon SNS SMS per impostare le SMS preferenze in base alle tue esigenze, come la regolazione delle quote di spesa e l'impostazione della registrazione dello stato della spedizione. Questo argomento fornisce anche dettagli su come pubblicare SMS messaggi su argomenti utilizzando la SNS console Amazon AWS SDK, gestire in modo efficiente le quote e recuperare statistiche dettagliate sull'SMSattività.

Argomenti

- Invio SMS di messaggi tramite Amazon SNS
- Impostazione delle preferenze di SMS messaggistica in Amazon SNS
- Gestione dei numeri SNS di telefono e degli abbonamenti Amazon
- Monitoraggio SNS SMS delle attività di Amazon
- Richiesta di supporto per la messaggistica Amazon SNS SMS

Invio SMS di messaggi tramite Amazon SNS

Questa sezione descrive come inviare SMS messaggi utilizzando AmazonSNS, inclusa la pubblicazione su un argomento, l'iscrizione di numeri di telefono agli argomenti, l'impostazione degli attributi sui messaggi e la pubblicazione diretta sui telefoni cellulari.

Argomenti

Pubblicazione SMS di messaggi su un SNS argomento Amazon

Puoi pubblicare un singolo SMS messaggio su più numeri di telefono contemporaneamente sottoscrivendo tali numeri di telefono a un SNS argomento Amazon. Un SNS argomento è un canale di comunicazione a cui puoi aggiungere abbonati e quindi pubblicare messaggi per tutti gli abbonati. Un abbonato riceve tutti i messaggi pubblicati sull'argomento fino a quando non annulli l'abbonamento o finché l'abbonato non rinuncia a ricevere SMS messaggi dal tuo account. AWS

Argomenti

Invio di un messaggio a un argomento tramite la console AWS

Per creare un argomento

Completa i seguenti passaggi se non hai già un argomento a cui inviare SMS messaggi.

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, scegli una regione che supporta la SMS messaggistica.
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 Nome visualizzato, inserisci un prefisso personalizzato per i tuoi SMS messaggi. Quando invii un messaggio all'argomento, Amazon SNS inserisce il nome visualizzato seguito da una parentesi ad angolo retto (>) e uno spazio. I nomi visualizzati non fanno distinzione tra maiuscole e minuscole e Amazon SNS converte i nomi visualizzati 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 Amazon Resource Name (ARN) vengono visualizzati nella pagina Argomenti.

Per creare un SMS abbonamento

Puoi utilizzare gli abbonamenti per inviare un SMS messaggio a più destinatari pubblicando il messaggio una sola volta nel tuo argomento.

> **Note**
>
> Quando inizi a utilizzare Amazon SNS per inviare SMS messaggi, il tuo AWS account si trova nella SMSsandbox. La SMS sandbox offre un ambiente sicuro in cui provare le SNS funzionalità di Amazon senza mettere a rischio la tua reputazione di mittenteSMS. Mentre il tuo account è nella SMS sandbox, puoi utilizzare tutte le funzionalità di AmazonSNS, ma puoi inviare SMS messaggi solo a numeri di telefono di destinazione verificati. Per ulteriori informazioni, consulta Utilizzo della SNS SMS sandbox di Amazon.

1. Accedi alla SNSconsole Amazon.
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 Argomento ARN, inserisci o scegli l'Amazon Resource Name (ARN) dell'argomento a cui desideri inviare SMS i messaggi.
   b. In Protocol (Protocollo), seleziona 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 su un argomento, Amazon SNS tenta di recapitare quel messaggio a tutti i numeri di telefono abbonati all'argomento.

1. Nella SNSconsole Amazon, nella pagina Argomenti, scegli il nome dell'argomento a cui desideri inviare SMS messaggi.
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 Oggetto, mantieni il campo vuoto a meno che l'argomento non contenga sottoscrizioni e-mail e desideri pubblicare sia su e-mail che su SMS abbonamenti. Amazon SNS utilizza l'oggetto che inserisci come oggetto dell'email.

   b. (Facoltativo) Per Time to Live (TTL), inserisci il numero di secondi a disposizione di Amazon SNS per inviare il tuo SMS messaggio 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 l'argomento ha un nome visualizzato, Amazon lo SNS aggiunge al messaggio, aumentando la lunghezza del messaggio. La lunghezza del nome visualizzato è il numero

di caratteri del nome più due caratteri per la parentesi ad angolo retto (>) e lo spazio SNS aggiunto da Amazon.

Per informazioni sulle quote di dimensione per i SMS messaggi, consulta. Pubblicazione di SMS messaggi su un telefono cellulare tramite Amazon SNS

5. (Facoltativo) Per gli attributi dei messaggi, aggiungi i metadati dei messaggi come timestamp, firme e. IDs
6. Seleziona Publish message (Pubblica messaggio). Amazon SNS invia il SMS messaggio e visualizza un messaggio di successo.

Invio di un messaggio a un argomento utilizzando il AWS SDKs

Per usare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

L'esempio di codice seguente mostra come:

- Crea un SNS argomento Amazon.
- Sottoscrivere un numero di telefono cellulare all'argomento.
- Pubblica SMS messaggi sull'argomento in modo che tutti i numeri di telefono abbonati ricevano il messaggio contemporaneamente.

Java

SDKper Java 2.x

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

Crea un argomento e restituisciloARN.

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

```
import software.amazon.awssdk.services.sns.model.SnsException;

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

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
```

```
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

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

Sottoscrivere un endpoint a un argomento.

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

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

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;
```

```
        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

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

Impostare gli attributi del messaggio, ad esempio l'ID del mittente, il prezzo massimo e il relativo tipo. Gli attributi del messaggio sono facoltativi.

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

```
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

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

```
    }
}
```

Pubblicare un messaggio in un argomento. Il messaggio viene inviato a ogni abbonato.

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

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

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

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

Pubblicazione di SMS messaggi su un telefono cellulare tramite Amazon SNS

Puoi utilizzare Amazon SNS per inviare SMS messaggi direttamente a un telefono cellulare senza associare il numero di telefono a un SNS argomento Amazon.

> **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 SMS messaggio su un argomento, consultaPubblicazione SMS di messaggi su un SNS argomento Amazon.

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 Amazon SNS API o il AWS SDKs, puoi specificare un prezzo massimo per la consegna del messaggio.

Ogni SMS messaggio può contenere fino a 140 byte e la quota di caratteri dipende dallo schema di codifica. Ad esempio, un SMS messaggio può contenere:

- 160 GSM caratteri
- 140 ASCII caratteri
- 70 UCS -2 caratteri

Se pubblichi un messaggio che supera la quota di dimensioni, Amazon lo SNS invia come più messaggi, ciascuno rientra nella quota di dimensioni. I messaggi non vengono troncati nel mezzo di una parola ma tra una parola e l'altra. La quota di dimensione totale per una singola azione di SMS pubblicazione è di 1.600 byte.

Quando si invia un SMS messaggio, si specifica il numero di telefono utilizzando il formato E.164, una struttura di numerazione telefonica 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 statunitense in formato E.164 viene visualizzato come +1 0100. XXX555

Argomenti

- Invio di un messaggio (Console)
- Invio di un messaggio ()AWS SDKs

Invio di un messaggio (Console)

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, scegli una regione che supporta la SMS messaggistica.
3. Nel riquadro di navigazione, scegli Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo mobili (SMS), scegli Pubblica messaggio di testo.
5. Nella pagina Pubblica SMS messaggio, per Tipo di messaggio, scegli una delle seguenti opzioni:
    - 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. Puoi impostare un tipo di messaggio predefinito a livello di account dalla sezione Preferenze per i messaggi di testo della pagina Messaggi di testo mobili (SMS).

Per informazioni sui prezzi dei messaggi promozionali e transazionali, consulta Prezzi a livello mondiale. SMS

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 il mittente IDs varia in base al Paese e/o all'area geografica. Ad esempio, i messaggi consegnati a numeri di telefono statunitensi non visualizzeranno l'ID mittente. Per i paesi e le aree geografiche che supportano il mittenteIDs, consulta Paesi e aree geografiche supportati con cui SMS inviare messaggi AWS End User Messaging SMS nella Guida per l'AWS End User Messaging SMS utente.

     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.
     - NOTICEViene visualizzato nei paesi in cui IDs sono supportati solo i mittenti.

     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 tuo paese Account AWS di destinazione.

Il numero di origine può essere un DLC numero 10, un numero verde, un codice person-to-person lungo o codici brevi. Per ulteriori informazioni, consulta Identità di origine per i messaggi Amazon SNS SMS.

Se non specifichi un numero di origine, Amazon SNS seleziona un numero di origine da utilizzare per il messaggio di SMS testo, in base alla tua configurazione. Account AWS

9. Se SMS invii messaggi a destinatari in India, espandi gli attributi specifici del Paese e specifica i seguenti attributi:

   - ID dell'entità: l'ID dell'entità o l'ID dell'entità principale (PE) per l'invio di SMS messaggi 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 presso la quale ti sei registrato. TRAI
   - ID modello: l'ID modello per l'invio di SMS messaggi ai destinatari in India. Questo ID è una stringa univoca TRAI fornita di 1-50 caratteri che identifica il modello che hai registrato con. TRAI L'ID modello deve essere associato all'ID mittente specificato per il messaggio.

   Per ulteriori informazioni sull'invio di SMS messaggi a destinatari in India, procedura di registrazione dell'ID mittente indiano, consulta la Guida per l'utente.AWS End User Messaging SMS

10. Seleziona Publish message (Pubblica messaggio).

> **Tip**
>
> Per inviare SMS messaggi da un numero di origine, puoi anche scegliere Origination numbers nel pannello di navigazione SNS della console Amazon. Scegli un numero di origine da includere SMSnella colonna Capacità, quindi scegli Pubblica messaggio di testo.

Invio di un messaggio ()AWS SDKs

Per inviare un SMS messaggio utilizzando uno di questi AWS SDKs, utilizza l'APIoperazione corrispondente alla `Publish` richiesta in Amazon SNSAPI. SDK Con questa richiesta, puoi inviare un SMS messaggio 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 il mittente IDs varia in base al Paese o all'area geografica. Nei messaggi recapitati a numeri di telefono degli Stati Uniti, ad esempio, non viene visualizzato l'ID mittente. Per un elenco dei paesi o delle aree geografiche che supportano il mittenteIDs, consulta Paesi e aree geografiche supportati per la SMS messaggistica AWS End User Messaging SMS nella Guida per l'AWS End User Messaging SMS utente.

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 aree geografiche che richiedono un ID mittente alfabetico, NOTICEviene visualizzato 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 DLC di 10, un numero verde, un codice lungo person-to-person (P2P) o un codice breve. Per ulteriori informazioni, consulta Numeri di telefono nella Guida per l'utente.AWS End User Messaging SMS

Se non specifichi un numero di origine, Amazon SNS sceglie un numero di origine in base alla configurazione del tuo AWS account.

`AWS.SNS.SMS.MaxPrice`

Il prezzo massimo USD che sei disposto a spendere per inviare il messaggio. SMS Se Amazon SNS stabilisce che l'invio del messaggio comporterebbe un costo superiore al prezzo massimo, non invia il messaggio.

Questo attributo non ha effetto se i month-to-date SMS costi hanno già superato la quota impostata per l'attributo. `MonthlySpendLimit` È possibile impostare `MonthlySpendLimit` utilizzando l'attributo `SetSMSAttributes`.

Se stai inviando il messaggio a un SNS argomento Amazon, il prezzo massimo si applica a ciascun messaggio recapitato a ciascun numero di telefono abbonato all'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 obbligatorio solo per l'invio di SMS messaggi a destinatari in India.

Questo è l'ID dell'entità o l'ID dell'entità principale (PE) per l'invio di SMS messaggi a 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 presso la quale ti sei registrato. TRAI

`AWS.MM.SMS.TemplateId`

Questo attributo è richiesto solo per l'invio di SMS messaggi a destinatari in India.

Questo è il modello per inviare SMS messaggi a destinatari in India. Questo ID è una stringa univoca TRAI fornita da 1 a 50 caratteri che identifica il modello che hai registrato con. TRAI L'ID modello deve essere associato all'ID mittente specificato per il messaggio.

Invio di un messaggio

I seguenti esempi di codice mostrano come pubblicare SMS messaggi utilizzando AmazonSNS.

.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 API i dettagli, consulta Publish in AWS SDK for .NET APIReference.

C++

SDKper 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.

```
/**
 * 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 API i dettagli, consulta Publish in AWS SDK for C++ APIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, consulta Publish in AWS SDK for Java 2.x APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Publish in AWS SDKfor Kotlin reference API.

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;


/**
 * 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 API i dettagli, consulta Publish in AWS SDK for PHP APIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, consulta Publish in AWS SDKfor Python (Boto3) Reference. API

Impostazione delle preferenze di SMS messaggistica in Amazon SNS

Usa Amazon SNS per specificare le preferenze per la SMS messaggistica. Ad esempio, puoi specificare se ottimizzare le consegne in termini di costi o affidabilità, il limite di spesa mensile, il modo in cui vengono registrate le consegne e se iscriverti ai report di utilizzo giornalieriSMS.

Queste preferenze hanno effetto per ogni SMS messaggio inviato dal tuo account, ma puoi sovrascrivere alcune di esse quando invii un messaggio singolo. Per ulteriori informazioni, consulta Pubblicazione di SMS messaggi su un telefono cellulare tramite Amazon SNS.

Argomenti

- Impostazione delle preferenze SMS di messaggistica tramite AWS Management Console
- Impostazione delle preferenze ()AWS SDKs
- Impostazione delle preferenze SMS di messaggistica per la consegna in base al Paese

Impostazione delle preferenze SMS di messaggistica tramite AWS Management Console

1. Accedi alla SNSconsole Amazon.
2. Scegli una regione che supporti la SMS messaggistica.
3. Nel pannello di navigazione, scegli Mobile e poi Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo per dispositivi mobili (SMS), nella sezione Preferenze per i messaggi di testo, scegli 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 garantire i costi più bassi.

- 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 massima affidabilità.

Per informazioni sui prezzi dei messaggi promozionali e transazionali, consulta la pagina Prezzi globali. SMS

b. (Facoltativo) Per il limite di spesa dell'Account, inserisci l'importo (inUSD) che desideri spendere per SMS i messaggi ogni mese di calendario.

> **Important**
>
> - Per impostazione predefinita, la quota di spesa è impostata su 1,00USD. 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 SMS dei messaggi.
> - Poiché Amazon SNS è un sistema distribuito, interrompe l'invio di SMS messaggi entro pochi minuti dal superamento della quota di spesa. Durante questo intervallo, se continui a inviare SMS messaggi, potresti incorrere in costi superiori alla tua quota.

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 il mittente IDs 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 del bucket S3 deve concedere l'accesso in scrittura ad Amazon. SNS

8. Scegli Save changes (Salva modifiche).

Impostazione delle preferenze ()AWS SDKs

Per impostare SMS le tue preferenze utilizzando uno di questi AWS SDKs, utilizza l'azione corrispondente alla `SetSMSAttributes` richiesta in Amazon SNSAPI. SDK Con questa richiesta, assegni valori ai diversi SMS attributi, come la quota di spesa mensile e il SMS tipo predefinito (promozionale o transazionale). Per tutti SMS gli attributi, consulta S etSMSAttributes nell'Amazon Simple Notification Service API Reference.

I seguenti esempi di codice mostrano come usare`SetSMSAttributes`.

C++

SDKper C++

> **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.

Come usare Amazon SNS per impostare l'efaultSMSType attributo D.

```
//! 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 API i dettagli, consulta S etSMSAttributes in AWS SDK for C++ APIReference.

CLI

AWS CLI

Per impostare gli attributi dei SMS messaggi

L'`set-sms-attributes`esempio seguente imposta l'ID mittente predefinito per SMS `MyName` i messaggi su.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Questo comando non produce alcun output.

- Per API i dettagli, vedere S etSMSAttributes in AWS CLI Command Reference.

Java

SDKper Java 2.x

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

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

```
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedere S etSMSAttributes in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere S etSMSAttributes in AWS SDK for JavaScript APIReference.

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'
]);

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 API i dettagli, vedere S etSMSAttributes in AWS SDK for PHP APIReference.

Impostazione delle preferenze SMS di messaggistica per la consegna in base al Paese

Puoi gestire e controllare il SMS traffico inviando messaggi solo a paesi di destinazione specifici. Ciò garantisce che i messaggi vengano inviati solo nei paesi approvati, evitando SMS addebiti indesiderati. Le seguenti istruzioni utilizzano la configurazione Protect di Amazon Pinpoint per specificare i paesi che desideri consentire o bloccare.

1. Apri la AWS SMS console all'indirizzo https://console.aws.amazon.com/sms-voice/.
2. Nel riquadro di navigazione, in Panoramica, nella sezione Avvio rapido, scegli Crea una configurazione di protezione.
3. In Proteggi i dettagli di configurazione, inserisci un nome adatto alle aziende per la tua configurazione di protezione (ad esempio, Allow-Only-AU).
4. In base alle regole SMS nazionali, seleziona la casella di controllo Regione/Paese per bloccare l'invio di messaggi a tutti i paesi supportati.
5. Deseleziona le caselle di controllo relative ai paesi in cui desideri inviare messaggi. Ad esempio, per consentire l'invio di messaggi solo in Australia, deseleziona la casella di controllo relativa all'Australia.
6. Nella sezione Proteggi le associazioni di configurazione, in Tipo di associazione, seleziona Account predefinito. Ciò garantirà che la configurazione AWS End User Messaging SMS Protect influisca su tutti i messaggi inviati tramite AmazonSNS, Amazon Cognito e la chiamata Amazon `SendMessages`APIPinpoint.
7. Scegli Crea configurazione di protezione per salvare le tue impostazioni.

Viene visualizzato il seguente messaggio di conferma:

```
Success Protect configuration protect-abc0123456789 has been created.
```

8. Accedi alla SNSconsole Amazon.
9. Pubblica un messaggio in uno dei paesi bloccati, come l'India.

   Il messaggio non verrà recapitato. È possibile verificarlo nei registri degli errori di consegna utilizzando CloudWatch. Cerca nel gruppo di sns/region/AccountID/DirectPublishToPhoneNumber/Failurelog una risposta simile all'esempio seguente:

```
{
"notification": {
"messageId": "bd59a509-XXXX-XXXX-82f8-fbdb8cb68217",
"timestamp": "YYYY-MM-DD XX:XX:XX.XXXX“
},
"delivery": {
"destination": "+91XXXXXXXXXX",
"smsType": "Transactional",
"providerResponse": "Cannot deliver message to the specified destination country",
"dwellTimeMs": 85
},
"status": "FAILURE"
}
```

Gestione dei numeri SNS di telefono e degli abbonamenti Amazon

Amazon SNS offre diverse opzioni per gestire chi riceve SMS messaggi dal tuo account. Con una frequenza limitata, puoi attivare i numeri di telefono che hanno scelto di non ricevere SMS messaggi dal tuo account. Per interrompere l'invio di messaggi agli SMS abbonati, puoi rimuovere gli abbonamenti o gli argomenti che li pubblicano.

Argomenti

- Disattivazione della ricezione di messaggi SMS
- Gestione dei numeri di telefono e degli abbonamenti tramite la console Amazon SNS

Disattivazione della ricezione di messaggi SMS

Laddove richiesto dalle leggi e dai regolamenti locali (come Stati Uniti e Canada), SMS i destinatari possono utilizzare i propri dispositivi per annullare l'iscrizione rispondendo al messaggio con una delle seguenti risposte:

- ARRET(francese)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

Per rinunciare, il destinatario deve rispondere con lo stesso numero di origine SNS utilizzato da Amazon per recapitare il messaggio. Dopo la rinuncia, il destinatario non riceverà più i SMS messaggi recapitati dal destinatario, a Account AWS meno che tu non inserisca il numero di telefono.

Se il numero di telefono è abbonato a un SNS argomento di Amazon, la disattivazione non rimuove l'abbonamento, ma i SMS messaggi non verranno recapitati a tale abbonamento a meno che tu non attivi il numero di telefono.

Gestione dei numeri di telefono e degli abbonamenti tramite la console Amazon SNS

Puoi utilizzare la SNS console Amazon per controllare quali numeri di telefono ricevono SMS messaggi dal tuo account.

Inserimento di un numero di telefono che è stato disattivato dalla console Amazon SNS

Puoi vedere quali numeri di telefono sono stati disattivati dalla ricezione di SMS messaggi dal tuo account e puoi attivare questi numeri di telefono per riprendere l'invio di messaggi.

Puoi inserire un numero di telefono solo una volta ogni 30 giorni.

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, imposta il selettore della regione su una regione che supporta la SMS messaggistica.
3. Nel pannello di navigazione, scegli Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo mobili (SMS), nella sezione Numeri di telefono disattivati, vengono visualizzati i numeri di telefono disattivati.
5. Seleziona la casella di controllo relativa al numero di telefono che desideri attivare e scegli Accetta. Il numero di telefono non è più disattivato e riceverà SMS i messaggi che gli invierai.

Eliminazione di un SMS abbonamento dalla console Amazon SNS

Elimina un SMS abbonamento per interrompere l'invio di SMS messaggi a quel numero di telefono quando pubblichi contenuti 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 l'abbonamento e visualizza un messaggio di successo.

Eliminazione di un argomento dalla console Amazon SNS

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 e visualizza un messaggio di successo.

Gestione dei numeri di telefono e degli abbonamenti tramite il AWS SDK

Puoi utilizzarlo AWS SDKs per effettuare richieste programmatiche ad Amazon SNS e gestire i numeri di telefono che possono ricevere SMS messaggi 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 strumenti AWS SDKs e agli strumenti.

Visualizzazione di tutti i numeri di telefono disattivati utilizzando il AWS SDK

Per visualizzare tutti i numeri di telefono esclusi, invia una `ListPhoneNumbersOptedOut` richiesta ad Amazon. SNS API

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

CLI

AWS CLI

Per elencare le opzioni di disattivazione dei SMS messaggi

L'`list-phone-numbers-opted-out`esempio seguente elenca i numeri di telefono a cui è stato negato il consenso alla ricezione dei messaggi. SMS

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Per API i dettagli, vedere ListPhoneNumbersOptedOutin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi ListPhoneNumbersOptedOut AWS SDK for Java 2.xAPIReference.

PHP

SDK per PHP

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

```
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 API i dettagli, vedi ListPhoneNumbersOptedOut AWS SDK for PHPAPIReference.

Verifica se un numero di telefono è stato disattivato utilizzando il AWS SDK

Per verificare se un numero di telefono è stato disattivato, invia una `CheckIfPhoneNumberIsOptedOut` richiesta ad Amazon. SNS API

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 API i dettagli, vedi CheckIfPhoneNumberIsOptedOut AWS SDK for .NETAPIReference.

CLI

AWS CLI

Per verificare la disattivazione dei SMS messaggi relativi a 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 SMS messaggi dall'account corrente AWS .

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Per API i dettagli, vedere CheckIfPhoneNumberIsOptedOutin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi CheckIfPhoneNumberIsOptedOut AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere CheckIfPhoneNumberIsOptedOutin AWS SDK for JavaScript APIReference.

PHP

SDK per PHP

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

```
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 API i dettagli, vedi CheckIfPhoneNumberIsOptedOut AWS SDK for PHPAPIReference.

Inserimento di un numero di telefono che è stato disattivato tramite Amazon SNS API

Per inserire un numero di telefono, invia una `OptInPhoneNumber` richiesta ad Amazon SNSAPI.

Puoi inserire un numero di telefono solo una volta ogni 30 giorni.

Eliminazione di un SMS abbonamento utilizzando il AWS SDK

Per eliminare un SMS abbonamento da un SNS argomento Amazon, ottieni l'abbonamento ARN inviando una `ListSubscriptions` richiesta ad Amazon SNSAPI, quindi passala ARN a una `Unsubscribe` richiesta.

I seguenti esempi di codice mostrano come usare`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 con un abbonamentoARN.

```
/// <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 API i dettagli, consulta Annullare l'iscrizione in ReferenceAWS SDK for .NET API.

C++

SDKper C++

> **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.

```
//! 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 API maggiori dettagli, consulta Annullare l'iscrizione in AWS SDK for C++ APIReference.

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 API i dettagli, consulta Annullare l'iscrizione in AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API maggiori dettagli, consulta Annullare l'iscrizione in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, consulta Annullare l'iscrizione in AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi Unsubscribe in AWS SDKfor Kotlin reference API.

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 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 API maggiori dettagli, consulta Annullare l'iscrizione in AWS SDK for PHP APIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, consulta Unsubscribe in AWS SDKfor Python (APIBoto3) Reference.

SAP ABAP

SDKper SAP ABAP

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

```
   TRY.
       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 API maggiori dettagli, vedi Annullare l'iscrizione SAP ABAPAPIcome AWS SDK riferimento.

Eliminazione di un argomento utilizzando il AWS SDK

Per eliminare un argomento e tutte le relative sottoscrizioni, scarica l'argomento ARN inviando una `ListTopics` richiesta ad Amazon SNSAPI, quindi passala ARN alla `DeleteTopic` richiesta.

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'argomentoARN.

```
/// <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 API i dettagli, DeleteTopicconsulta AWS SDK for .NET APIReference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi DeleteTopic AWS SDK for C++APIReference.

CLI

AWS CLI

Per eliminare un SNS argomento

L'`delete-topic`esempio seguente elimina l'SNSargomento specificato.

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

Questo comando non produce alcun output.

- Per API i dettagli, vedere DeleteTopicin AWS CLI Command Reference.

Go

SDKper 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
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &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 API i dettagli, vedi DeleteTopic AWS SDK for GoAPIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi DeleteTopic AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere DeleteTopicin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento DeleteTopic AWSSDKa Kotlin API.

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 API i dettagli, vedi DeleteTopic AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere DeleteTopicPython (Boto3) Reference.AWS SDK API

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi DeleteTopicSAPABAPAPIcome riferimento.AWS SDK

Monitoraggio SNS SMS delle attività di Amazon

Monitorando la tua SMS attività, puoi tenere traccia dei numeri di telefono di destinazione, delle consegne riuscite o meno, dei motivi dell'errore, dei costi e di altre informazioni. Amazon ti SNS aiuta riepilogando le statistiche nella console, inviando informazioni ad Amazon CloudWatch e inviando report di SMS utilizzo giornalieri a un bucket Amazon S3 da te specificato.

Argomenti

- Visualizzazione delle statistiche SNS SMS di consegna di Amazon
- Monitoraggio delle SNS SMS consegne di Amazon con CloudWatch metriche e log di Amazon
- Iscrizione ai report di SMS utilizzo SNS giornalieri di Amazon

Visualizzazione delle statistiche SNS SMS di consegna di Amazon

Puoi utilizzare la SNS console Amazon per visualizzare le statistiche sulle tue SMS consegne recenti.

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, imposta il selettore della regione su una regione che supporta la SMS messaggistica.
3. Nel pannello di navigazione, scegli Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo (SMS), nella sezione Statistiche dell'account, visualizza i grafici relativi alle consegne di SMS messaggi transazionali e promozionali. 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 Iscrizione ai report di SMS utilizzo SNS giornalieri di Amazon.

Monitoraggio delle SNS SMS consegne di Amazon con CloudWatch metriche e log di Amazon

Puoi utilizzare Amazon CloudWatch e Amazon CloudWatch Logs per monitorare le consegne dei SMS messaggi.

Argomenti

Visualizzazione dei CloudWatch parametri di Amazon

Amazon raccoglie SNS automaticamente i parametri relativi alla consegna dei SMS messaggi e li invia ad Amazon. CloudWatch Puoi utilizzarli CloudWatch per monitorare questi parametri e creare allarmi per avvisarti quando una metrica supera una soglia. Ad esempio, puoi monitorare le CloudWatch metriche per conoscere la tariffa di SMS spedizione e i costi. month-to-date SMS

Per informazioni sul monitoraggio delle CloudWatch metriche, sull'impostazione degli CloudWatch allarmi e sui tipi di metriche disponibili, consulta. Monitoraggio SNS degli argomenti di Amazon tramite CloudWatch

Visualizzazione dei registri CloudWatch

Puoi raccogliere informazioni sulle consegne di SMS messaggi riuscite e non riuscite abilitando Amazon SNS a scrivere su Amazon CloudWatch Logs. Per ogni SMS messaggio inviato, Amazon SNS scrive un registro che include il prezzo del messaggio, lo stato di successo o di errore, il motivo dell'errore (se il messaggio non è riuscito), la durata del messaggio e altre informazioni.

Per abilitare e visualizzare CloudWatch i log dei tuoi messaggi SMS

1. Accedi alla SNSconsole Amazon.
2. Nel menu della console, imposta il selettore della regione su una regione che supporta la SMS messaggistica.
3. Nel pannello di navigazione, scegli Messaggi di testo (SMS).
4. Nella pagina Messaggi di testo mobili (SMS), nella sezione Preferenze per i messaggi di testo, scegli Modifica.

5. Nella pagina successiva, espandere la sezione Delivery status logging (Registrazione dello stato di consegna).
6. Per la frequenza di campionamento Success, specifica la percentuale di SMS consegne riuscite per cui Amazon SNS scriverà i log in CloudWatch Logs. Per 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 scrive i 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 Consenti per consentire ad Amazon l'accesso in SNS scrittura alle risorse del tuo account.
   - Per utilizzare un ruolo di servizio esistente, scegli Usa il ruolo di servizio esistente e quindi incolla il ARN nome nella casella del IAMruolo 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 IAM ruoli, vedere Creazione di un ruolo per un AWS servizio nella Guida per l'IAMutente.
8. Scegli Save changes (Salva modifiche).
9. Tornando alla pagina dei messaggi di testo per dispositivi mobili (SMS), vai alla sezione Registri dello stato di consegna per visualizzare tutti i log disponibili.

> **Note**
>
> A seconda dell'operatore del numero di telefono di destinazione, possono essere necessarie fino a 72 ore prima che i registri di consegna vengano visualizzati nella SNS console Amazon.

Esempio di registro per una consegna avvenuta con successo SMS

Il registro dello stato della consegna per una SMS consegna riuscita 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"
}
```

Registro di esempio per la consegna non riuscita SMS

Il registro dello stato della consegna per una SMS consegna non riuscita 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"
}
```

SMSmotivi di mancata consegna

La ragione per un SMS fallito viene fornita dall'attributo `providerResponse`. SMSi messaggi potrebbero non essere recapitati 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 è bloccato 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

Iscrizione ai report di SMS utilizzo SNS giornalieri di Amazon

Puoi monitorare le tue SMS consegne iscrivendoti ai report di utilizzo giornalieri di Amazon. SNS Per ogni giorno in cui invii almeno un SMS messaggio, Amazon SNS invia un report di utilizzo come CSV file nel bucket Amazon S3 specificato. Sono necessarie 24 ore prima che il rapporto SMS sull'utilizzo sia disponibile nel bucket S3.

Argomenti

- Informazioni contenute nei report di utilizzo giornalieri
- Sottoscrizione ai report di utilizzo giornalieri

Informazioni contenute nei report di utilizzo giornalieri

Il rapporto sull'utilizzo include le seguenti informazioni per ogni SMS messaggio 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.

- Ora di pubblicazione del messaggio (inUTC)
- ID messaggio
- Numero di telefono di destinazione
- Tipo di messaggio
- Stato della consegna
- Prezzo del messaggio (inUSD)
- 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 codice articolo, ne impostiamo il 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 chi Account AWS invia SMS messaggi, accedi alla console Amazon S3.
2. Scegli 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. Scegli Create (Crea) .
5. Nella tabella All Buckets (Tutti i bucket), seleziona il bucket.
6. Nella sezione Permissions (Autorizzazioni), scegliere Bucket policy (Policy bucket).

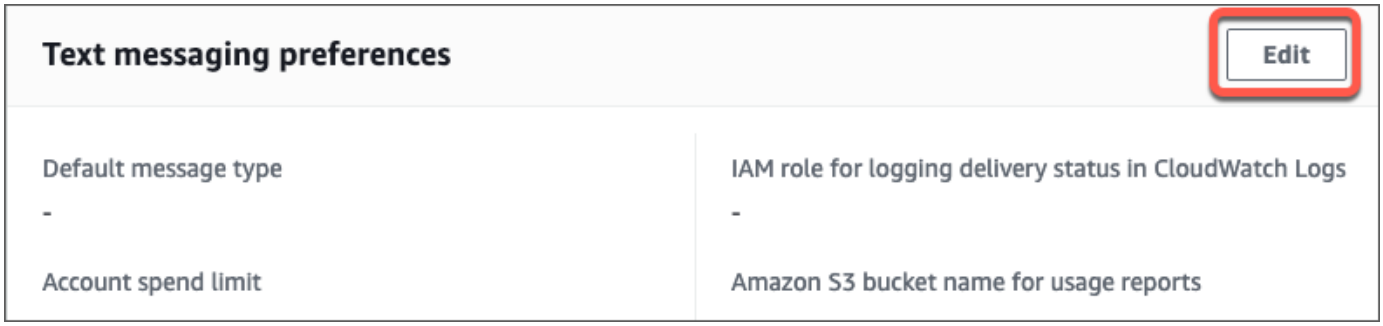
7. Nella finestra Bucket Policy Editor, fornisci una politica che consenta al responsabile del SNS servizio Amazon di scrivere nel tuo bucket. Per vedere un esempio, consulta Esempio di policy di bucket.

   Se utilizzi la politica di esempio, ricordati di sostituirla *my-s3-bucket* con il nome del bucket che hai scelto nel passaggio 3.

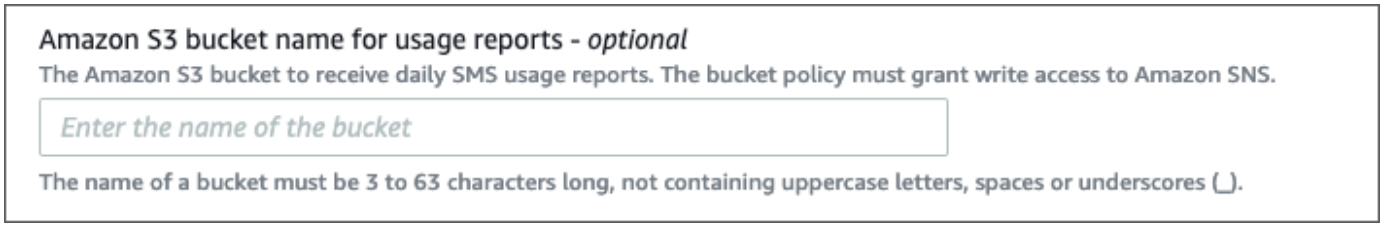
8. Seleziona Salva.

Sottoscrizione ai report di utilizzo giornalieri

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegli Messaggi di testo (SMS).
3. Nella pagina Messaggi di testo (SMS), nella sezione Preferenze per i messaggi di testo, scegli 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. Scegli Save changes (Salva modifiche).

Esempio di policy di bucket

La seguente politica consente al responsabile del SNS servizio Amazon di eseguire le `s3:ListBucket` azioni `s3:PutObject``s3:GetBucketLocation`, e.

AWS fornisce strumenti per tutti i servizi con responsabili del servizio a cui è stato concesso l'accesso alle risorse del tuo account. Quando il principale di una dichiarazione sulla politica del bucket di Amazon S3 è un problema secondario confuso. 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 conosci la ARN risorsa, usa. `aws:SourceAccount`

Usa l'esempio seguente che include la protezione confusa dei deputati quando crei un bucket Amazon S3 per ricevere report di SMS utilizzo 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:::amzn-s3-demo-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:::amzn-s3-demo-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:::amzn-s3-demo-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 report sull'utilizzo in un bucket Amazon S3 di Account AWS proprietà di un altro, vedi Come posso copiare oggetti S3 da un altro? Account AWS.

Esempio di report di utilizzo giornaliero

Dopo esserti abbonato ai report giornalieri sull'utilizzo, ogni giorno Amazon SNS inserisce un CSV file con i dati di 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 per un giorno superano questa quota, Amazon SNS aggiungerà 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
```

Richiesta di supporto per la messaggistica Amazon SNS SMS

> **Important**
>
> L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

Alcune SMS opzioni con Amazon SNS non sono disponibili per il tuo AWS account finché non ci contatti AWS Support. Apri un caso nel Center AWS Support per fare una delle richieste seguenti:

- Un aumento della soglia di SMS spesa mensile

  Per impostazione predefinita, la soglia di spesa mensile è di 1,00 USD ()USD. La tua soglia di spesa determina il volume di messaggi che puoi inviare con AmazonSNS. Puoi richiedere una soglia di spesa che soddisfi il volume mensile di messaggi previsto per il tuo caso SMS d'uso.
- Una mossa dalla SMSsandbox in modo da poter inviare SMS messaggi senza restrizioni. Per ulteriori informazioni, consulta Uscire dalla SNS SMS sandbox di Amazon.
- 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 il mittente IDs varia in base al

Paese o all'area geografica. Per ulteriori informazioni, consulta Paesi e aree geografiche supportati per la SMS messaggistica AWS End User Messaging SMS nella Guida per l'AWS End User Messaging SMS utente.

Argomenti

- Richiesta di aumenti della tua quota di SNS SMS spesa mensile su Amazon

Richiesta di aumenti della tua quota di SNS SMS spesa mensile su Amazon

Amazon SNS fornisce quote di spesa per aiutarti a gestire i costi mensili massimi sostenuti per l'invio SMS tramite 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 per interrompere la pubblicazione di SMS messaggi quando determina che l'invio di un SMS messaggio comporterà un costo superiore alla 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 Step 1: Aprire un SNS SMS case Amazon. Una volta ricevuta la quota, puoi gestire il rischio applicando la quota completa o un valore inferiore, come descritto nel Passaggio 2: Aggiorna SMS le impostazioni. Applicando un valore più basso, puoi controllare la spesa mensile con la possibilità di aumentarla in base alle esigenze.

> **Important**
>
> Poiché Amazon SNS è un sistema distribuito, interrompe l'invio di SMS messaggi in pochi minuti se la quota di spesa viene superata. Durante questo periodo, se continui a inviare SMS messaggi, potresti incorrere in costi superiori alla tua quota.

Abbiamo fissato la quota di spesa per tutti i nuovi account a $1,00 () USD al mese. Questa quota ha lo scopo di consentirti di testare le funzionalità di invio di messaggi di Amazon. SNS Per richiedere un aumento della quota di SMS spesa per il tuo account, apri una richiesta di aumento della quota nel AWS Support Center.

Argomenti

- Passaggio 1: apri una SNS SMS custodia Amazon
- Passaggio 2: aggiorna SMS le impostazioni sulla SNS console Amazon

Passaggio 1: apri una SNS SMS custodia Amazon

Puoi richiedere un aumento della tua quota di spesa mensile aprendo un caso di aumento della quota nel 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 AWS Management Console a 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 SNS Messaggi di testo.
   - (Facoltativo) Per fornire un link al sito o all'app che invierà SMS i messaggi, fornisci informazioni sul sito Web, sull'applicazione o sul servizio che invierà SMS i messaggi.
   - 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) Da quale AWS regione invierai i messaggi, scegli la regione da cui invierai i messaggi.
   - (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, inserisci l'importo massimo (inUSD) che puoi spendere per SMS 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 SMS messaggi.
   - Il servizio fornito dal tuo sito Web o dalla tua app e il modo in cui SMS i tuoi messaggi contribuiscono a tale servizio.
   - In che modo gli utenti si iscrivono per ricevere volontariamente SMS i tuoi messaggi sul tuo sito web, app o altro luogo.

   Se la quota di spesa richiesta (il valore che hai specificato per Nuovo valore della quota) supera i 10.000 USD (USD), fornisci i seguenti dettagli aggiuntivi per ogni paese in cui stai inviando 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 valore massimo previsto transactions-per-second (TPS) per i tuoi 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 Amazon SNS SMS.
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 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

Passaggio 2: aggiorna SMS le impostazioni sulla SNS console Amazon

Dopo che ti abbiamo comunicato che la tua quota di spesa mensile è stata aumentata, devi modificare la quota di spesa per il tuo account sulla SNS console Amazon.

> **Important**
>
> Devi completare i seguenti passaggi o il tuo limite di SMS spesa non verrà aumentato.

Per modificare la quota di spesa nella console

1. Accedi alla SNSconsole Amazon.
2. Apri il menu di navigazione a sinistra, espandi Mobile, quindi scegli Messaggi di testo (SMS).
3. Nella pagina Messaggi di testo mobili (SMS), nella sezione Preferenze per i messaggi di testo, scegli Modifica.

4. Nella pagina Modifica delle preferenze per i messaggi di testo, nella sezione Dettagli, inserisci il nuovo limite di SMS spesa nel campo Limite di spesa dell'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. Scegli Save changes (Salva modifiche).

> **Note**
>
> Se ricevi un errore «Parametro non valido», controlla il contatto di AWS Support e conferma di aver inserito il nuovo limite di SMS spesa corretto. Se il problema persiste, apri una richiesta nel AWS Support Center.

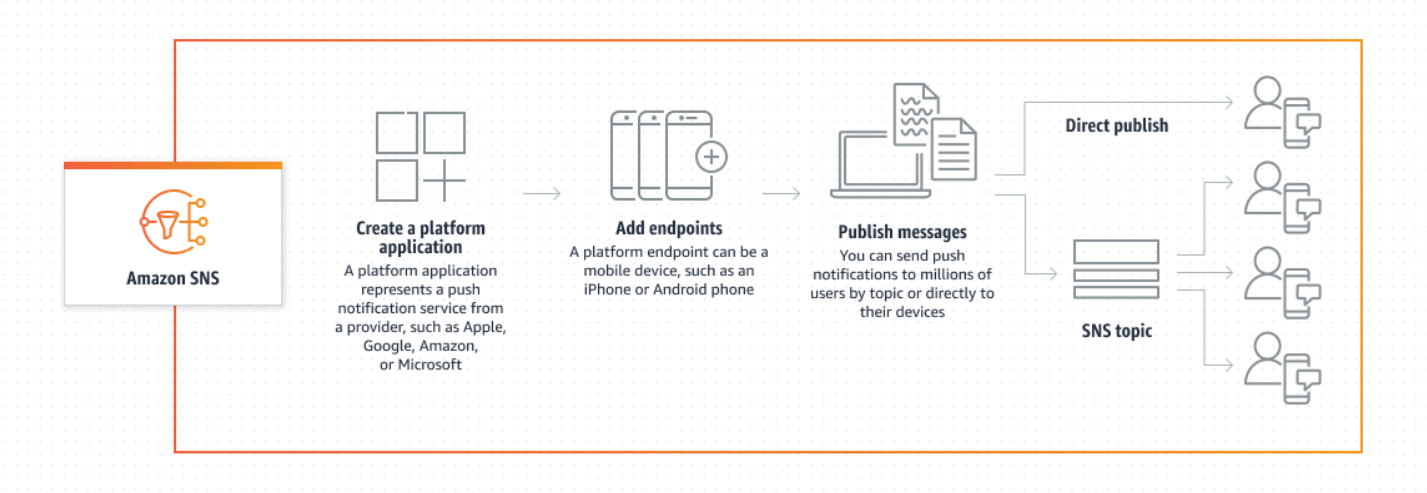
Quando crei il tuo caso nel AWS Support Centro, assicurati di includere tutte le informazioni richieste per il tipo di richiesta che stai inviando. Altrimenti, AWS Support devi contattarti per ottenere queste informazioni prima di procedere. Inviando un caso dettagliato, aumenti le probabilità che venga soddisfatto senza ritardi. Per i dettagli richiesti per tipi specifici di SMS richieste, consulta i seguenti argomenti.

Per ulteriori informazioni sul mittenteIDs, consulta la seguente documentazione nella Guida per l'AWS End User Messaging SMS utente:

AWS End User Messaging SMS Argomento	Descrizione
Richiesta di un aumento della quota di spesa	La tua quota di spesa determina la quantità di denaro che puoi spendere per inviare SMS messaggi AWS End User Messaging SMS ogni mese.
Apri una richiesta nel centro assistenza per un ID mittente	Se prevedi di inviare messaggi ai destinatari di un Paese in cui è richiesto il mittenteIDs, puoi richiedere un ID mittente creando una nuova richiesta nel Centro. AWS Support

Invio di notifiche push per dispositivi mobili con Amazon SNS

Puoi usare Amazon SNS per inviare messaggi di notifica push direttamente alle app sui dispositivi mobili. I messaggi di notifica push inviati a un dispositivo mobile possono apparire nell'app mobile come avvisi di messaggi, aggiornamenti di badge o avvisi sonori.



Argomenti

- Come funzionano le notifiche agli SNS utenti di Amazon
- Configurazione delle notifiche push con Amazon SNS
- Configurazione di un'app mobile in Amazon SNS
- Utilizzo di Amazon SNS per le notifiche push su dispositivi mobili
- Attributi SNS delle app mobili Amazon
- Notifiche degli eventi delle SNS applicazioni Amazon per applicazioni mobili
- APIAzioni push per dispositivi mobili
- APIErrori push comuni di Amazon SNS mobile
- Utilizzo dell'attributo Amazon SNS time to live message per le notifiche push mobili
- Regioni supportate dalle applicazioni SNS mobili Amazon
- Le migliori pratiche per la gestione delle notifiche push di Amazon per SNS dispositivi mobili

Come funzionano le notifiche agli SNS utenti di Amazon

L'invio di messaggi di notifica push a desktop e dispositivi mobili viene eseguito mediante uno dei seguenti servizi di notifica push supportati:

- Messaggistica per dispositivi Amazon (ADM)
- Servizio Apple Push Notification (APNs) per iOS e Mac OS X
- Baidu Cloud Push (Baidu)
- Messaggistica Firebase Cloud () FCM
- Servizio Microsoft Push Notification per Windows Phone (MPNS)
- Servizi di notifica push di Windows (WNS)

I servizi di notifica push, come APNs eFCM, mantengono una connessione con ogni app e dispositivo mobile associato registrati per utilizzare il loro servizio. Quando si registra un'app e un dispositivo mobile, la notifica push restituisce un token di dispositivo. Amazon SNS utilizza il token del dispositivo per creare un endpoint mobile, al quale può inviare messaggi di notifica push diretti. Per consentire SNS ad Amazon di comunicare con i diversi servizi di notifica push, devi inviare le credenziali del servizio di notifica push SNS ad Amazon affinché vengano utilizzate per tuo conto. Per ulteriori informazioni, consulta Configurazione delle notifiche push con Amazon SNS.

Oltre a inviare messaggi di notifica push diretti, puoi utilizzare Amazon anche SNS per inviare messaggi agli endpoint mobili abbonati a un argomento. Il concetto è lo stesso della sottoscrizione di altri tipi di endpoint, come AmazonSQS, HTTP /S, e-mail eSMS, a un argomento, come descritto in. Che cos'è AmazonSNS? La differenza è che Amazon SNS comunica utilizzando i servizi di notifica push in modo che gli endpoint mobili abbonati ricevano messaggi di notifica push inviati all'argomento.

Configurazione delle notifiche push con Amazon SNS

1. Ottenere le credenziali e il token del dispositivo per le piattaforme mobili che si desidera supportare.
2. Utilizza le credenziali per creare un oggetto applicativo della piattaforma (`PlatformApplicationArn`) utilizzando AmazonSNS. Per ulteriori informazioni, consulta Creazione di un'applicazione SNS della piattaforma Amazon.
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. Usa il token del dispositivo e il `PlatformApplicationArn` per creare un oggetto endpoint della piattaforma (`EndpointArn`) utilizzando AmazonSNS. Per ulteriori informazioni, consulta Configurazione di un endpoint SNS della piattaforma Amazon per le notifiche mobili.

5. `EndpointArn` viene quindi utilizzato per pubblicare un messaggio in una app in un dispositivo mobile. Per ulteriori informazioni, consulta la Messaggistica diretta SNS sui dispositivi mobili Amazon pagina di API riferimento per la pubblicazione API in Amazon Simple Notification Service.

Configurazione di un'app mobile in Amazon SNS

Questo argomento descrive come configurare le applicazioni mobili AWS Management Console utilizzando le informazioni descritte inPrerequisiti per le notifiche agli SNS utenti di Amazon.

Argomenti

- Prerequisiti per le notifiche agli SNS utenti di Amazon
- Creazione di un'applicazione SNS della piattaforma Amazon
- Configurazione di un endpoint SNS della piattaforma Amazon per le notifiche mobili
- Integrazione dei token dei dispositivi con Amazon SNS per le notifiche mobili
- Metodi di autenticazione delle notifiche push di Amazon SNS Apple
- SNSIntegrazione di Amazon con la configurazione dell'autenticazione di Firebase Cloud Messaging
- SNSGestione Amazon degli endpoint Firebase Cloud Messaging

Prerequisiti per le notifiche agli SNS utenti di Amazon

Per iniziare a utilizzare le notifiche push di Amazon SNS per dispositivi mobili, avrai bisogno di quanto segue:

- Un set di credenziali per la connessione a uno dei servizi di notifica push supportati:ADM,APNs, Baidu,FCM, MPNS o. WNS
- Un token di dispositivo o un ID di registrazione per l'app per dispositivi mobili e il dispositivo.
- Amazon è SNS configurato per inviare messaggi di notifica push agli 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 messaggi di notifica push diretti 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 utilizza APNs come servizio di notifica push, è necessario un token del dispositivo. In alternativa, quando si utilizzaFCM, l'equivalente del token del dispositivo viene chiamato 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 memorizza le credenziali (più alcune altre impostazioni) come risorsa applicativa della piattaforma. I token del dispositivo (sempre con alcune impostazioni aggiuntive) sono rappresentati come oggetti chiamati endpoint della 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. Una volta ottenute le informazioni sui prerequisiti, puoi inviare un messaggio di notifica push utilizzando il AWS Management Console o il push SNS APIs mobile di Amazon. Per ulteriori informazioni, consulta Configurazione delle notifiche push con Amazon SNS.

Creazione di un'applicazione SNS della piattaforma Amazon

SNSAffinché Amazon possa inviare messaggi di notifica agli endpoint mobili, direttamente o tramite abbonamenti a un argomento, devi prima creare un'applicazione di piattaforma. Dopo aver registrato l'app con AWS, devi creare un endpoint per l'app e il dispositivo mobile. Amazon SNS utilizza questo endpoint per inviare messaggi di notifica all'app e al dispositivo.

Per creare un'applicazione della piattaforma

1. Accedi alla SNSconsole Amazon.
2. Nel riquadro di navigazione, scegli Notifiche push.

3. Nella sezione Platform applications (Applicazioni di piattaforma), scegli Create platform application (Crea applicazione di piattaforma).

   Per un elenco delle AWS regioni in cui è possibile creare applicazioni mobili, consultaRegioni supportate dalle applicazioni SNS mobili Amazon.

4. Inserisci un nome per rappresentare la tua app. I nomi delle app devono essere composti solo da ASCII lettere 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 utilizzi una delle piattaforme Apple Push Notification Service (APNs), puoi scegliere tra l'autenticazione basata su token o certificato, quindi scegliere 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).

   Questo registra l'app con AmazonSNS, che crea un oggetto applicativo della piattaforma per la piattaforma selezionata e quindi ne restituisce uno corrispondente PlatformApplicationArn.

Configurazione di un endpoint SNS della piattaforma Amazon per le notifiche mobili

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 del dispositivo per creare un endpoint mobile, al quale può inviare messaggi di notifica push diretti. Per ulteriori informazioni, consulta Prerequisiti per le notifiche agli SNS utenti di Amazon e Configurazione delle notifiche push con Amazon SNS.

Questa sezione descrive l'approccio consigliato per creare un endpoint di piattaforma.

Argomenti

- Creazione di un endpoint di piattaforma
- Pseudocodice
- AWS SDKesempio
- Risoluzione dei problemi

Creazione di un endpoint di piattaforma

Per inviare notifiche a un'app con AmazonSNS, il token del dispositivo dell'app deve prima essere registrato su Amazon SNS chiamando l'azione create platform endpoint. Questa azione prende l'Amazon Resource Name (ARN) dell'applicazione della piattaforma e il token del dispositivo come parametri e restituisce l'endpoint ARN della piattaforma creato.

L'`CreatePlatformEndpoint`azione esegue le seguenti operazioni:

- Se l'endpoint della piattaforma esiste già, non crearlo di nuovo. Restituisci al chiamante l'ARNendpoint della piattaforma esistente.
- Se l'endpoint della piattaforma con lo stesso token del dispositivo ma impostazioni diverse esiste già, non crearlo di nuovo. Genera un'eccezione per l'intermediario.
- Se l'endpoint della piattaforma non esiste, crealo. Restituisci al chiamante l'endpoint ARN della 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
      (GetEndpointAttributes 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 Può essere utilizzato anche per notificare ad Amazon una modifica SNS del token del dispositivo. 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. Può essere richiamata all'inizio, quando l'app non conosce l'endpoint della propria piattaformaARN, come accade durante la prima registrazione. Viene chiamata anche se la chiamata all'`GetEndpointAttributes`azione iniziale fallisce con un'eccezione non trovata, come accadrebbe se l'applicazione conoscesse il suo endpoint ARN ma venisse eliminata.
- L'`GetEndpointAttributes`azione viene richiamata per verificare lo stato dell'endpoint della piattaforma anche se l'endpoint della piattaforma 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 SDKesempio

Il codice seguente mostra come implementare lo pseudo codice precedente utilizzando SNS i client Amazon forniti da. AWS SDKs

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

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

SDKper Java 2.x

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

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

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

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

Per ulteriori informazioni, consulta APIAzioni 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 FCM endpoint, 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 poiché libera l'app dalla necessità di gestire lo stato del token del dispositivo e Amazon SNS aggiornerà automaticamente il token del dispositivo al valore più recente. In verità, questa soluzione presenta alcuni seri inconvenienti:

- Amazon SNS si affida FCM al feedback ricevuto per aggiornare i token del dispositivo scaduti con nuovi token del dispositivo. FCMconserva le informazioni sui token dei vecchi dispositivi per un certo periodo di tempo, ma non a tempo indeterminato. Una volta FCM dimenticata la connessione tra il vecchio token del dispositivo e il nuovo token del dispositivo, Amazon non SNS sarà più in grado di aggiornare il token del dispositivo archiviato nell'endpoint della piattaforma al suo valore corretto; si limiterà invece a disabilitare l'endpoint della piattaforma.
- L'applicazione di piattaforma conterrà molteplici endpoint di piattaforma corrispondenti allo stesso token di dispositivo.
- Amazon SNS impone una quota sul numero di endpoint della piattaforma che possono essere creati a partire dallo stesso token del 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 FCM endpoint, consulta. SNSGestione Amazon degli endpoint Firebase Cloud Messaging

Riattivazione di un endpoint di piattaforma associato a un token di dispositivo non valido

Quando una piattaforma mobile (come APNs oFCM) informa Amazon SNS che il token del dispositivo utilizzato nella richiesta di pubblicazione non era valido, Amazon SNS disabilita l'endpoint della piattaforma associato a quel token del dispositivo. Amazon SNS rifiuterà quindi le pubblicazioni successive su quel token del 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. La prossima volta che verrà pubblicato su, la piattaforma mobile informerà nuovamente Amazon SNS che il token del dispositivo non è valido e Amazon SNS disabiliterà nuovamente l'endpoint della 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.

Integrazione dei token dei dispositivi con Amazon SNS per le notifiche mobili

Quando registri per la prima volta un'app e un dispositivo mobile con un servizio di notifica, come Apple Push Notification Service (APNs) e Firebase Cloud Messaging (FCM), i token del dispositivo o la registrazione IDs vengono restituiti dal servizio di notifica. Quando aggiungi i token del dispositivo o la registrazione IDs ad AmazonSNS, vengono utilizzati con il `PlatformApplicationArn` API per creare un endpoint per l'app e il dispositivo. Quando Amazon SNS crea l'endpoint, `EndpointArn` viene restituito un. In `EndpointArn` questo modo Amazon SNS sa a quale app e dispositivo mobile inviare il messaggio di notifica.

Puoi aggiungere i token del dispositivo e la registrazione IDs ad Amazon SNS utilizzando i seguenti metodi:

- Aggiungi manualmente un singolo token all' AWS utilizzo di AWS Management Console
- Carica diversi token utilizzando il `CreatePlatformEndpoint` API
- 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 SNSconsole Amazon.
2. Nel riquadro di navigazione, scegli Notifiche push.
3. Nella sezione Applicazioni della piattaforma, seleziona l'applicazione, quindi scegli Modifica. Se non hai già creato un'applicazione di piattaforma, creane una ora. Per istruzioni su come eseguire questa operazione, consultare Creazione di un'applicazione SNS della piattaforma Amazon.
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 inserisci l'ID di registrazione.
6. (Opzionale) Nella casella User Data (Dati utente), immetti le informazioni arbitrarie da associare all'endpoint. Amazon SNS non utilizza questi dati. I dati devono essere in formato UTF -8 e inferiori a 2 KB.

7. Scegli Add endpoints.

   Con l'endpoint creato, è possibile inviare messaggi direttamente a un dispositivo mobile o inviare messaggi a dispositivi mobili abbonati a un argomento.

Per caricare diversi token utilizzando il **`CreatePlatformEndpoint`** API

I passaggi seguenti mostrano come utilizzare l'app Java di esempio (`bulkupload`pacchetto) fornita da AWS per caricare diversi token (token del dispositivo o registrazioneIDs) su Amazon. SNS Puoi usare questa app di esempio per iniziare a caricare i token esistenti.

> **Note**
>
> I passaggi seguenti utilizzano Eclipse Java. IDE I passaggi presuppongono che tu abbia installato AWS SDK for Java e che tu disponga delle credenziali AWS di sicurezza 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 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. Scarica una copia della CSVlibreria Open e aggiungila al Build Path del `bulkupload` pacchetto.
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 CSV file contenente i token.
   - I nomi dei CSV file (come `goodTokens.csv` e`badTokens.csv`) da creare per registrare i token che Amazon SNS analizza correttamente e quelli che falliscono.
   - (Facoltativo) I caratteri per specificare il delimitatore e la citazione nel file contenente i CSV 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' BatchCreatePlatformEndpointSampleapplicazione.java per caricare i token su Amazon. SNS

   In questo esempio, gli endpoint creati per i token che sono stati caricati correttamente su Amazon SNS verrebbero registrati`goodTokens.csv`, mentre i token con formato errato verrebbero registrati. `badTokens.csv` Inoltre, dovresti vedere STD OUT i log scritti sulla 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:

- Usa il servizio Amazon Cognito: la tua app mobile avrà bisogno di credenziali per creare endpoint associati alla tua 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 su AmazonSNS, puoi registrarti per ricevere un SNS evento Amazon che fornirà il nuovo endpointARN. Puoi anche utilizzare il

`ListEndpointByPlatformApplication` API per ottenere l'elenco completo degli endpoint registrati con AmazonSNS.

- 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 di Amazon per SNS dispositivi mobili, insieme a tutti i dati utente che desideri archiviare. A tal fine, il server proxy si connetterà ad Amazon SNS utilizzando AWS le tue credenziali e utilizzerà la `CreatePlatformEndpoint` API chiamata per caricare le informazioni sul token. Verrà restituito l'endpoint Amazon Resource Name (ARN) appena creato, che il server può archiviare per effettuare successive chiamate di pubblicazione ad AmazonSNS.

Metodi di autenticazione delle notifiche push di Amazon SNS Apple

Puoi autorizzare Amazon SNS a inviare notifiche push alla tua app iOS o macOS fornendo informazioni che ti identificano 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 che Amazon SNS utilizza per firmare i token di autenticazione Apple Push Notification Service (APNs).

Se fornisci una chiave di firma, Amazon SNS utilizza un token per l'autenticazione APNs per ogni notifica push che invii. Con la tua chiave di firma, puoi inviare notifiche push agli ambienti di APNs 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 APNs utilizzando i token di autenticazione nella sezione Aiuto per gli account sviluppatore del sito web di Apple.

Certificate

Un TLS certificato che Amazon SNS utilizza per l'autenticazione APNs quando invii notifiche push. Si può ottenere il certificato dal proprio account sviluppatore Apple.

I certificati scadono dopo un anno. Quando ciò accade, devi creare un nuovo certificato e fornirlo ad AmazonSNS. Per ulteriori informazioni, consulta Stabilire una connessione basata su certificati APNs sul sito Web per sviluppatori Apple.

Per gestire le APNs impostazioni utilizzando la console di gestione AWS

1. Accedi alla SNSconsole Amazon.
2. In Mobile, scegli Push notification (Notifiche push).
3. Seleziona l'applicazione per la quale desideri modificare le APNs impostazioni, quindi scegli Modifica.
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 nel tuo account Apple Developer, scegli Certificati IDs e profili, quindi scegli la tua chiave nella sezione 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 Certificati IDs e profili, scegli App IDs nella sezione Identificatori, quindi scegli la tua app.
   - Se si sceglie Certificate (Certificato), è necessario fornire le seguenti informazioni:
     - SSLcertificato: il file.p12 per il tuo 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.
     - Carica certificato: scegli Carica certificato per caricare il certificato.
7. Al termine, scegliere Save changes (Salva modifiche).

SNSIntegrazione di Amazon con la configurazione dell'autenticazione di Firebase Cloud Messaging

Questo argomento descrive come ottenere da Google le credenziali richieste FCM API (HTTPv1) da utilizzare con AWS API, AWS CLI e. AWS Management Console

Argomenti

> **Important**
>
> 20 giugno 2023: Google ha dichiarato obsoleta la sua versione legacy di Firebase Cloud Messaging (). FCM HTTP API Amazon SNS ora supporta la distribuzione a tutti i tipi di dispositivi utilizzando la FCM HTTP versione 1API. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima versione FCM HTTP v1 entro il API 1° giugno 2024 per evitare interruzioni.
> 18 gennaio 2024 — Amazon SNS ha introdotto il supporto per la FCM HTTP versione 1 API per la consegna di notifiche push mobili ai dispositivi Android.
> 26 marzo 2024: Amazon SNS supporta la versione FCM HTTP 1 API per i dispositivi Apple e le destinazioni Webpush. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima versione FCM HTTP v1 entro il API 1° giugno 2024 per evitare interruzioni delle applicazioni.

Puoi autorizzare Amazon SNS a inviare notifiche push alle tue applicazioni fornendo informazioni che ti identificano come sviluppatore dell'app. Per l'autenticazione, fornisci una APIchiave o un token durante la creazione di un'applicazione di piattaforma. Puoi ottenere le seguenti informazioni dalla console dell'applicazione Firebase:

APIChiave

La API chiave è una credenziale utilizzata per chiamare Firebase's Legacy. API The FCM Legacy APIs verrà rimosso da Google il 20 giugno 2024. Se attualmente utilizzi una API chiave come

credenziale di piattaforma, puoi aggiornare la credenziale della piattaforma selezionando Token come opzione e caricando il JSON file associato per la tua applicazione Firebase.

Token

Quando si chiama la v1, viene utilizzato un token di accesso di breve durata. HTTP API Questo è quello consigliato da Firebase API 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

Devi ottenere le tue credenziali FCM service.json prima di poter iniziare a gestire le impostazioni FCM in Amazon. SNS Per ottenere le tue credenziali service.json, consulta Migrare dalla versione precedente alla versione 1 nella documentazione di Google Firebase. FCM APIs HTTP

Gestione delle impostazioni tramite FCM CLI

Puoi creare notifiche FCM push utilizzando AWS API. Il numero e le dimensioni delle SNS risorse Amazon 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 FCM push insieme a un SNS argomento Amazon (AWS API)

Quando utilizzi le credenziali chiave, `PlatformCredential` è `API key`. Quando si utilizzano le credenziali del token, si `PlatformCredential` tratta di un file di chiave privata JSON formattato:

- `CreatePlatformApplication`

Per recuperare un tipo di FCM credenziale per un SNS argomento Amazon esistente ()AWS API

Recupera il tipo di credenziali, `"AuthenticationMethod": "Token"` o `"AuthenticationMethod": "Key"`:

- GetPlatformApplicationAttributes

Per impostare un FCM attributo per un SNS argomento Amazon esistente (AWS API)

Imposta l'FCMattributo:

- SetPlatformApplicationAttributes

Gestione FCM delle impostazioni tramite la console

È possibile creare notifiche FCM push utilizzando AWS Command Line Interface (CLI). Il numero e le dimensioni delle SNS risorse Amazon in un AWS account sono limitati. Per ulteriori informazioni, consulta Endpoint e quote di Amazon Simple Notification Service.

Per creare una notifica FCM push insieme a un SNS argomento Amazon (AWS CLI)

Quando utilizzi le credenziali chiave, `PlatformCredential` è `API key`. Quando si utilizzano le credenziali del token, si `PlatformCredential` tratta di un file di chiave privata JSON formattato. Quando si utilizza il 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

Per recuperare un tipo di FCM credenziale per un SNS argomento Amazon esistente ()AWS CLI

Recupera il tipo di credenziali, `"AuthenticationMethod": "Token"` o `"AuthenticationMethod": "Key"`:

- get-platform-application-attributes

Per impostare un FCM attributo per un SNS argomento Amazon esistente (AWS CLI)

Imposta l'FCMattributo:

- set-platform-application-attributes

Gestione FCM delle impostazioni (console)

Utilizza la procedura seguente per inserire le credenziali utilizzate dall'applicazione per la connessione. FCM

1. Accedi alla SNSconsole Amazon.
2. In Mobile, scegli Push notification (Notifiche push).

3. Seleziona un'FCMapplicazione esistente e scegli Modifica. Se non hai già creato un'applicazione di piattaforma, consulta Creazione di un'applicazione SNS della piattaforma Amazon.
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 API chiave Google.
5. Al termine, scegliere Save changes (Salva modifiche).

Argomenti correlati

- Utilizzo dei payload di Google Firebase Cloud Messaging v1 in Amazon SNS

SNSGestione Amazon degli endpoint Firebase Cloud Messaging

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, l'SNSendpoint ARNs Amazon corrispondente e i timestamp sul tuo server delle applicazioni.
2. Rimuovi tutti i token obsoleti ed elimina l'endpoint Amazon SNS corrispondente. ARNs

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 su Amazon SNS come endpoint della piattaforma. Ti consigliamo di archiviare il token del dispositivo, l'endpoint ARN della SNS piattaforma Amazon e il timestamp salvandoli sul tuo server delle applicazioni o su un altro archivio persistente. Per configurare FCM l'applicazione 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'SNSendpoint Amazon corrispondente con il nuovo token. Ciò consente SNS ad Amazon 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 FCM pseudo codice per gestire e mantenere i token del dispositivo.

```
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`(HTTP404) — Quando Amazon SNS riceve questa eccezione, riceverai un evento di errore `FailureType` di `InvalidPlatformToken` consegna con un token di piattaforma *`FailureMessage`* of e uno di Platform associato all'endpoint non valido. Amazon SNS disabiliterà l'endpoint della tua piattaforma quando una consegna fallisce, con questa eccezione.
- `INVALID_ARGUMENT`(HTTP400) — 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 ErrorCodedi Google su Firebase.

Poiché `INVALID_ARGUMENT` può essere restituito in entrambi i casi, Amazon SNS restituirà un corpo *`FailureType`* *`FailureMessage`* di *`InvalidNotification`* notifica e uno 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 non SNS disattiverà l'endpoint della tua 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 tua piattaforma quando una consegna fallisce, con questa eccezione.

Tutti i codici di errore rilevati ricevuti dalla FCM v1 API sono disponibili CloudWatch quando configuri la registrazione dello stato di consegna per la tua applicazione.

Per ricevere gli eventi di consegna per la 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 un `EventDeliveryFailure` evento con `FailureType` of `EndpointDisabled` e un `FailureMessage` endpoint è disabilitato. Per ricevere gli eventi di consegna della tua applicazione, consulta. Eventi applicazione disponibili

Quando ricevi questo errore da AmazonSNS, devi rimuovere o aggiornare il token obsoleto nell'applicazione della tua piattaforma.

Utilizzo di Amazon SNS per le notifiche push su dispositivi mobili

Questa sezione descrive come inviare un messaggio di notifica push.

Argomenti

- Pubblicazione in un argomento
- Messaggistica diretta SNS sui dispositivi mobili Amazon
- Pubblicazione di SNS notifiche Amazon con payload specifici della piattaforma

Pubblicazione in un argomento

Puoi anche usare Amazon SNS per inviare messaggi agli endpoint mobili abbonati a un argomento. Il concetto è lo stesso della sottoscrizione di altri tipi di endpoint, come AmazonSQS, HTTP /S, e-mail eSMS, a un argomento, come descritto in. Che cos'è AmazonSNS? La differenza è che Amazon SNS comunica tramite 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.

Messaggistica diretta SNS sui dispositivi mobili Amazon

Puoi inviare messaggi di notifica SNS push di Amazon direttamente a un endpoint che rappresenta un'applicazione su un dispositivo mobile.

Per inviare un messaggio diretto

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegli Push notifications (Notifiche push).
3. Nella pagina delle notifiche push per dispositivi mobili, nella sezione Applicazioni della piattaforma, scegli il nome dell'applicazione, ad esempio ***MyApp***.
4. Sul ***MyApp*** nella sezione Endpoint, 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 di SNS notifiche Amazon con payload specifici della piattaforma

Puoi utilizzare Amazon AWS Management Console o Amazon SNS APIs per inviare messaggi personalizzati con payload specifici della piattaforma ai dispositivi mobili. Per informazioni sull'utilizzo di Amazon SNSAPIs, consulta APIAzioni push per dispositivi mobili e allega il `SNSMobilePush.java` file`snsmobilepush.zip`.

Argomenti

- Invio JSON di messaggi in formato
- Invio di messaggi specifici della piattaforma
- Invio di messaggi a un'applicazione su più piattaforme
- Invio di messaggi a notifiche APNs di avviso o in background
- Utilizzo dei payload di Google Firebase Cloud Messaging v1 in Amazon SNS

Invio JSON di messaggi in formato

Quando inviate payload specifici per una piattaforma, i dati devono essere formattati come stringhe di coppie JSON chiave-valore, senza virgolette.

Gli esempi seguenti 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 FCM parametri `time_to_live` e `collapse_key` dopo le coppie chiave-valore dei dati personalizzati nel FCM `data` parametro.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Per un elenco delle coppie chiave-valore supportate da ciascuno dei servizi di notifica push supportati in AmazonSNS, consulta quanto segue:

> **Important**
>
> Amazon SNS ora supporta Firebase Cloud Messaging (FCM) HTTP v1 API per l'invio di notifiche push mobili a dispositivi Android.
> 26 marzo 2024: Amazon SNS supporta la versione FCM HTTP 1 API per i dispositivi Apple e le destinazioni Webpush. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima versione FCM HTTP v1 entro il API 1° giugno 2024 per evitare interruzioni delle applicazioni.

- Riferimento chiave di Payload nella documentazione APNs
- Firebase Cloud Messaging HTTP Protocol nella documentazione FCM
- Invia un messaggio nella documentazione ADM

Invio di messaggi a un'applicazione su più piattaforme

Per inviare un messaggio a un'applicazione installata su dispositivi per più piattaforme, ad esempio FCM eAPNs, devi prima sottoscrivere gli endpoint mobili a un argomento in Amazon SNS e poi pubblicare il messaggio sull'argomento.

L'esempio seguente mostra un messaggio da inviare agli endpoint mobili abbonati suAPNs, 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 notifiche APNs di avviso o in background

Amazon SNS può inviare messaggi APNs ad annunci `alert` o `background` notifiche (per ulteriori informazioni, consulta Pushing Background Updates to Your App nella APNs documentazione).

- Una `alert` APNs notifica informa l'utente visualizzando un messaggio di avviso, riproducendo un suono o aggiungendo un badge all'icona dell'applicazione.
- Una `background` APNs notifica si attiva o indica all'applicazione di agire in base al contenuto della notifica, senza informare l'utente.

Specificare i valori di intestazione APNs personalizzati

Ti consigliamo di specificare valori personalizzati per l'attributo del messaggio `AWS.SNS.MOBILE.APNS.PUSH_TYPE` riservato utilizzando l'SNS`Publish`APIazione Amazon o il AWS CLI. AWS SDKs L'CLIesempio seguente imposta `content-available` `1` e imposta `apns-push-type` `background` 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
```

> **Note**
>
> Assicuratevi che la JSON struttura sia valida. Aggiungi una virgola dopo ogni coppia chiave-valore, tranne l'ultima.

Dedurre l'intestazione del tipo di APNs push dal payload

Se non imposti l'`apns-push-type`APNsintestazione, Amazon SNS imposta l'intestazione su `alert` o in `background` base alla `content-available` chiave nel `aps` dizionario della tua configurazione del payload in JSON formatoAPNs.

> **Note**
>
> Amazon SNS è in grado di dedurre solo `alert` le `background` intestazioni, sebbene l'`apns-push-type`intestazione possa 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 non elaborato APNs come notifica solo in background, devi includere `content-available` set to `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'`apns-push-type`APNsintestazione del messaggio su`background`. Quando Amazon SNS rileva che il `apn` dizionario contiene la `content-available` chiave impostata su 1 e non contiene altre chiavi in grado di attivare le interazioni con l'utente, imposta l'intestazione su. `background`

Utilizzo dei payload di Google Firebase Cloud Messaging v1 in Amazon SNS

Amazon SNS supporta l'utilizzo della FCM HTTP versione 1 API 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 Amazon o Amazon SNSAPI. CLI

Puoi includere i seguenti tipi di messaggi nel tuo payload quando invii una FCM notifica:

- 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 JSON oggetto 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
  - APNStasti di notifica
  - chiavi di notifica Webpush

Per ulteriori informazioni sui tipi di FCM messaggi, consulta Tipi di messaggio nella documentazione di Firebase di Google.

Indice

Utilizzo della struttura di payload FCM v1 per inviare messaggi

Se state creando un'FCMapplicazione per la prima volta o desiderate sfruttare le funzionalità della versione FCM 1, potete 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 dalla versione precedente FCM APIs alla HTTP v1 e Personalizzazione di un messaggio tra piattaforme nella documentazione di Google Firebase.

FCMesempio di payload v1 inviato ad Amazon: SNS

> **Note**
>
> Il valore della `GCM` chiave utilizzato nell'esempio seguente deve essere codificato come String quando si pubblica una notifica tramite AmazonSNS.

```
{
  "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 JSON payload, assicurati di includere l'`message-structure`attributo nella richiesta e di impostarlo su. `json`

CLIesempio:

```
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 dalla versione precedente alla v1 FCM APIs HTTP
- Informazioni sui messaggi FCM
- RESTRisorsa: projects.messages

Utilizzo della struttura di payload legacy per inviare messaggi alla FCM v1 API

Durante la migrazione alla FCM v1, non è necessario modificare la struttura del payload utilizzata per le credenziali legacy. 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 i `fcm_options` tasti. Se desideri utilizzare questi campi, invia un payload FCM v1.
- In alcuni casi, le intestazioni dei messaggi sono richieste dalla FCM v1 per inviare notifiche silenziose ai tuoi dispositivi. APNs Se attualmente invii notifiche silenziose ai tuoi APNs dispositivi, queste non funzioneranno con l'approccio precedente. Consigliamo invece di utilizzare il payload FCM v1 per evitare problemi imprevisti. Per trovare un elenco delle APNs intestazioni e per cosa vengono utilizzate, consulta Communicating with APNs nella Apple Developer Guide.
- Se utilizzi l'SNSattributo TTL Amazon per inviare la notifica, questo verrà aggiornato solo nel `android` campo. Se desideri impostare l'TTLAPNSattributo, 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.

FCMeventi di mancata consegna

La tabella seguente fornisce il tipo di SNS errore Amazon che corrisponde ai codici di errore/stato ricevuti da Google per le richieste di notifica FCM v1. Tutti i codici di errore rilevati ricevuti dalla versione FCM v1 API sono disponibili CloudWatch quando configuri la registrazione dello stato di consegna per la tua applicazione.

FCMcodice di errore/stato	Tipo di SNS errore Amazon	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 ha SNS disabilitato il tuo endpoint. Aggiorna l'SNSendpoint Amazon 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'SNSendpoint Amazon al token del dispositivo più recente.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	Il token della piattaforma associato	L'applicazione della piattaforma associata al token del dispositi

FCMcodice di errore/ stato	Tipo di SNS errore Amazon	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 FCM credenziali corrette nell'applicazione della SNS piattaforma Amazon.
UNAVAILABLE	DependencyUnavailable	La dipendenza non è disponibile.	FCMnon è riuscito a elaborare la richiesta in tempo. Tutti i nuovi tentativi eseguiti da Amazon non SNS sono riusciti. Puoi archiviare questi messaggi in una coda di lettere morte (DLQ) e reindirizzarli in un secondo momento.
INTERNAL	UnexpectedFailure	Guasto imprevisto; contatta Amazon. Frase di errore [Errore interno].	Il FCM server ha riscontrato un errore durante il tentativo di elaborare la richiesta . Tutti i nuovi tentativi eseguiti da Amazon non SNS sono riusciti. Puoi archiviare questi messaggi in una coda di lettere morte (DLQ) e reindirizzarli in un secondo momento.

FCMcodice di errore/ stato	Tipo di SNS errore Amazon	Messaggio di errore	Causa e mitigazione
`THIRD_PAR TY_AUTH_ERROR`	`InvalidCr edentials`	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.

FCMcodice di errore/ stato	Tipo di SNS errore Amazon	Messaggio di errore	Causa e mitigazione
`PERMISSIO N_DENIED`	`InvalidNo tification`	Il corpo della notifica non è valido.	In caso di `PERMISSIO N_DENIED` eccezione, il chiamante (l'FCMapp licazione) non è autorizzato a eseguire l'operazione specifica ta nel payload. Accedi alla tua FCM console e verifica che le tue credenziali abbiano abilitato le azioni richieste. API

Attributi SNS delle app mobili Amazon

Amazon Simple Notification Service (AmazonSNS) fornisce supporto per registrare lo stato di consegna dei messaggi di notifica push. Dopo aver configurato gli attributi dell'applicazione, le voci di registro verranno inviate a CloudWatch Logs per i messaggi inviati da Amazon SNS agli endpoint mobili. La registrazione dello stato di consegna dei messaggi consente di ottenere informazioni operative più precise, ad esempio:

- Scopri se un messaggio di notifica push è stato recapitato da Amazon SNS al servizio di notifica push.
- Identifica la risposta inviata dal servizio di notifica push ad AmazonSNS.
- 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 dell'applicazione per lo stato di consegna dei messaggi AWS Management Console, puoi utilizzare i kit di sviluppo AWS software (SDKs) o una queryAPI.

Argomenti

- Configurazione degli attributi dello stato di recapito dei messaggi utilizzando AWS Management Console
- Esempi di CloudWatch log sullo stato di consegna dei SNS messaggi di Amazon
- Configurazione degli attributi dello stato di recapito dei messaggi con AWS SDKs
- Codici di risposta della piattaforma

Configurazione degli attributi dello stato di recapito dei messaggi utilizzando AWS Management Console

1. Accedi alla SNSconsole Amazon.
2. Nel riquadro di navigazione, scegliere Mobile (Dispositivi mobili), Push notifications (Notifiche push).
3. Dalla sezione Applicazioni della piattaforma, scegli l'applicazione che contiene gli endpoint per i quali desideri ricevere i CloudWatch log.
4. Scegli Application Actions (Operazioni applicazione), quindi scegli Delivery status (Stato consegna).
5. Nella finestra di dialogo Delivery Status, scegliete Crea IAM ruoli.

   Verrai quindi reindirizzato alla IAM console.
6. Scegli Consenti per consentire ad Amazon l'accesso in SNS scrittura per utilizzare CloudWatch Logs per tuo conto.
7. Ora, tornando alla finestra di dialogo Delivery Status, inserisci un numero nel campo Percentuale di successo rispetto al campione (0-100) per la percentuale di messaggi inviati con successo per i quali desideri ricevere i log. CloudWatch

   > **Note**
   >
   > Dopo aver configurato gli attributi dell'applicazione per lo stato di consegna dei messaggi, tutti i recapiti dei messaggi non riusciti generano registri. CloudWatch
8. Infine scegli Save Configuration (Salva configurazione). Ora sarai in grado di visualizzare e analizzare i CloudWatch registri contenenti lo stato di consegna dei messaggi. Per ulteriori informazioni sull'utilizzo CloudWatch, consulta la CloudWatch documentazione.

Esempi di CloudWatch log sullo stato di consegna dei SNS messaggi di Amazon

Dopo aver configurato gli attributi dello stato di consegna dei messaggi per un endpoint dell'applicazione, verranno CloudWatch generati i log. I log di esempio, in JSON formato, sono mostrati come segue:

SUCCESS


```
{
  "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"
  }
}
```


FAILURE


```
{
  "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 recapito dei messaggi con AWS SDKs

AWS SDKsForniscono APIs in diverse lingue per l'utilizzo degli attributi dello stato di consegna dei messaggi con AmazonSNS.

Il seguente esempio di Java mostra come utilizzare `SetPlatformApplicationAttributes` API per configurare gli attributi dell'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`. `FailureFeedbackRoleArn`Gli attributi `SuccessFeedbackRoleArn` and vengono utilizzati per consentire ad Amazon l'accesso in SNS scrittura per utilizzare CloudWatch i log per tuo conto. L'attributo `SuccessFeedbackSampleRate` consente di specificare la percentuale della frequenza di campionamento (0-100) dei messaggi consegnati. Dopo aver configurato l'`FailureFeedbackRoleArn`attributo, tutte le consegne di messaggi non riuscite generano CloudWatch log.

```
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 for Java, consulta Getting Started with the. 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 di notifiche push	Codice di risposta
Messaggistica per dispositivi Amazon (ADM)	Vedi Response Format nella ADM documenta zione.
Servizio di notifica push Apple (APNs)	Vedere HTTP/2 Response from APNs in Comunicazione con APNs nella Guida alla programmazione delle notifiche locali e remote.
Firebase Cloud Messaging () FCM	Vedi Codici di risposta a messaggi di errore downstream nella documentazione di Firebase Cloud Messaging.
Servizio Microsoft Push Notification per Windows Phone (MPNS)	Vedi Push Notification Service Response Codes for Windows Phone 8 nella documenta zione di Windows 8 Development.
Servizi di notifica push di Windows (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.

Notifiche degli eventi delle SNS applicazioni Amazon per applicazioni mobili

Amazon SNS fornisce supporto per attivare notifiche quando si verificano determinati eventi dell'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 SNS AWS CLI console Amazon o il AWS SDKs.

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 Using Amazon SNS Application Attributes for Message Delivery Status.

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. Quando Amazon SNS invia le notifiche, l'argomento può indirizzarle agli endpoint che intraprenderanno azioni programmatiche.

> **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 abbonamenti per endpoint programmatici, come HTTPS endpoint HTTP o, code SQS Amazon o funzioni. 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 SNS console Amazon, il AWS Command Line Interface (AWS CLI) o il AWS SDKs.

AWS Management Console

1. Accedi alla SNSconsole Amazon.
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. Inserisci gli ARNs argomenti da utilizzare per i seguenti eventi:
   - Creazione endpoint
   - Eliminazione endpoint
   - Aggiornamento endpoint
   - Errore di consegna
7. Scegli Save changes (Salva modifiche).

AWS CLI

Esegui il comando set-platform-application-attributes.

L'esempio seguente imposta lo stesso SNS argomento Amazon 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 SDKs

Imposta le notifiche degli eventi dell'applicazione inviando una `SetPlatformApplicationAttributes` richiesta ad Amazon SNS API utilizzando un AWS SDK.

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, tra cui assistenza per iniziare e informazioni sulle versioni precedenti, consultaUsare Amazon SNS con un AWS SDK.

APIAzioni push per dispositivi mobili

Per utilizzare Amazon SNS mobile pushAPIs, devi prima soddisfare i prerequisiti per il servizio di notifica push, come Apple Push Notification Service (APNs) e Firebase Cloud Messaging (). FCM Per ulteriori informazioni sui prerequisiti, consulta Prerequisiti per le notifiche agli SNS utenti di Amazon.

Per inviare un messaggio di notifica push a un'app mobile e a un dispositivo utilizzando ilAPIs, devi prima utilizzare l'`CreatePlatformApplication`azione, che restituisce 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 Configurazione delle notifiche push con Amazon SNS.

Le push di Amazon SNS per dispositivi mobili APIs sono le seguenti:

`CreatePlatformApplication`

Crea un oggetto applicativo della piattaforma per uno dei servizi di notifica push supportati, ad esempio APNs andFCM, a cui possono registrarsi dispositivi e app mobili. 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 effettivamente un abbonamento, il proprietario dell'endpoint deve eseguire l' ConfirmSubscription azione con il token contenuto nel messaggio di conferma.

`Unsubscribe`

Elimina una sottoscrizione.

APIErrori push comuni di Amazon SNS mobile

Gli errori restituiti da Amazon SNS APIs for mobile push sono elencati nella tabella seguente. Per ulteriori informazioni su Amazon SNS APIs for mobile push, consultaAPIAzioni push per dispositivi mobili.

Errore	Descrizione	HTTPScodice di stato	APIAzione
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`
Platform Name is invalid (Nome	È stato fornito un out-of-range valore or	400	`CreatePlatformApplication`

Errore	Descrizione	HTTPScodice di stato	APIAzione
piattaforma non valido)	non valido per il nome della piattaforma.		
APNs— Il principale non è un certificato valido	È stato fornito un certificato non valido per il APNs principale, che è il SSL certificato. Per ulteriori informazioni, consulta CreatePlatformApplicationAmazon Simple Notification Service API Reference.	400	`CreatePlatformApplication`
APNs— Principal è un certificato valido ma non in formato.pem	Per il APNs principale, che è il certificato, è stato fornito un certificato valido che non è in formato pem. SSL	400	`CreatePlatformApplication`
APNs— Il principale è un certificato scaduto	È stato fornito un certificato scaduto per il APNs principale, che è il SSL certificato.	400	`CreatePlatformApplication`
APNs— Il principale non è un certificato emesso da Apple	È stato fornito un certificato non rilasciato da Apple per il APNs principale, che è il SSL certificato.	400	`CreatePlatformApplication`
APNs— Il capitale non è fornito	Il APNs principale, che è il SSL certificato, non è stato fornito.	400	`CreatePlatformApplication`

Errore	Descrizione	HTTPScodice di stato	APIAzione
APNs— La credenziale non è stata fornita	La APNs credenziale, che è la chiave privata, non è stata fornita. Per ulteriori informazioni, consulta CreatePlatformApplicationAmazon Simple Notification Service API Reference.	400	`CreatePlatformApplication`
APNs— Le credenziali non sono in un formato .pem valido	La APNs credenziale, che è la chiave privata, non è in un formato .pem valido.	400	`CreatePlatformApplication`
FCM— serverAPIKey non è fornito	La FCM credenziale, che è la API chiave, non è stata fornita. Per ulteriori informazioni, consulta CreatePlatformApplicationAmazon Simple Notification Service API Reference.	400	`CreatePlatformApplication`
FCM— serverAPIKey è vuoto	La FCM credenziale, che è la API chiave, è vuota.	400	`CreatePlatformApplication`
FCM— serverAPIKey è una stringa nulla	La FCM credenziale, che è la API chiave, è nulla.	400	`CreatePlatformApplication`
FCM— serverAPIKey non è valido	La FCM credenziale, che è la API chiave, non è valida.	400	`CreatePlatformApplication`

Errore	Descrizione	HTTPScodice di stato	APIAzione
ADM— clientsecret non è fornito	Il segreto client richiesto non è fornito.	400	`CreatePlatformApplication`
ADM— clientsecret è una stringa nulla	La stringa richiesta per il segreto client è null.	400	`CreatePlatformApplication`
ADM— client_secret è una stringa vuota	La stringa richiesta per il segreto client è vuota.	400	`CreatePlatformApplication`
ADM— client_secret non è valido	La stringa richiesta per il segreto client non è valida.	400	`CreatePlatformApplication`
ADM— client_id è una stringa vuota	La stringa richiesta per l'ID client è vuota.	400	`CreatePlatformApplication`
ADM— non clientId è fornito	La stringa richiesta per l'ID client non è fornita.	400	`CreatePlatformApplication`
ADM— clientid è una stringa nulla	La stringa richiesta per l'ID client è null.	400	`CreatePlatformApplication`
ADM— client_id non è valido	La stringa richiesta per l'ID client non è valida.	400	`CreatePlatformApplication`
EventEndpointCreated ha un formato non valido ARN	EventEndpointCreated ha un formato non validoARN.	400	`CreatePlatformApplication`

Errore	Descrizione	HTTPScodice di stato	APIAzione
EventEndpointDeleted ha un formato non valido ARN	EventEndpointDeleted ha un formato non validoARN.	400	`CreatePlatformApplication`
EventEndpointUpdated ha un formato non valido ARN	EventEndpointUpdated ha un formato non validoARN.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure ha un formato non valido ARN	EventDeliveryAttemptFailure ha un formato non validoARN.	400	`CreatePlatformApplication`
EventDeliveryFailure ha un formato non valido ARN	EventDeliveryFailure ha un formato non validoARN.	400	`CreatePlatformApplication`
EventEndpointCreated non è un argomento esistente	EventEndpointCreated non è un argomento esistente.	400	`CreatePlatformApplication`
EventEndpointDeleted non è un argomento esistente	EventEndpointDeleted non è un argomento esistente.	400	`CreatePlatformApplication`
EventEndpointUpdated non è un argomento esistente	EventEndpointUpdated non è un argomento esistente.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure non è un argomento esistente	EventDeliveryAttemptFailure non è un argomento esistente.	400	`CreatePlatformApplication`
EventDeliveryFailure non è un argomento esistente	EventDeliveryFailure non è un argomento esistente.	400	`CreatePlatformApplication`

Errore	Descrizione	HTTPScodice di stato	APIAzione
La piattaforma ARN non è valida	La piattaforma ARN non è valida.	400	`SetPlatformAttributes`
La piattaforma ARN è valida ma non appartiene all'utente	ARNLa piattaforma è valida ma non appartiene all'utente.	400	`SetPlatformAttributes`
APNs— Il principale non è un certificato valido	È stato fornito un certificato non valido per il APNs principale, che è il SSL certificato. Per ulteriori informazioni, consulta CreatePlatformApplicationAmazon Simple Notification Service API Reference.	400	`SetPlatformAttributes`
APNs— Principal è un certificato valido ma non in formato.pem	Per il APNs principale, che è il certificato, è stato fornito un certificato valido che non è in formato pem. SSL	400	`SetPlatformAttributes`
APNs— Il principale è un certificato scaduto	È stato fornito un certificato scaduto per il APNs principale, che è il SSL certificato.	400	`SetPlatformAttributes`
APNs— Il principale non è un certificato emesso da Apple	È stato fornito un certificato non rilasciato da Apple per il APNs principale, che è il SSL certificato.	400	`SetPlatformAttributes`

Errore	Descrizione	HTTPScodice di stato	APIAzione
APNs— Il capitale non è fornito	Il APNs principale, che è il SSL certifica to, non è stato fornito.	400	`SetPlatfo rmAttributes`
APNs— La credenzia le non è stata fornita	La APNs credenzia le, che è la chiave privata, non è stata fornita. Per ulteriori informazioni, consulta CreatePlatformAppl icationAmazon Simple Notification Service API Reference.	400	`SetPlatfo rmAttributes`
APNs— Le credenzia li non sono in un formato .pem valido	La APNs credenzia le, che è la chiave privata, non è in un formato .pem valido.	400	`SetPlatfo rmAttributes`
FCM— serverAPIKey non è fornito	La FCM credenzia le, che è la API chiave, non è stata fornita. Per ulteriori informazioni, consulta CreatePlatformAppl icationAmazon Simple Notification Service API Reference.	400	`SetPlatfo rmAttributes`
FCM— serverAPIKey è una stringa nulla	La FCM credenziale, che è la API chiave, è nulla.	400	`SetPlatfo rmAttributes`
ADM— non clientId è fornito	La stringa richiesta per l'ID client non è fornita.	400	`SetPlatfo rmAttributes`

Errore	Descrizione	HTTPScodice di stato	APIAzione
ADM— clientid è una stringa nulla	La stringa richiesta per l'ID client è null.	400	`SetPlatformAttributes`
ADM— clientsecret non è fornito	Il segreto client richiesto non è fornito.	400	`SetPlatformAttributes`
ADM— clientsecret è una stringa nulla	La stringa richiesta per il segreto client è null.	400	`SetPlatformAttributes`
EventEndpointUpdated ha un formato non valido ARN	EventEndpointUpdated ha un formato non validoARN.	400	`SetPlatformAttributes`
EventEndpointDeleted ha un formato non valido ARN	EventEndpointDeleted ha un formato non validoARN.	400	`SetPlatformAttributes`
EventEndpointUpdated ha un formato non valido ARN	EventEndpointUpdated ha un formato non validoARN.	400	`SetPlatformAttributes`
EventDeliveryAttemptFailure ha un formato non valido ARN	EventDeliveryAttemptFailure ha un formato non validoARN.	400	`SetPlatformAttributes`
EventDeliveryFailure ha un formato non valido ARN	EventDeliveryFailure ha un formato non validoARN.	400	`SetPlatformAttributes`
EventEndpointCreated non è un argomento esistente	EventEndpointCreated non è un argomento esistente.	400	`SetPlatformAttributes`

Errore	Descrizione	HTTPScodice di stato	APIAzione
EventEndpointDeleted non è un argomento esistente	EventEndpointDeleted non è un argomento esistente.	400	SetPlatformAttributes
EventEndpointUpdated non è un argomento esistente	EventEndpointUpdated non è un argomento esistente.	400	SetPlatformAttributes
EventDeliveryAttemptFailure non è un argomento esistente	EventDeliveryAttemptFailure non è un argomento esistente.	400	SetPlatformAttributes
EventDeliveryFailure non è un argomento esistente	EventDeliveryFailure non è un argomento esistente.	400	SetPlatformAttributes
La piattaforma ARN non è valida	La piattaforma ARN non è valida.	400	GetPlatformApplicationAttributes
La piattaforma ARN è valida ma non appartiene all'utente	La piattaforma ARN è valida, ma non appartiene all'utente.	403	GetPlatformApplicationAttributes
Token specified is invalid (Token specificato non valido)	Il token specificato non è valido.	400	ListPlatformApplications
La piattaforma ARN non è valida	La piattaforma ARN non è valida.	400	ListEndpointsByPlatformApplication

Errore	Descrizione	HTTPScodice di stato	APIAzione
La piattaforma ARN è valida ma non appartiene all'utente	La piattaforma ARN è valida, ma non appartiene all'utente.	404	`ListEndpointsByPlatformApplication`
Token specified is invalid (Token specificato non valido)	Il token specificato non è valido.	400	`ListEndpointsByPlatformApplication`
La piattaforma ARN non è valida	La piattaforma ARN non è valida.	400	`DeletePlatformApplication`
La piattaforma ARN è valida ma non appartiene all'utente	La piattaforma ARN è valida, ma non appartiene all'utente.	403	`DeletePlatformApplication`
La piattaforma ARN non è valida	La piattaforma ARN non è valida.	400	`CreatePlatformEndpoint`
La piattaforma ARN è valida ma non appartiene all'utente	La piattaforma ARN è valida, 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`

Errore	Descrizione	HTTPScodice di stato	APIAzione
Customer User data is too large (Dimensione dati utente cliente troppo grande)	I dati utente del cliente non possono superare i 2048 byte con la codifica UTF-8.	400	`CreatePlatformEndpoint`
L'endpoint non è valido ARN	L'endpoint ARN non è valido.	400	`DeleteEndpoint`
L'endpoint ARN è valido ma non appartiene all'utente	L'endpoint ARN è valido, ma non appartiene all'utente.	403	`DeleteEndpoint`
L'endpoint ARN non è valido	L'endpoint ARN non è valido.	400	`SetEndpointAttributes`
L'endpoint ARN è valido ma non appartiene all'utente	L'endpoint ARN 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)	I dati utente del cliente non possono superare i 2048 byte con la codifica -8. UTF	400	`SetEndpointAttributes`
L'endpoint non è valido ARN	L'endpoint ARN non è valido.	400	`GetEndpointAttributes`

Errore	Descrizione	HTTPScodice di stato	APIAzione
L'endpoint ARN è valido ma non appartiene all'utente	L'endpoint ARN è valido, ma non appartiene all'utente.	403	`GetEndpointAttributes`
L'obiettivo ARN non è valido	L'obiettivo ARN non è valido.	400	`Publish`
ARNLa destinazione è valida ma non appartiene all'utente	L'obiettivo ARN 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 Amazon SNS time to live message per le notifiche push mobili

Amazon Simple Notification Service (AmazonSNS) fornisce supporto per l'impostazione di un attributo di messaggio Time To Live (TTL) per i messaggi di notifica push per dispositivi mobili. Questa funzionalità si aggiunge alla funzionalità esistente di impostazione TTL all'interno del corpo del SNS messaggio Amazon per i servizi di notifica push mobili che la supportano, come Amazon Device Messaging (ADM) e Firebase Cloud Messaging (FCM) per l'invio ad Android.

L'attributo TTL message viene utilizzato per specificare i metadati di scadenza relativi a un messaggio. Ciò consente di specificare il periodo di tempo a disposizione del servizio di notifica push, ad esempio Apple Push Notification Service (APNs) orFCM, per recapitare il messaggio all'endpoint. Se per qualche motivo (ad esempio, il dispositivo mobile è spento) il messaggio non è recapitabile entro i limiti specificatiTTL, il messaggio verrà eliminato e non verranno effettuati ulteriori tentativi di recapito. Per specificare TTL gli attributi del messaggio, è possibile utilizzare i AWS Management Console kit di sviluppo AWS software (SDKs) o una query. API

Argomenti

TTLattributi dei messaggi per i servizi di notifica push

Di seguito è riportato un elenco degli attributi dei TTL messaggi per i servizi di notifica push che è possibile utilizzare per impostare quando si utilizza la query AWS SDKs orAPI:

Servizio di notifiche push	TTLattributo del messaggio
Messaggistica per dispositivi Amazon (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Servizio di notifica push Apple (APNs)	`AWS.SNS.MOBILE.APNS.TTL`
Sandbox del servizio di notifica push Apple (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCMquando si invia ad Android)	`AWS.SNS.MOBILE.FCM.TTL`
Servizi di notifica push di Windows () WNS	`AWS.SNS.MOBILE.WNS.TTL`

Ciascuno dei servizi di notifica push funziona in TTL modo diverso. Amazon SNS fornisce una visione astratta TTL di tutti i servizi di notifica push, il che semplifica la specificazioneTTL. Quando utilizzi il AWS Management Console to specific TTL (in secondi), devi inserire il TTL valore una sola volta

e Amazon SNS calcolerà quindi il valore TTL per ciascuno dei servizi di notifica push selezionati al momento della pubblicazione del messaggio.

TTLè relativo all'ora di pubblicazione. Prima di inviare un messaggio di notifica push a uno specifico servizio di notifica push, Amazon SNS calcola il tempo di permanenza (il tempo che intercorre tra il timestamp di pubblicazione e il momento prima della consegna a un servizio di notifica push) per la notifica push e trasmette il resto TTL al servizio di notifica push specifico. Se TTL è inferiore al tempo di permanenza, Amazon SNS non tenterà di pubblicare.

Se specifichi un TTL per un messaggio di notifica push, il TTL valore 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 APNs e FCM (quando si invia ad Android). Se il TTL valore è impostato su `0` e il servizio di notifica push non ha un significato specifico per`0`, Amazon SNS rilascerà il messaggio. Per ulteriori informazioni sul TTL parametro impostato su `0` when useAPNs, consulta la Tabella A-3 Item identifiers for remote notification nella documentazione di Binary Provider API.

Ordine di precedenza per la determinazione TTL

La precedenza SNS utilizzata da Amazon TTL per determinare la priorità di un messaggio di notifica push si basa sul seguente ordine, in cui il numero più basso ha la priorità più alta:

1. Attributo del messaggio TTL
2. Corpo del messaggio TTL
3. Servizio di notifica push predefinito TTL (varia in base al servizio)
4. Amazon SNS default TTL (4 settimane)

Se TTL imposti valori diversi (uno negli attributi del messaggio e un altro nel corpo del messaggio) per lo stesso messaggio, Amazon SNS modificherà i valori nel corpo del messaggio TTL in modo che corrispondano a quelli TTL specificati nell'attributo del messaggio.

Specificare TTL utilizzando il AWS Management Console

1. Accedi alla SNSconsole Amazon.
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. Sul ***MyApplication*** nella sezione Endpoints, scegli un endpoint dell'applicazione, quindi scegli Pubblica messaggio.

5. Nella sezione Dettagli del messaggio, inserisci TTL (il numero di secondi a disposizione del servizio di notifica push per recapitare il messaggio all'endpoint).
6. Seleziona Publish message (Pubblica messaggio).

Regioni supportate dalle applicazioni SNS mobili Amazon

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 (UAE)
- Sud America (San Paolo)
- AWS GovCloud (Stati Uniti occidentali)

Le migliori pratiche per la gestione delle notifiche push di Amazon per SNS dispositivi mobili

Questa sezione illustra diverse best practice che potrebbero rivelarsi utili per migliorare il coinvolgimento dei clienti.

Gestione di endpoint

Potrebbero verificarsi problemi di consegna in situazioni in cui i token del dispositivo cambiano a causa di un'azione dell'utente sul dispositivo (ad esempio, un'app viene reinstallata sul dispositivo) o gli aggiornamenti dei certificati riguardano i dispositivi che eseguono una particolare versione di iOS. La registrazione a APNs ogni avvio dell'app è una best practice consigliata da Apple.

Poiché il token del dispositivo non cambia ogni volta che un'app viene aperta da un utente, è possibile utilizzare l'idempotente. `CreatePlatformEndpoint`API 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. SNSGestione Amazon degli endpoint Firebase Cloud Messaging

Registrazione dello stato della consegna

Per monitorare lo stato di consegna delle notifiche push, ti consigliamo di abilitare la registrazione dello stato della spedizione per la tua applicazione SNS della piattaforma Amazon. 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'attivazione della registrazione dello stato della spedizione, consulta Come posso accedere ai log di consegna per SNS argomenti di Amazon 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'SNSargomento Amazon configurato di evitare eventi agli abbonati, ad esempio una funzione Lambda, per gli eventi applicativi della piattaforma relativi alla creazione, all'eliminazione, agli aggiornamenti e agli errori di consegna degli endpoint.

Configurazione e gestione dell'abbonamento SNS e-mail Amazon

Puoi iscrivere un indirizzo e-mail a un SNS argomento di Amazon utilizzando AWS Management Console AWS SDK for Java, o AWS SDK for .NET.

> **Note**
>
> - La personalizzazione del corpo del messaggio e-mail non è supportata. 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 limitata. Per ulteriori informazioni, consulta le SNSquote Amazon.

> **Important**
>
> Per impedire ai destinatari della mailing list di annullare l'iscrizione a tutti i destinatari delle e-mail SNS tematiche di Amazon, consulta Configurare un abbonamento e-mail che richieda l'autenticazione per annullare l'iscrizione a Support. AWS

Sottoscrizione di un indirizzo e-mail a un SNS argomento di Amazon utilizzando il AWS Management Console

1. Accedi alla SNSconsole Amazon.
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.
   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 Politiche di filtro degli SNS abbonamenti Amazon.

e. (Facoltativo) Per abilitare il filtro basato sul payload, imposta `Filter Policy Scope` su `MessageBody`. Per ulteriori informazioni, consulta Ambito della politica di filtro degli SNS abbonamenti Amazon.

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

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 abbonamento nell'e-mail di AmazonSNS.
2. Amazon SNS apre il browser Web e visualizza una conferma dell'abbonamento con l'ID dell'abbonamento.

Sottoscrizione di un indirizzo e-mail a un SNS argomento di Amazon utilizzando un AWS SDK

Per usare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta I file di configurazione e credenziali condivisi nella AWS SDKs and Tools Reference Guide.

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

.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.

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 API i dettagli, consulta Subscribe in AWS SDK for .NET APIReference.

C++

SDKper 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 SQS coda 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 API i dettagli, consulta Subscribe in AWS SDK for C++ APIReference.

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 API i dettagli, consulta Subscribe in AWS CLI Command Reference.

Go

SDKper 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.

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(ctx context.Context, 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(ctx, &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 API i dettagli, consulta Subscribe in AWS SDK for Go APIReference.

Java

SDKper 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.

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 HTTP 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 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 API i dettagli, consulta Subscribe in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 SQS coda 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 API i dettagli, consulta Subscribe in AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

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

Sottoscrivi un indirizzo email a un argomento.

```
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 API i dettagli, consulta Subscribe in AWS SDKfor Kotlin reference API.

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.

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 HTTP endpoint 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 API i dettagli, consulta Subscribe in AWS SDK for PHP APIReference.

Python

SDKper Python (Boto3)

> **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.

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 API i dettagli, consulta Subscribe in AWS SDKfor Python (Boto3) Reference. API

Ruby

SDKper Ruby

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

Sottoscrivi un indirizzo email a un argomento.

```
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 API i dettagli, consulta Subscribe in AWS SDK for Ruby APIReference.

Rust

SDKper Rust

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

Sottoscrivi un indirizzo email a un argomento.

```
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 API i dettagli, consulta Subscribe in AWS SDKfor Rust API reference.

SAP ABAP

SDKper SAP ABAP

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

Sottoscrivi un indirizzo email a un argomento.

```
    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 API i dettagli, vedi Iscriviti SAP ABAPAPIcome riferimento.AWS SDK

SNSLe migliori pratiche di Amazon

Di seguito sono riportate le best practice consigliate per l'utilizzo di AmazonSNS:

Argomenti

Best practice SNS di sicurezza di Amazon

AWS offre molte funzionalità di sicurezza per AmazonSNS. 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 sicurezza preventiva per AmazonSNS.

Argomenti

Assicurarsi che gli argomenti non siano accessibili pubblicamente

A meno che tu non richieda esplicitamente a chiunque su Internet di essere in grado di leggere o scrivere sul tuo SNS argomento Amazon, dovresti assicurarti che l'argomento non sia accessibile al pubblico (accessibile da tutti nel mondo o da qualsiasi AWS utente 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 concedi le autorizzazioni, decidi chi le riceve, a quali argomenti sono destinate le autorizzazioni e API le azioni specifiche che desideri consentire per questi 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 editore-abbonato, che richiede tre tipi di accesso all'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:

- Gestione delle identità e degli accessi in Amazon SNS
- SNSAPIAutorizzazioni Amazon: riferimento ad azioni e risorse

Usa IAM i ruoli per applicazioni e AWS servizi che richiedono SNS l'accesso ad Amazon

Affinché applicazioni o AWS servizi, come AmazonEC2, possano accedere agli SNS argomenti di Amazon, devono utilizzare AWS credenziali valide nelle loro AWS API richieste. Poiché queste

credenziali non vengono ruotate automaticamente, non dovresti archiviarle direttamente nell' AWS applicazione o nell'istanza. EC2

È necessario utilizzare un IAM ruolo per gestire le credenziali temporanee per le applicazioni o i servizi che devono accedere ad AmazonSNS. Quando utilizzi un ruolo, non è necessario distribuire credenziali a lungo termine (come nome utente, password e chiavi di accesso) a un'EC2istanza o AWS servizio, ad esempio. AWS Lambda Al contrario, il ruolo fornisce autorizzazioni temporanee che le applicazioni possono utilizzare quando effettuano chiamate ad altre AWS risorse.

Per ulteriori informazioni, consulta IAMRuoli e scenari comuni per i ruoli: utenti, applicazioni e servizi nella Guida per l'IAMutente.

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 crittografa i tuoi dati a livello di messaggio quando li archivia e decrittografa i messaggi per te quando accedi ad essi. SSEutilizza 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 Protezione dei SNS dati Amazon con la crittografia lato server e Gestione delle chiavi e dei costi di SNS crittografia Amazon.

Applica la crittografia dei dati in transito

È possibile, ma non consigliato, pubblicare messaggi non crittografati durante il transito utilizzandoHTTP. Tuttavia, quando un argomento viene crittografato in inattivo utilizzando AWS KMS, è necessario utilizzarlo HTTPS per la pubblicazione dei messaggi per garantire la crittografia sia a riposo che in transito. Sebbene l'argomento non rifiuti automaticamente i HTTP messaggi, HTTPS è necessario utilizzarlo per mantenere gli standard di sicurezza.

AWS consiglia di utilizzare HTTPS al posto di. HTTP Quando si utilizzaHTTPS, i messaggi vengono crittografati automaticamente durante il transito, anche se l'SNSargomento stesso non è crittografato. HTTPSIn caso contrario, un utente malintenzionato basato sulla rete può intercettare il traffico di rete o manipolarlo utilizzando un attacco come. man-in-the-middle

Per applicare solo le connessioni crittografateHTTPS, aggiungi la `aws:SecureTransport`condizione nella policy allegata agli argomenti non crittografatiIAM.

SNS Ciò obbliga gli editori di messaggi a utilizzare invece di. HTTPS 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": "*"
    }
  ]
}
```

Prendi in considerazione l'utilizzo VPC degli endpoint per accedere ad Amazon SNS

Se hai argomenti con cui devi essere in grado di interagire, ma questi argomenti non devono assolutamente essere esposti a Internet, utilizza gli VPC endpoint per limitare l'accesso agli argomenti solo agli host all'interno di un determinato VPC argomento. Puoi utilizzare le policy tematiche per controllare l'accesso agli argomenti da VPC endpoint Amazon specifici o da specificiVPCs.

SNSVPCGli endpoint Amazon offrono due modi per controllare l'accesso ai tuoi messaggi:

- Puoi controllare le richieste, gli utenti o i gruppi consentiti tramite un VPC endpoint specifico.
- Puoi controllare quali VPCs o quali VPC endpoint hanno accesso al tuo argomento utilizzando una policy tematica.

Per ulteriori informazioni, consulta Creazione dell'endpoint e Creazione di una policy VPC sugli endpoint 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`.

SNSSMSLe migliori pratiche di Amazon

> **Important**
>
> L'Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS ha integrato un sistema AWS End User Messaging SMSper la consegna dei SMS messaggi. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i tuoi SNS SMS messaggi Amazon.

Gli utenti di telefoni cellulari tendono ad avere una tolleranza molto bassa per i messaggi indesideratiSMS. I tassi di risposta per SMS le campagne non richieste saranno quasi sempre bassi e quindi il ritorno sull'investimento sarà scarso.

Inoltre, gli operatori di telefonia mobile controllano continuamente i mittenti in blocco. SMS 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 SNS team di Amazon controlla regolarmente le SMS campagne e potrebbe limitare o bloccare la tua capacità di inviare messaggi se sembra che tu stia inviando messaggi non richiesti.

Infine, in molti paesi, regioni e giurisdizioni, sono previste severe sanzioni per l'invio di messaggi indesiderati. SMS Ad esempio, negli Stati Uniti, il Telephone Consumer Protection Act (TCPA) stabilisce che i consumatori hanno diritto a un risarcimento danni compreso tra 500 e 1.500 dollari (a carico del mittente) per ogni messaggio indesiderato 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

La violazione di leggi e normative in vigore nei luoghi di residenza dei clienti può comportare multe e sanzioni elevate. Per questo motivo, è fondamentale comprendere le leggi relative alla SMS messaggistica in ogni paese o area geografica in cui si opera.

L'elenco seguente include collegamenti alle leggi chiave che si applicano alle SMS comunicazioni nei principali mercati di tutto il mondo.

- Stati Uniti: il Telephone Consumer Protection Act del 1991, noto anche comeTCPA, si applica a determinati tipi di SMS messaggi. Per ulteriori informazioni, consulta le regole e le normative nel sito Web della Federal Communications Commission.

- Regno Unito: i regolamenti sulla privacy e le comunicazioni elettroniche (direttiva CE) del 2003, noti anche comePECR, si applicano a determinati tipi di SMS messaggi. Per ulteriori informazioni, consulta Cosa sonoPECR? sul sito web dell'Ufficio del Commissario per l'informazione del Regno Unito.
- Unione europea: la Direttiva sulla privacy e le comunicazioni elettroniche del 2002, a volte nota come ePrivacy Direttiva, si applica ad alcuni tipi di SMS messaggi. Per ulteriori informazioni, consulta il testo integrale della legge nel sito Web europa.eu.
- Canada: il Fighting Internet and Wireless Spam Act, più comunemente noto come Legge anti-spam canadese oCASL, si applica a determinati tipi di SMS messaggi. Per ulteriori informazioni, consulta il testo integrale della legge nel sito Web del parlamento canadese.
- Giappone: la legge sulla regolamentazione della trasmissione di posta elettronica specifica può applicarsi a determinati tipi di SMS messaggi. 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 di questo elenco sono state originariamente create per affrontare le e-mail o le telefonate indesiderate, ma sono state interpretate o ampliate per applicarsi anche ai SMS messaggi. Altri paesi e regioni potrebbero avere leggi proprie relative 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 imporre restrizioni sui SMS contenuti che superano i 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 di SMS attivazione, invia al destinatario un messaggio in cui gli chiedi di confermare che desidera ricevere messaggi da te. 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, gli operatori di telefonia mobile richiedono di fornire modelli (mockup) o schermate dell'intero flusso di lavoro di opt-in. 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

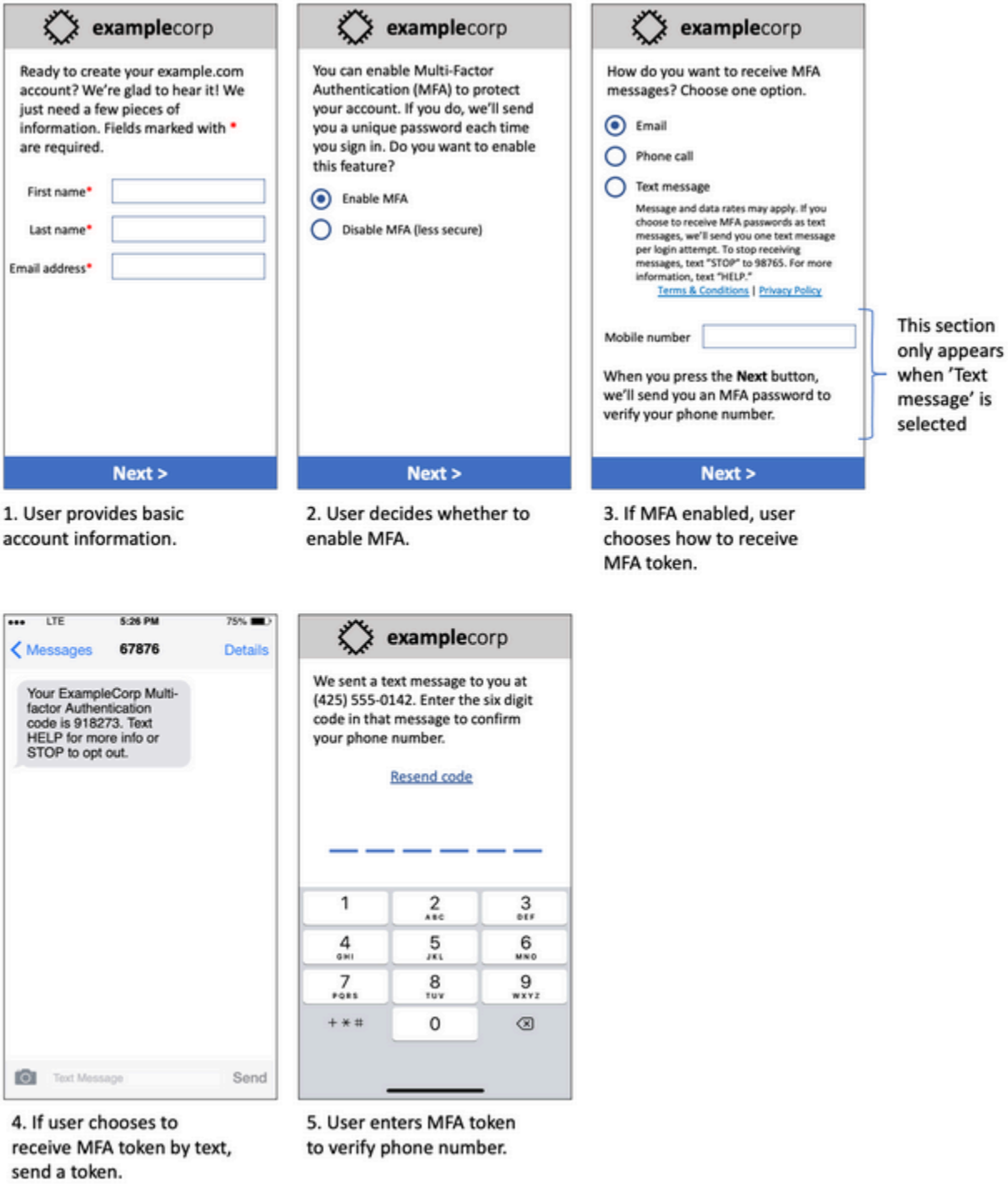
- Una descrizione del caso d'uso relativo alla messaggistica che invierai tramite il programma.
- La frase "È possibile che vengano applicate tariffe specifiche per messaggi e dati".
- Un'indicazione della frequenza con cui i destinatari riceveranno messaggi. Ad esempio, un programma di messaggistica 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 di un caso d'uso dell'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 SMS campagne ricorrenti, controlla regolarmente gli elenchi dei tuoi 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 registri che mostrino quando ogni cliente ha richiesto di ricevere SMS messaggi da te e quali messaggi hai inviato a ciascun cliente. Molti paesi e regioni in tutto il mondo richiedono

ai SMS mittenti di conservare questi record in un modo che possa essere facilmente recuperato. 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 dei registri, consulta le normative sulla SMS messaggistica commerciale in ogni paese o area geografica in cui risiedono 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 queste situazioni, ti AWS Support contatta fornendoti un elenco delle informazioni richieste dal corriere o dall'agenzia. Se non puoi fornire le informazioni necessarie, potremmo sospendere la tua capacità di inviare SMS messaggi aggiuntivi.

Rendi i tuoi messaggi chiari, attendibili e concisi

SMSè un mezzo unico. Il character-per-message limite di 160 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 SMS canale e potrebbero persino sembrare disoneste 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 del messaggio efficaceSMS.

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 person-to-person messaggio

Alcuni esperti di marketing sono tentati di aggiungere un tocco personale ai propri SMS messaggi facendoli apparire come se provenissero da una persona. 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 di caratteri standard (noto come GSM alfabeto) che può essere incluso in un messaggio di 160 SMS 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 SMS caratteri, vedere. SMSlimiti di caratteri 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 quasi identici, ma il primo esempio contiene un simbolo di 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, SMS i messaggi molto lunghi URLs potrebbero essere suddivisi in più messaggi. È necessario utilizzare i domini di reindirizzamento per fornire informazioni abbreviate. URLs 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

Il limite di 160 caratteri del SMS canale 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. Ad esempio, quando un cliente risponde a uno dei tuoi messaggi con la parola chiave "HELP«, inviagli informazioni sul programma a cui è iscritto, sul numero di messaggi che invierai ogni mese e sui modi in cui può contattarti per ulteriori informazioni. Una HELP risposta potrebbe essere simile al seguente esempio:

```
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«, fagli sapere che non riceverà altri messaggi. Una STOP risposta potrebbe essere simile al seguente esempio:

```
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. Inoltre, non ha senso inviare SMS messaggi quando i 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

Nelle tue campagne, se utilizzi più canali di comunicazione (come e-mail e messaggi push), non inviare lo stesso messaggio su tutti i canali. SMS 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 con cui SMS inviare messaggi AWS End User Messaging SMS nella Guida per l'AWS End User Messaging SMS utente.

Verifica i numeri di telefono di destinazione

Quando SMS invii messaggi tramite AmazonSNS, ti viene addebitata una fattura per ogni parte del messaggio che invii. Il prezzo da pagare per parte di messaggio varia in base al Paese o all'area geografica del destinatario. Per ulteriori informazioni sui SMS prezzi, consulta la pagina SMSPrezzi AWS mondiali.

Quando Amazon SNS accetta una richiesta di invio di un SMS messaggio (come risultato di una chiamata o come risultato del SendMessagesAPIlancio di una campagna o di un viaggio), ti viene addebitato un costo per l'invio di quel 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 SMS messaggi e tenta di recapitarle. Per questo motivo, è necessario verificare che i numeri di telefono a cui si inviano messaggi siano numeri di cellulare validi. È possibile utilizzare AWS End User Messaging SMS per inviare un messaggio di prova per determinare se un numero di telefono è valido e di che tipo di numero si tratta (ad esempio cellulare, rete fissa o VoIP). Per ulteriori informazioni, consulta Inviare un messaggio di prova con il SMS simulatore nella Guida per l'AWS End User Messaging SMS utente.

Progetta considerando la ridondanza

Per i programmi di messaggistica mission critical, ti consigliamo di configurare Amazon SNS in più di uno. Regione AWS Amazon SNS è disponibile in diversi formati Regioni AWS. Per un elenco completo delle regioni in cui Amazon SNS è disponibile, consulta la Riferimenti generali di AWS.

I numeri di telefono che utilizzi per i SMS messaggi, inclusi codici brevi, codici lunghi, numeri verdi e 10 DLC numeri, non possono essere replicati. Regioni AWS Di conseguenza, per utilizzare Amazon SNS in più regioni, devi richiedere numeri di telefono separati in ogni regione in cui desideri utilizzare AmazonSNS. Ad esempio, se utilizzi un codice breve per inviare messaggi di testo a destinatari negli Stati Uniti, devi richiedere codici brevi separati per ciascuno di essi Regione AWS che intendi 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, 10 DLC numeri e numeri verdi. 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, contemporaneamente Regione AWS o distribuiti su più numeri, Regioni AWS offre un ulteriore livello di ridondanza, che può contribuire a migliorare la resilienza.

SMSlimiti e restrizioni

Per SMS limiti e restrizioni, consulta la sezione relativa MMSai limiti SMS e alle restrizioni nella Guida per l'AWS End User Messaging SMS utente.

Gestione delle parole chiave di esclusione

SMSi destinatari possono utilizzare i propri dispositivi per disattivare i messaggi rispondendo con una parola chiave. Per ulteriori informazioni, consulta Disattivazione della ricezione di messaggi SMS.

CreatePool

Utilizza l'`CreatePool`APIazione per creare un nuovo pool e associare un'identità di origine specificata al pool. Per ulteriori informazioni, vedere CreatePoolin AWS End User Messaging SMS APIReference.

PutKeyword

Utilizza l'`PutKeyword`APIazione per creare o aggiornare una configurazione di parole chiave su un numero di telefono o un pool di origine. Per ulteriori informazioni, vedere PutKeywordin AWS End User Messaging SMS APIReference.

Gestione delle impostazioni del numero

Per gestire le impostazioni per i codici brevi e i codici lunghi dedicati che hai richiesto al AWS Supporto e assegnati al tuo account, vedi Modificare le funzionalità di un numero di telefono con l' AWS CLIin AWS End User Messaging SMS.

SMSlimiti di caratteri in Amazon SNS

Un singolo SMS messaggio può contenere fino a 140 byte di informazioni. Il numero di caratteri che è possibile includere in un singolo SMS messaggio dipende dal tipo di caratteri contenuti nel messaggio.

Se il messaggio utilizza solo caratteri nel set di caratteri GSM 03.38, noto anche come alfabeto a GSM 7 bit, può contenere fino a 160 caratteri. Se il messaggio contiene caratteri che non rientrano nel set di caratteri GSM 03.38, può contenere fino a 70 caratteri. Quando invii un SMS messaggio, Amazon determina SNS 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 in ogni parte del messaggio viene ridotto a 153 (per i messaggi che contengono solo GSM 03,38 caratteri) o 67 (per i messaggi che contengono altri caratteri). Puoi stimare quante parti del messaggio contiene il messaggio prima di inviarlo utilizzando gli strumenti di calcolo della SMS lunghezza, molti dei quali sono disponibili online. La dimensione massima supportata per ogni messaggio è di 1600 GSM caratteri o 630 caratteri diversi. GSM Per ulteriori informazioni sulla velocità effettiva e sulla dimensione dei messaggi, consulta i limiti di SMS caratteri 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 lo fai, Amazon SNS genera un `_SMS.SUCCESS` evento quando il messaggio viene recapitato all'operatore di telefonia mobile 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.

GSM03.38 set di caratteri

La tabella seguente elenca 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.

GSM03.38 caratteri standard												
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
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						

Il set di caratteri GSM 03.38 include diversi simboli oltre a quelli mostrati nella tabella precedente. Ognuno di essi, tuttavia, viene conteggiato come due caratteri poiché include anche un carattere di escape non visibile:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Infine, il set di caratteri GSM 03.38 include anche i seguenti caratteri non stampati:

- 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 diversi SMS messaggi di esempio. Per ogni esempio, questa sezione mostra il numero totale di caratteri e il numero di parti del messaggio.

Esempio 1: un messaggio lungo che contiene solo caratteri dell'alfabeto GSM 03.38

Il messaggio seguente contiene solo caratteri 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 suddiviso in più parti del messaggio, ogni parte può contenere 153 GSM 03,38 caratteri. Di conseguenza, questo messaggio viene inviato come messaggio in due parti.

Esempio 2: un messaggio che contiene caratteri multibyte

Il messaggio seguente contiene diversi caratteri cinesi, tutti non compresi nell'alfabeto GSM 03.38.

```
##################################################·####1994#7#########
```

Il messaggio precedente contiene 71 caratteri. Tuttavia, poiché quasi tutti i caratteri del messaggio non sono compresi nell'alfabeto GSM 03.38, viene inviato come due parti del messaggio. Ciascuna di queste parti può contenere un massimo di 67 caratteri.

Esempio 3: Un messaggio che contiene un solo carattere diverso da un carattere GSM

Il messaggio seguente 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 scrivi i tuoi SMS messaggi

in Microsoft Word e li incolli in AmazonSNS, devi 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 le virgolette singole di chiusura, che non fa parte dell'alfabeto GSM 03.38, viene inviato come due parti del messaggio.

Se sostituisci la virgoletta singola di chiusura di questo messaggio con un apostrofo (che fa parte dell'alfabeto GSM 03.38), il messaggio viene inviato come parte unica del messaggio.

Esempi di codice per l'SNSutilizzo di Amazon AWS SDKs

I seguenti esempi di codice mostrano come usare Amazon SNS con un kit di sviluppo AWS software (SDK).

Le nozioni di base sono esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le operazioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Mentre le azioni mostrano come richiamare le singole funzioni di servizio, è possibile visualizzare le azioni nel loro contesto nei relativi scenari.

Gli scenari sono esempi di codice che mostrano come eseguire attività specifiche richiamando più funzioni all'interno di un servizio o combinandole con altre Servizi AWS.

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Nozioni di base

Ciao Amazon SNS

I seguenti esempi di codice mostrano come iniziare a usare AmazonSNS.

.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 API i dettagli, vedi ListTopics AWS SDK for .NETAPIReference.

C++

SDKper C++

> **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.

Codice per il CMakeLists file.txtCMake.

```
# 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 API i dettagli, vedere ListTopicsin AWS SDK for C++ API Reference.

Go

SDKper 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.

```
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() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 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(ctx)
  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 API i dettagli, vedi ListTopics AWS SDK for GoAPIReference.

Java

SDKper Java 2.x

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

```
package com.example.sns;

import software.amazon.awssdk.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 API i dettagli, vedi ListTopics AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

Inizializza un SNS client 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 API i dettagli, vedi [ListTopics AWS SDK for JavaScript](#)APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento ListTopics AWSSDKa Kotlin API.

Esempi di codice

- Esempi di base per l'SNSutilizzo di Amazon AWS SDKs
  - Ciao Amazon SNS
  - Azioni per l'SNSutilizzo di Amazon AWS SDKs
    - Usalo CheckIfPhoneNumberIsOptedOut con un AWS SDK o CLI
    - Utilizzare ConfirmSubscription con un AWS SDK o CLI
    - Utilizzare CreateTopic con un AWS SDK o CLI
    - Utilizzare DeleteTopic con un AWS SDK o CLI
    - Utilizzare GetSMSAttributes con un AWS SDK o CLI
    - Utilizzare GetTopicAttributes con un AWS SDK o CLI
    - Utilizzare ListPhoneNumbersOptedOut con un AWS SDK o CLI
    - Utilizzare ListSubscriptions con un AWS SDK o CLI
    - Utilizzare ListTopics con un AWS SDK o CLI

Esempi di base per l'SNSutilizzo di Amazon AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di Amazon Simple Notification Service con AWS SDKs.

Esempi

Ciao Amazon SNS

I seguenti esempi di codice mostrano come iniziare a usare AmazonSNS.

.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 API i dettagli, vedi ListTopics AWS SDK for .NETAPIReference.

C++

SDKper C++

> **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.

Codice per il CMakeLists file.txtCMake.

```
# 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 API i dettagli, vedere ListTopicsin AWS SDK for C++ API Reference.

Go

SDKper 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.

```
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() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
```

```
 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(ctx)
  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 API i dettagli, vedi [ListTopics AWS SDK for Go](#)APIReference.

Java

SDKper Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.sns;

import software.amazon.awssdk.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 API i dettagli, vedi ListTopics AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

Inizializza un SNS client 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 API i dettagli, vedi ListTopics AWS SDK for JavaScriptAPIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento ListTopics AWSSDKa Kotlin API.

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consulta. Usare Amazon SNS con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Azioni per l'SNSutilizzo di Amazon AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole SNS azioni Amazon con AWS SDKs. Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Questi estratti si chiamano Amazon SNS API e sono estratti di codice da programmi più grandi che devono essere eseguiti nel contesto. Puoi vedere le azioni nel contesto in. Scenari per l'SNSutilizzo di Amazon AWS SDKs

Gli esempi seguenti includono solo le operazioni più comunemente utilizzate. Per un elenco completo, consulta Amazon Simple Notification Service API Reference.

Esempi

- Usalo CheckIfPhoneNumberIsOptedOut con un AWS SDK o CLI
- Utilizzare ConfirmSubscription con un AWS SDK o CLI
- Utilizzare CreateTopic con un AWS SDK o CLI
- Utilizzare DeleteTopic con un AWS SDK o CLI
- Utilizzare GetSMSAttributes con un AWS SDK o CLI
- Utilizzare GetTopicAttributes con un AWS SDK o CLI
- Utilizzare ListPhoneNumbersOptedOut con un AWS SDK o CLI
- Utilizzare ListSubscriptions con un AWS SDK o CLI
- Utilizzare ListTopics con un AWS SDK o CLI
- Utilizzare Publish con un AWS SDK o CLI
- Utilizzare SetSMSAttributes con un AWS SDK o CLI
- Utilizzare SetSubscriptionAttributes con un AWS SDK o CLI
- Da utilizzare SetSubscriptionAttributesRedrivePolicy con un AWS SDK
- Utilizzare SetTopicAttributes con un AWS SDK o CLI
- Utilizzare Subscribe con un AWS SDK o CLI
- Utilizzare TagResource con un AWS SDK o CLI
- Utilizzare Unsubscribe con un AWS SDK o CLI

Usalo **`CheckIfPhoneNumberIsOptedOut`** con un AWS SDK o 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 API i dettagli, vedi CheckIfPhoneNumberIsOptedOut AWS SDK for .NETAPIReference.

CLI

AWS CLI

Per verificare la disattivazione dei SMS messaggi relativi a 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 SMS messaggi dall'account corrente AWS .

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Per API i dettagli, vedere CheckIfPhoneNumberIsOptedOutin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi CheckIfPhoneNumberIsOptedOut AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere CheckIfPhoneNumberIsOptedOutin AWS SDK for JavaScript APIReference.

PHP

SDK per PHP

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

```
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 API i dettagli, vedi CheckIfPhoneNumberIsOptedOut AWS SDK for PHPAPIReference.

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`ConfirmSubscription`** con un AWS SDK o CLI

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

CLI

AWS CLI

Conferma di una sottoscrizione

Il `confirm-subscription` comando seguente completa il processo di conferma avviato quando l'utente ha sottoscritto un SNS argomento 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 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Output:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Per API i dettagli, vedere ConfirmSubscriptionin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi ConfirmSubscription AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere ConfirmSubscriptionin AWS SDK for JavaScript APIReference.

PHP

SDK per PHP

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

```
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 API i dettagli, vedi ConfirmSubscription AWS SDK for PHPAPIReference.

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`CreateTopic`** con un AWS SDK o 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:

- Crea e pubblica su un FIFO argomento
- 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 specifici FIFO e di deduplicazione.

```
    /// <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 API i dettagli, vedere CreateTopicin AWS SDK for .NET API Reference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi CreateTopic AWS SDK for C++APIReference.

CLI

AWS CLI

Per creare un argomento SNS

L'`create-topic`esempio seguente crea un SNS argomento 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 AWS Command Line Interface User Guide.

- Per API i dettagli, consulta CreateTopic AWS CLICommand Reference.

Go

SDKper 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
}



// 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(ctx context.Context, 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(ctx, &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 API i dettagli, vedi CreateTopic AWS SDK for GoAPIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi CreateTopic AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere CreateTopicin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento CreateTopic AWSSDKa Kotlin API.

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 API i dettagli, vedi CreateTopic AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere CreateTopicPython (Boto3) Reference.AWS SDK API

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi CreateTopic AWS SDK for RubyAPIReference.

Rust

SDKper Rust

> **Note**
>
> c'è altro da fare 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 API i dettagli, CreateTopicconsulta AWS SDKRust API Reference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi CreateTopicSAPABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`DeleteTopic`** con un AWS SDK o 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 per argomentoARN.

```
    /// <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 API i dettagli, DeleteTopicconsulta AWS SDK for .NET APIReference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi DeleteTopic AWS SDK for C++APIReference.

CLI

AWS CLI

Per eliminare un SNS argomento

L'`delete-topic`esempio seguente elimina l'SNSargomento specificato.

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

Questo comando non produce alcun output.

- Per API i dettagli, vedere DeleteTopicin AWS CLI Command Reference.

Go

SDKper 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
}
```

```
// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &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 API i dettagli, vedi DeleteTopic AWS SDK for GoAPIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi DeleteTopic AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere DeleteTopicin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento DeleteTopic AWSSDKa Kotlin API.

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 API i dettagli, vedi DeleteTopic AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere DeleteTopicPython (Boto3) Reference.AWS SDK API

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi DeleteTopicSAPABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`GetSMSAttributes`** con un AWS SDK o CLI

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

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi G etSMSAttributes in AWS SDK for C++ APIReference.

CLI

AWS CLI

Per elencare gli attributi predefiniti dei SMS messaggi

L'`get-sms-attributes`esempio seguente elenca gli attributi predefiniti per l'invio SMS di messaggi.

```
aws sns get-sms-attributes
```

Output:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Per API i dettagli, vedere G etSMSAttributes in AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi G etSMSAttributes in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere G etSMSAttributes in AWS SDK for JavaScript APIReference.

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;


/**
 * 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 API i dettagli, vedi G etSMSAttributes in AWS SDK for PHP APIReference.

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`GetTopicAttributes`** con un AWS SDK o 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 API i dettagli, vedi GetTopicAttributes AWS SDK for .NETAPIReference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi GetTopicAttributes AWS SDK for C++APIReference.

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 API i dettagli, vedere GetTopicAttributesin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi GetTopicAttributes AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere GetTopicAttributesin AWS SDK for JavaScript APIReference.

SDKper JavaScript (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.

Importa i moduli SDK e client e chiama ilAPI.

```
// 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 API i dettagli, vedere GetTopicAttributesin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento GetTopicAttributes AWSSDKa Kotlin API.

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 API i dettagli, vedi GetTopicAttributes AWS SDK for PHPAPIReference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi GetTopicAttributesSAPABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`ListPhoneNumbersOptedOut`** con un AWS SDK o CLI

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

CLI

AWS CLI

Per elencare le opzioni di disattivazione dei SMS messaggi

L'`list-phone-numbers-opted-out`esempio seguente elenca i numeri di telefono a cui è stato negato il consenso alla ricezione dei messaggi. SMS

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Per API i dettagli, vedere ListPhoneNumbersOptedOutin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi ListPhoneNumbersOptedOut AWS SDK for Java 2.xAPIReference.

PHP

SDK per PHP

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

```
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 API i dettagli, vedi ListPhoneNumbersOptedOut AWS SDK for PHPAPIReference.

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`ListSubscriptions`** con un AWS SDK o 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 API i dettagli, vedi ListSubscriptions AWS SDK for .NETAPIReference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi [ListSubscriptions AWS SDK for C++](#)APIReference.

CLI

AWS CLI

Per elencare i tuoi SNS abbonamenti

L'`list-subscriptions`esempio seguente mostra un elenco degli SNS abbonamenti 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 API i dettagli, vedere ListSubscriptionsin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi ListSubscriptions AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere ListSubscriptionsin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento ListSubscriptions AWSSDKa Kotlin API.

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 API i dettagli, vedi ListSubscriptions AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere ListSubscriptionsPython (Boto3) Reference.AWS SDK API

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi ListSubscriptions AWS SDK for RubyAPIReference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi ListSubscriptionsSAPABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`ListTopics`** con un AWS SDK o 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 API i dettagli, vedi ListTopics AWS SDK for .NETAPIReference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, vedi ListTopics AWS SDK for C++APIReference.

CLI

AWS CLI

Per elencare i tuoi SNS argomenti

L'`list-topics`esempio seguente elenca tutti SNS gli argomenti del tuo AWS account.

```
aws sns list-topics
```

Output:

```
{
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Per API i dettagli, vedere ListTopicsin AWS CLI Command Reference.

Go

SDKper 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.

```
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() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 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(ctx)
  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 API i dettagli, vedi ListTopics AWS SDK for GoAPIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi ListTopics AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere ListTopicsin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento ListTopics AWSSDKa Kotlin API.

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 API i dettagli, vedi ListTopics AWS SDK for PHPAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere ListTopicsPython (Boto3) Reference.AWS SDK API

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Per ulteriori informazioni, consulta la Guida per sviluppatori di AWS SDK for Ruby.
- Per API i dettagli, vedi ListTopics AWS SDK for RubyAPIReference.

Rust

SDKper Rust

> **Note**
>
> c'è altro da fare 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 API i dettagli, ListTopicsconsulta AWS SDKRust API Reference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi ListTopicsSAPABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`Publish`** con un AWS SDK o 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:

- Crea e pubblica su un FIFO argomento
- Pubblica un messaggio SMS di testo
- 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 API i dettagli, consulta Publish in AWS SDK for .NET APIReference.

C++

SDKper C++

> **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.

```
//! 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 API i dettagli, consulta Publish in AWS SDK for C++ APIReference.

CLI

AWS CLI

Esempio 1: pubblicazione di un messaggio in un argomento

L'`publish`esempio seguente pubblica il messaggio specificato SNS sull'argomento 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 SMS messaggio 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 API i dettagli, consulta Pubblica in AWS CLI Command Reference.

Go

SDKper 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(ctx context.Context, 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(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Per API i dettagli, consulta Publish in AWS SDK for Go APIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, consulta Publish in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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;
    let deduplicationId;
```

```
    let 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 API i dettagli, consulta Publish in AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Publish in AWS SDKfor Kotlin reference API.

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;


/**
 * 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 API i dettagli, consulta Publish in AWS SDK for PHP APIReference.

PowerShell

Strumenti 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 API i dettagli, vedere Publish in AWS Tools for PowerShell Cmdlet Reference.

Python

SDKper Python (Boto3)

> **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.

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 API i dettagli, consulta Publish in AWS SDKfor Python (Boto3) Reference. API

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Publish in AWS SDK for Ruby APIReference.

Rust

SDKper Rust

> **Note**
>
> c'è altro da fare 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 API i dettagli, consulta Publish in AWS SDKfor Rust API reference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, consulta Pubblica AWS SDKin SAP ABAP API come riferimento.

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`SetSMSAttributes`** con un AWS SDK o CLI

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

C++

SDKper C++

> **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.

Come usare Amazon SNS per impostare l'efaultSMSType attributo D.

```
//! 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 API i dettagli, consulta S etSMSAttributes in AWS SDK for C++ APIReference.

CLI

AWS CLI

Per impostare gli attributi dei SMS messaggi

L'`set-sms-attributes`esempio seguente imposta l'ID mittente predefinito per SMS `MyName` i messaggi su.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Questo comando non produce alcun output.

- Per API i dettagli, vedere S etSMSAttributes in AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedere S etSMSAttributes in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere S etSMSAttributes in AWS SDK for JavaScript APIReference.

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'
]);

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 API i dettagli, vedere S etSMSAttributes in AWS SDK for PHP APIReference.

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`SetSubscriptionAttributes`** con un AWS SDK o CLI

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

CLI

AWS CLI

Impostazione degli attributi della sottoscrizione

L'`set-subscription-attributes`esempio seguente imposta l'`RawMessageDelivery`attributo su un SQS abbonamento.

```
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.

L'`set-subscription-attributes`esempio seguente imposta un `FilterPolicy` attributo a un SQS abbonamento.

```
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.

L'`set-subscription-attributes`esempio seguente rimuove l'`FilterPolicy`attributo da una SQS sottoscrizione.

```
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 API i dettagli, vedere SetSubscriptionAttributesin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi SetSubscriptionAttributes AWS SDK for Java 2.xAPIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, vedere SetSubscriptionAttributesPython (Boto3) Reference.AWS SDK API

Per un elenco completo delle guide per gli AWS SDK sviluppatori e degli esempi di codice, consulta. Usare Amazon SNS con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Da utilizzare **`SetSubscriptionAttributesRedrivePolicy`** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare`SetSubscriptionAttributesRedrivePolicy`.

Java

SDKper 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.

```
// 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`SetTopicAttributes`** con un AWS SDK o 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 API i dettagli, vedere SetTopicAttributesin AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi SetTopicAttributes AWS SDK for Java 2.xAPIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, vedere SetTopicAttributesin AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento SetTopicAttributes AWSSDKa Kotlin API.

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 API i dettagli, vedi SetTopicAttributes AWS SDK for PHPAPIReference.

Ruby

SDKper Ruby

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi SetTopicAttributes AWS SDK for RubyAPIReference.

SAP ABAP

SDKper SAP ABAP

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

```
    TRY.
        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 API i dettagli, vedi SetTopicAttributesSAPABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`Subscribe`** con un AWS SDK o 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:

- Crea e pubblica su un FIFO argomento
- 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 API i dettagli, consulta Subscribe in AWS SDK for .NET APIReference.

C++

SDKper C++

> **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.

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 SQS coda 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 API i dettagli, consulta Subscribe in AWS SDK for C++ APIReference.

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 API i dettagli, consulta Subscribe in AWS CLI Command Reference.

Go

SDKper 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.

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(ctx context.Context, 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(ctx, &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 API i dettagli, consulta Subscribe in AWS SDK for Go APIReference.

Java

SDKper Java 2.x

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

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 HTTP 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 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 API i dettagli, consulta Subscribe in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 SQS coda 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 API i dettagli, consulta Subscribe in AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

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

Sottoscrivi un indirizzo email a un argomento.

```
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 API i dettagli, consulta Subscribe in AWS SDKfor Kotlin reference API.

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.

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 HTTP endpoint 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 API i dettagli, consulta Subscribe in AWS SDK for PHP APIReference.

Python

SDKper Python (Boto3)

> **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.

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 API i dettagli, consulta Subscribe in AWS SDKfor Python (Boto3) Reference. API

Ruby

SDKper Ruby

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

Sottoscrivi un indirizzo email a un argomento.

```
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 API i dettagli, consulta Subscribe in AWS SDK for Ruby APIReference.

Rust

SDKper Rust

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

Sottoscrivi un indirizzo email a un argomento.

```
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 API i dettagli, consulta Subscribe in AWS SDKfor Rust API reference.

SAP ABAP

SDKper SAP ABAP

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

Sottoscrivi un indirizzo email a un argomento.

```
   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 API i dettagli, vedi Iscriviti SAP ABAPAPIcome riferimento.AWS SDK

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`TagResource`** con un AWS SDK o CLI

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

CLI

AWS CLI

Aggiungere un tag a un argomento

L'`tag-resource`esempio seguente aggiunge un tag di metadati all'SNSargomento Amazon 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 API i dettagli, consulta TagResource AWS CLICommand Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, vedi TagResource AWS SDK for Java 2.xAPIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, vedi il riferimento TagResource AWSSDKa Kotlin API.

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consulta. Usare Amazon SNS con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Utilizzare **`Unsubscribe`** con un AWS SDK o 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 con un abbonamentoARN.

```
/// <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 API i dettagli, consulta Annullare l'iscrizione in ReferenceAWS SDK for .NET API.

C++

SDKper C++

> **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.

```
//! 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 API maggiori dettagli, consulta Annullare l'iscrizione in AWS SDK for C++ APIReference.

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 API i dettagli, consulta Annullare l'iscrizione in AWS CLI Command Reference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API maggiori dettagli, consulta Annullare l'iscrizione in AWS SDK for Java 2.x APIReference.

JavaScript

SDKper JavaScript (v3)

> **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.

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({});
```

Importa i moduli SDK e client e chiama ilAPI.

```
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 API i dettagli, consulta Annullare l'iscrizione in AWS SDK for JavaScript APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Annulla l'iscrizione AWS SDKper Kotlin reference API.

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 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 API maggiori dettagli, consulta Annullare l'iscrizione in AWS SDK for PHP APIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, consulta Unsubscribe in AWS SDKfor Python (APIBoto3) Reference.

SAP ABAP

SDKper SAP ABAP

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

```
   TRY.
       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 API maggiori dettagli, vedi Annullare l'iscrizione SAP ABAPAPIcome AWS SDK riferimento.

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Scenari per l'SNSutilizzo di Amazon AWS SDKs

I seguenti esempi di codice mostrano come implementare scenari comuni in Amazon SNS con AWS SDKs. Questi scenari mostrano come eseguire attività specifiche richiamando più funzioni all'interno di Amazon SNS o combinandole con altre Servizi AWS. Ogni scenario include un collegamento al codice sorgente completo, dove puoi trovare istruzioni su come configurare ed eseguire il codice.

Gli scenari si basano su un livello intermedio di esperienza per aiutarti a comprendere le azioni di servizio nel contesto.

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
- Crea un endpoint della piattaforma per le notifiche SNS push di Amazon utilizzando un AWS SDK
- 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
- Crea e pubblica su un SNS argomento FIFO Amazon utilizzando un AWS SDK
- Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un AWS SDK
- Pubblica SMS messaggi su un SNS argomento Amazon utilizzando un AWS SDK
- Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un AWS SDK
- Pubblica un messaggio SNS SMS di testo Amazon utilizzando un AWS SDK
- Pubblica SNS messaggi Amazon nelle SQS code Amazon utilizzando un AWS SDK
- Usa 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

SDKper Java 2.x

Mostra come creare un'applicazione Web dinamica che invia dati utilizzando Amazon API DynamoDB Java e invia un messaggio di testo utilizzando Amazon Simple Notification Service Java. API

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

SDKper JavaScript (v3)

Questo esempio mostra come creare un'app che consenta agli utenti di inviare dati a una tabella Amazon DynamoDB e inviare 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, consulta 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

SDKper Kotlin

Mostra come creare un'applicazione Android nativa che invia dati utilizzando Amazon API DynamoDB Kotlin e invia un messaggio di testo utilizzando Amazon Kotlin. SNS API

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

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

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 usare Amazon Simple Notification Service. NETAPIper creare un'applicazione web con funzionalità di abbonamento e pubblicazione. Inoltre, questa applicazione di esempio traduce anche i messaggi.

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 SNS
- Amazon Translate

Java

SDKper Java 2.x

Mostra come utilizzare Amazon Simple Notification Service Java API 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.

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire l'esempio che utilizza Java AsyncAPI, vedi l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Kotlin

SDKper Kotlin

Mostra come usare Amazon SNS Kotlin API per creare un'applicazione con funzionalità di abbonamento 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Crea un endpoint della piattaforma per le notifiche SNS push di Amazon utilizzando un AWS SDK

I seguenti esempi di codice mostrano come creare un endpoint di piattaforma per le notifiche SNS push di Amazon.

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

SDKper Java 2.x

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
```

```
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application.
 You can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
```

```
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

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

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

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

- APIGateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDKper 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, vedi l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- APIGateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDKper 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, vedi l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- APIGateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDKper 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, vedi l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- APIGateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDKper 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

- APIGateway
- 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

- APIGateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDKper 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, vedi l'esempio completo su GitHub.

Per approfondire l'origine di questo esempio, consulta il post su AWS Community.

Servizi utilizzati in questo esempio

- APIGateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

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

SDKper 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 Amazon Simple Queue Service (Amazon) sottoscritta a un argomento SQS Amazon Simple Notification Service (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

SDKper 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 asynchronous APIs per avviare un processo che pubblica una notifica su un argomento di Amazon Simple Notification Service (AmazonSNS) al termine del processo.

- Esegui il polling di una coda di Amazon Simple Queue Service SQS (Amazon) per un messaggio di completamento del lavoro 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Crea e pubblica su un SNS argomento FIFO Amazon utilizzando un AWS SDK

I seguenti esempi di codice mostrano come creare e pubblicare FIFO su un SNS argomento Amazon.

Java

SDKper Java 2.x

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

In questo esempio

- crea un SNS FIFO argomento Amazon, due SQS FIFO code Amazon 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 API i dettagli, consulta i seguenti argomenti in AWS SDK for Java 2.x API Reference.
  - CreateTopic
  - Pubblicare
  - Subscribe

Python

SDKper Python (Boto3)

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

Crea un SNS FIFO argomento Amazon, sottoscrivi Amazon SQS FIFO e le code standard all'argomento e pubblica un messaggio sull'argomento.

```
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 API i dettagli, consulta i seguenti argomenti in AWS SDKPython (Boto3) Reference. API
  - CreateTopic
  - Pubblicare
  - Subscribe

SAP ABAP

SDKper SAP ABAP

> **Note**
>
> C'è 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 FIFO argomento, iscriviti a una SQS FIFO coda Amazon all'argomento e pubblica un messaggio su un SNS argomento Amazon.

```
    " 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 API i dettagli, consulta i seguenti argomenti SAPABAPAPIcome AWS SDK riferimento.
  - CreateTopic
  - Pubblicare
  - Subscribe

Per un elenco completo delle guide per AWS SDK gli sviluppatori e degli esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come rilevare persone e oggetti in un video con Amazon Rekognition.

Python

SDKper Python (Boto3)

Usa Amazon Rekognition per rilevare volti, oggetti e persone nei video avviando processi di rilevamento asincrono. Questo esempio configura anche Amazon Rekognition per notificare un argomento di Amazon Simple Notification Service SNS (Amazon) quando i lavori vengono completati e sottoscrive una coda Amazon Simple Queue Service SQS (Amazon) 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Pubblica SMS messaggi su un SNS argomento Amazon utilizzando un AWS SDK

L'esempio di codice seguente mostra come:

- Crea un SNS argomento Amazon.
- Sottoscrivere un numero di telefono cellulare all'argomento.
- Pubblica SMS messaggi sull'argomento in modo che tutti i numeri di telefono abbonati ricevano il messaggio contemporaneamente.

Java

SDKper Java 2.x

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

Crea un argomento e restituisciloARN.

```
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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un AWS SDK

Il seguente esempio di codice mostra come pubblicare un messaggio di grandi dimensioni su Amazon SNS utilizzando Amazon S3 per archiviare il payload del messaggio.

Java

SDKper 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, usa Amazon SNS Extended Client Library for 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Pubblica un messaggio SNS SMS di testo Amazon utilizzando un AWS SDK

I seguenti esempi di codice mostrano come pubblicare SMS messaggi utilizzando AmazonSNS.

.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 API i dettagli, consulta Publish in AWS SDK for .NET APIReference.

C++

SDKper C++

> **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.

```
/**
 * 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 API i dettagli, consulta Publish in AWS SDK for C++ APIReference.

Java

SDKper Java 2.x

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.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 API i dettagli, consulta Publish in AWS SDK for Java 2.x APIReference.

Kotlin

SDKper Kotlin

> **Note**
>
> c'è altro da fare. 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 API i dettagli, consulta Publish in AWS SDKfor Kotlin reference API.

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;


/**
 * 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 API i dettagli, consulta Publish in AWS SDK for PHP APIReference.

Python

SDKper Python (Boto3)

> **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.

```
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 API i dettagli, consulta Publish in AWS SDKfor Python (Boto3) Reference. API

Per un elenco completo delle guide per gli AWS SDK sviluppatori e degli esempi di codice, consulta. Usare Amazon SNS con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Pubblica SNS messaggi Amazon nelle SQS code Amazon utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come:

- Crea un argomento (FIFOo menoFIFO).
- 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 racchiuda le operazioni di AmazonSQS.

```
/// <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 racchiuda le operazioni di AmazonSNS.

```
/// <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 API i dettagli, consulta i seguenti argomenti in AWS SDK for .NET API Reference.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Pubblicare
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

C++

SDKper C++

> **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.

```
        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 API i dettagli, consulta i seguenti argomenti in AWS SDK for C++ APIRiferimento.
  - CreateQueue
  - CreateTopic

- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Pubblicare
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Go

SDKper 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.

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(ctx context.Context) (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(ctx, 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(ctx context.Context, 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(ctx, 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(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, 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(ctx, 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(ctx context.Context, 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(ctx, 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(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, 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(ctx, 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(
 ctx context.Context, 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(ctx)
  }
 }()
 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(ctx)
 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(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, 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(ctx)
 }
```

```
 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

Definisci una struttura che racchiuda SNS le azioni di Amazon 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(ctx context.Context, 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(ctx, &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(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &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(ctx context.Context, 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(ctx, &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(ctx context.Context, 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(ctx, &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 racchiuda SQS le azioni di Amazon 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(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &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(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &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(ctx context.Context, 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(ctx, &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(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &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(ctx context.Context, 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(ctx, &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(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &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 API i dettagli, consulta i seguenti argomenti in AWS SDK for Go API Reference.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Pubblicare
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Java

SDKper Java 2.x

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

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
```

```
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Gives the user three options to choose from.
```

```
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <fifoQueueARN>\n\n" +
                "Where:\n" +
                "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
        // }

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)

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

        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_EAST_1)

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

        Scanner in = new Scanner(System.in);
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
        String topicName;
```

```
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this workflow, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
                "You can select from several options for configuring the topic
 and the subscriptions for the queue.\n" +
                "You can then post to the topic and see the results in the
 queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
                "FIFO topics deliver messages in order and support deduplication
 and message filtering.\n" +
                "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                    "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                    +
                    "        If a message is successfully published to an SNS
 FIFO topic, any message published and determined to have the same deduplication
 ID,\n"
                    +
```

```
                    "       within the five-minute deduplication interval, is
accepted but not delivered.\n" +
                    "       For more information about deduplication, see
https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                    "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
```

```
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = "{\n" +
                "     \"Statement\": [\n" +
                "     {\n" +
                "         \"Effect\": \"Allow\",\n" +
                "                 \"Principal\": {\n" +
                "             \"Service\": \"sns.amazonaws.com\"\n" +
                "         },\n" +
                "         \"Action\": \"sqs:SendMessage\",\n" +
                "                 \"Resource\": \"arn:aws:sqs:us-east-1:" +
accountId + ":" + sqsQueueName + "\",\n" +
                "                 \"Condition\": {\n" +
                "             \"ArnEquals\": {\n" +
                "                 \"aws:SourceArn\": \"arn:aws:sns:us-east-1:" +
accountId + ":" + topicName + "\"\n" +
                "             }\n" +
                "         }\n" +
                "     }\n" +
                "     ]\n" +
                " }";

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                    "If you add a filter to this subscription, then only the
filtered messages will be received in the queue.\n"
                            +
                            "For information about message filtering, see
https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                            +
                            "For this example, you can filter messages by a
\"tone\" attribute.");
            System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
                    + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
```

```
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
```

```
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
```

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

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

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

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

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

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                    + "\nSubscription was removed for " +
request.subscriptionArn());
```

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

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                        .id(msg.messageId())
                        .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

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

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .maxNumberOfMessages(5)
                        .build();
```

```
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .messageAttributeNames(msgAttValue) // Include other
message attributes if needed.
                        .maxNumberOfMessages(5)
                        .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

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

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

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

    public static void pubMessageFIFO(SnsClient snsClient,
            String message,
            String topicArn,
```

```
            String msgAttValue,
            String duplication,
            String groupId,
            String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                        .dataType("String")
                        .stringValue("true")
                        .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                            .topicArn(topicArn)
                            .message(message)
```

```
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .messageAttributes(messageAttributes)
                            .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
```

```
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                        .subscriptionArn(result.subscriptionArn())
                        .attributeName(attributeName)
                        .attributeValue(updatedJsonString)
                        .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

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

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);
```

```
            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                    .queueUrl(queueUrl)
                    .attributes(attrMap)
                    .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

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

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
```

```
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                        .queueName(queueName)
                        .attributes(attrs)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                        .queueName(queueName)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

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

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

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

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

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

- Per API i dettagli, consulta i seguenti argomenti in AWS SDK for Java 2.x APIRiferimento.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Pubblicare
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

JavaScript

SDKper JavaScript (v3)

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

Questo 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";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  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.
    const 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;
    let deduplicationId;
    let 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 API i dettagli, consulta i seguenti argomenti in AWS SDK for JavaScript APIRiferimento.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Pubblicare
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

Kotlin

SDKper Kotlin

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

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
```

```
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this workflow, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
```

```
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
```

```
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
```

```
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
 \"tone\" attributes.")
```

```
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)
```

```
    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

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

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
```

```
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
```

```
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: 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.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

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

```
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
```

```
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
```

```
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")
```

```
            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

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
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Per API i dettagli, consulta i seguenti argomenti in riferimento AWS SDKa Kotlin API.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Pubblicare
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

Per un elenco completo delle guide per AWS SDK sviluppatori e degli esempi di codice, consulta. Usare Amazon SNS con un AWS SDK Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Usa 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

SDKper Java 2.x

Mostra come creare una AWS Lambda funzione utilizzando il runtime Lambda Java. API Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. Questo esempio dimostra come creare una funzione Lambda richiamata da API Amazon Gateway che scansiona una tabella Amazon DynamoDB per gli anniversari di lavoro e utilizza Amazon Simple SNS Notification Service (Amazon) per inviare un messaggio di testo ai dipendenti in cui si congratula per la data del primo anniversario.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. GitHub

Servizi utilizzati in questo esempio

- APIGateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDKper JavaScript (v3)

Mostra come creare una AWS Lambda funzione utilizzando il runtime Lambda JavaScript . API Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. Questo esempio dimostra come creare una funzione Lambda richiamata da API Amazon Gateway che scansiona una tabella Amazon DynamoDB per gli anniversari di lavoro e utilizza Amazon Simple SNS Notification Service (Amazon) per inviare un messaggio di testo ai dipendenti in cui si congratula per la data del primo anniversario.

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

- APIGateway
- DynamoDB
- Lambda
- Amazon SNS

Per un elenco completo di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

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

SDKper 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, si crea una funzione Lambda utilizzando il runtime Lambda Java. API 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

SDKper 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, si crea una funzione Lambda utilizzando il runtime Lambda. JavaScript API 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

Esempi serverless per l'utilizzo di Amazon SNS AWS SDKs

I seguenti esempi di codice mostrano come usare Amazon SNS con AWS SDKs.

Esempi

- Richiama una funzione Lambda da un trigger Amazon SNS

Richiama una funzione Lambda da un trigger Amazon SNS

I seguenti esempi di codice mostrano come implementare una funzione Lambda che riceve un evento attivato dalla ricezione di messaggi da un argomento. SNS La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

.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.

Consumo di un SNS evento con Lambda che utilizza. 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

SDKper Go V2

> **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.

Consumare un SNS evento con Lambda utilizzando 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

SDKper Java 2.x

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un SNS evento 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

SDKper JavaScript (v3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un SNS evento con Lambda che utilizza. JavaScript

```
// 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 SNS evento con Lambda che utilizza. TypeScript

```
// 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.

Consumo di un SNS evento con Lambda che utilizza. 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

SDKper Python (Boto3)

> **Note**
>
> C'è di più su. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di Esempi serverless.

Consumo di un SNS evento con Lambda usando 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

SDKper Ruby

> **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.

Consumare un SNS evento con Lambda usando 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

SDKper 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.

Consumare un SNS evento con Lambda utilizzando 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 di guide per AWS SDK sviluppatori ed esempi di codice, consultaUsare Amazon SNS con un AWS SDK. Questo argomento include anche informazioni su come iniziare e dettagli sulle SDK versioni precedenti.

SNSSicurezza Amazon

Questa sezione fornisce informazioni sulla SNS sicurezza, l'autenticazione e il controllo degli accessi di Amazon e su Amazon SNS Access Policy Language.

Argomenti

Protezione SNS dei dati Amazon

Il modello di responsabilità AWS condivisa modello si applica alla protezione dei dati in Amazon Simple Notification Service. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile di mantenere il controllo sui contenuti ospitati su questa infrastruttura. Questo contenuto include le attività di configurazione e gestione della sicurezza per AWS i servizi utilizzati. Per ulteriori informazioni sulla privacy dei dati, consulta la sezione Privacy dei dati FAQ. Per informazioni sulla protezione dei dati in Europa, consulta il Modello di responsabilitàAWS condivisa e GDPR il post sul blog sulla AWS sicurezza.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e di configurare account utente individuali con AWS Identity and Access Management (IAM). In questo modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere il proprio lavoro. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- UsaSSL/TLSper comunicare con AWS le risorse. Consigliamo TLS 1.2 o versioni successive.
- Configurazione API e registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno AWS dei servizi.

- 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 hai bisogno di FIPS 140-2 moduli crittografici convalidati per accedere AWS tramite un'interfaccia a riga di comando o unAPI, usa un endpoint. FIPS Per ulteriori informazioni sugli FIPS endpoint disponibili, vedere Federal Information Processing Standard () 140-2. FIPS
- Protezione dei dati dei messaggi
  - La protezione dei dati dei messaggi è una nuova importante funzionalità di Amazon SNS
  - Utilizzalo MDP per scansionare 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 Nome. Ciò include quando lavori con Amazon SNS o altri Amazon Web Services utilizzando la consoleAPI, AWS CLI, o AWS SDKs. I dati inseriti nei tag o nei campi in formato libero utilizzati per i nomi possono essere utilizzati per i log di fatturazione o di diagnostica. Se fornisci un URL a un server esterno, ti consigliamo vivamente di non includere le informazioni sulle credenziali URL per convalidare la tua richiesta a quel server.

Le seguenti sezioni forniscono ulteriori informazioni sulla protezione dei dati in AmazonSNS.

Argomenti

- Crittografia SNS dei dati Amazon
- Protezione del SNS traffico Amazon con VPC endpoint
- Migliorare la SNS sicurezza di Amazon con Message Data Protection

Crittografia SNS dei dati Amazon

La protezione dei dati si riferisce alla protezione dei dati durante il transito (mentre viaggiano da e verso AmazonSNS) e a riposo (mentre sono archiviati su dischi nei SNS data center Amazon). Puoi

proteggere i dati in transito utilizzando Secure Sockets Layer (SSL) o la crittografia lato client. Per impostazione predefinita, Amazon SNS archivia messaggi e file utilizzando la crittografia del disco. Puoi proteggere i dati inattivi richiedendo SNS ad Amazon di crittografare i tuoi messaggi prima di salvarli nel file system crittografato dei suoi data center. Amazon SNS consiglia di utilizzare SSE una crittografia dei dati ottimizzata.

Argomenti

Protezione dei SNS dati Amazon con la crittografia lato server

La crittografia lato server (SSE) consente di archiviare dati sensibili in argomenti crittografati proteggendo il contenuto dei messaggi negli SNS argomenti di Amazon utilizzando chiavi gestite in AWS Key Management Service ()AWS KMS.

SSEcrittografa i messaggi non appena Amazon li SNS riceve. I messaggi vengono archiviati in forma crittografata e decrittografati solo quando vengono inviati.

- Per informazioni sulla gestione dell'SSEutilizzo di AWS Management Console o AWS SDK for Java (impostando l'`KmsMasterKeyId`attributo utilizzando le `SetTopicAttributes` API azioni `CreateTopic` and), vedere. Configurazione della crittografia per SNS argomenti Amazon con crittografia lato server
- Per informazioni sulla creazione di argomenti crittografati utilizzando AWS CloudFormation (impostando la `KmsMasterKeyId` proprietà utilizzando la `AWS::SNS::Topic` risorsa), consultate la Guida per AWS CloudFormation l'utente.

> **Important**
>
> Tutte le richieste agli argomenti con SSE abilitato devono essere utilizzate HTTPS e Signature Version 4.
> Per informazioni sulla compatibilità di altri servizi con argomenti crittografati, consulta la documentazione del servizio.

> Amazon supporta SNS solo chiavi di crittografia KMS simmetriche. Non puoi utilizzare nessun altro tipo di KMS chiave per crittografare le risorse del servizio. Per informazioni su come determinare se una KMS chiave è una chiave di crittografia simmetrica, vedi Identificazione delle chiavi asimmetriche. KMS

AWS KMS combina hardware e software sicuri e ad alta disponibilità per fornire un sistema di gestione delle chiavi scalabile per il cloud. Quando usi Amazon SNS con AWS KMS, anche le chiavi dati che crittografano i dati dei tuoi messaggi 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 AWS-managed KMS keys per AmazonSNS, che sono uniche per ogni account e regione.
- Gli standard AWS KMS di sicurezza possono aiutarti a soddisfare i requisiti di conformità relativi alla crittografia.

Per ulteriori informazioni, consulta Cos'è? AWS Key Management Service nella Guida per gli AWS Key Management Service sviluppatori.

Argomenti

- Ambito della crittografia
- Termini chiave

Ambito della crittografia

SSEcrittografa il corpo di un messaggio in un SNS argomento di Amazon.

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 arretrati.
> - 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 la funzionalità diSSE. Per descrizioni dettagliate, consulta Amazon Simple Notification Service API Reference.

Chiave di dati

La chiave di crittografia dei dati (DEK) responsabile della crittografia dei contenuti dei SNS messaggi Amazon.

Per ulteriori informazioni, consulta Chiave dati nella AWS Key Management Service Guida per sviluppatori e Pacchetto Crittografia nella AWS Encryption SDK Guida per sviluppatori.

AWS KMS key ID

L'alias, l'aliasARN, l'ID della chiave o la chiave ARN di un account personalizzato AWS KMS key, AWS KMS nel tuo account o in un altro account. Sebbene l'alias di AWS managed AWS KMS for Amazon SNS sia sempre`alias/aws/sns`, l'alias di un account personalizzato AWS KMS può, ad esempio, essere. `alias/MyAlias` Puoi utilizzare queste AWS KMS chiavi per proteggere i messaggi negli SNS argomenti di Amazon.

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

Puoi creare AWS KMS chiavi, definire le politiche che controllano il modo in cui AWS KMS le chiavi possono essere utilizzate e verificarne l' AWS KMS utilizzo utilizzando la AWS KMS keyssezione della AWS KMS console o l'`CreateKey` AWS KMS azione. 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 KeyIdla sezione AWS Key Management Service APIReference. Per informazioni sulla ricerca degli AWS KMS identificatori, consulta Find the Key ID e ARN nella AWS Key Management Service Developer Guide.

> **Important**
>
> Sono previsti costi aggiuntivi per l'utilizzo AWS KMS. Per ulteriori informazioni, consulta AWS KMS Stima dei costi e Prezzi di AWS Key Management Service.

Gestione delle chiavi e dei costi di SNS crittografia Amazon

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 supporta SNS solo chiavi di crittografia KMS simmetriche. Non puoi utilizzare nessun altro tipo di KMS chiave per crittografare le risorse del servizio. Per informazioni su come determinare se una KMS chiave è una chiave di crittografia simmetrica, vedi Identificazione delle chiavi asimmetriche. KMS

Argomenti

- AWS KMS Stima dei costi
- Configurazione delle autorizzazioni AWS KMS
- AWS KMS errori

AWS KMS Stima dei costi

Per prevedere i costi e comprendere meglio la tua AWS bolletta, potresti voler sapere con che frequenza Amazon SNS utilizza la tua AWS KMS key.

> **Note**
>
> Sebbene la formula seguente possa darti un'idea molto precisa dei costi previsti, i costi effettivi potrebbero essere più elevati a causa della natura distribuita di AmazonSNS.

Per calcolare il numero di API richieste (R) per argomento, usa la formula seguente:

```
R = B / D * (2 * P)
```

B è il periodo di fatturazione (in secondi).

Dè il periodo di riutilizzo della chiave dati (in secondi: Amazon SNS riutilizza una chiave dati per un massimo di 5 minuti).

Pè il numero di principali editoriali che inviano all'SNSargomento Amazon.

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 AWS KMS API chiamate 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 AWS KMS API chiamate 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 AWS KMS

Prima di poterle utilizzareSSE, è necessario configurare AWS KMS key le policy per consentire la crittografia degli argomenti e la crittografia e la decrittografia dei messaggi. Per esempi e ulteriori informazioni sulle AWS KMS autorizzazioni, consulta AWS KMS APIPermissions: Actions and Resources Reference nella Developer Guide.AWS Key Management Service Per informazioni dettagliate su come configurare un SNS argomento di Amazon con crittografia lato server, consulta. Imposta un SNS argomento Amazon con crittografia lato server

> **Note**
>
> Puoi anche gestire le autorizzazioni per le chiavi di crittografia simmetriche utilizzando le policy. KMS IAM Per ulteriori informazioni, vedere Utilizzo IAM delle politiche con. AWS KMS Sebbene sia possibile configurare le autorizzazioni globali per l'invio e la ricezione da AmazonSNS, è AWS KMS necessario indicare esplicitamente il nome completo ARN KMSs in aree specifiche nella `Resource` sezione di una politica. IAM

È inoltre necessario assicurarsi che le politiche chiave di AWS KMS key consentano le autorizzazioni necessarie. A tale scopo, indica i responsabili che producono e utilizzano messaggi crittografati in Amazon SNS come utenti nella policy KMS chiave.

In alternativa, puoi specificare le AWS KMS azioni richieste e KMS ARN in una IAM politica assegnata ai mandanti che pubblicano e sottoscrivono per ricevere messaggi crittografati in AmazonSNS. Per ulteriori informazioni, consulta Gestione dell'accesso a AWS KMS nella Guida per sviluppatori di AWS Key Management Service .

Se selezioni una chiave gestita dal cliente per il tuo SNS argomento Amazon e utilizzi degli alias per controllare l'accesso alle KMS chiavi utilizzando IAM policy o policy KMS chiave con la condition key`kms:ResourceAliases`, assicurati che alla chiave gestita dal cliente selezionata sia associato anche un alias. Per ulteriori informazioni sull'utilizzo degli alias per controllare l'accesso alle KMS chiavi, consulta Usare gli alias per controllare l'accesso alle chiavi nella Developer Guide. KMS AWS Key Management Service

Consenti a un utente di inviare messaggi a un argomento con SSE

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"
  }]
}
```

Abilita la compatibilità tra le fonti di eventi provenienti dai AWS servizi e gli argomenti crittografati

Diversi AWS servizi pubblicano eventi su SNS argomenti di Amazon. 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 AWS servizio di disporre delle `kms:Decrypt` autorizzazioni `kms:GenerateDataKey*` and, aggiungi la seguente dichiarazione alla KMS policy.

   ```
   {
     "Statement": [{
       "Effect": "Allow",
       "Principal": {
         "Service": "service.amazonaws.com"
       },
       "Action": [
         "kms:GenerateDataKey*",
         "kms:Decrypt"
   ```

```
    ],
    "Resource": "*"
  }]
}
```

Origine eventi	Principale del servizio
Amazon CloudWatch	`cloudwatch.amazonaws.com`
CloudWatch Eventi Amazon	`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`
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 fonti di SNS eventi Amazon richiedono che tu fornisca un IAM ruolo (anziché il responsabile del servizio) nella AWS KMS key politica:
>
> - Amazon EC2 Auto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. Aggiungi i tasti `aws:SourceAccount` e `aws:SourceArn` condition alla politica delle KMS risorse per proteggere ulteriormente la KMS chiave da confusi attacchi secondari. Fare riferimento all'elenco della documentazione specifica del servizio (sopra) per i dettagli esatti in ciascun caso.

> **Important**
>
> L'aggiunta di `aws:SourceAccount` e `aws:SourceArn` a una AWS KMS politica non è supportata per EventBridge-to-encrypted gli argomenti.

```
{
  "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. Abilita SSE il tuo argomento usando il tuoKMS.
5. Fornisci ARN l'argomento crittografato alla fonte dell'evento.

AWS KMS errori

Quando lavori con Amazon SNS e AWS KMS, potresti riscontrare 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.

HTTPCodice di stato: 400

KMSDisabledException

La richiesta è stata rifiutata perché il valore specificato KMS non è abilitato.

HTTPCodice di stato: 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 .

HTTPCodice di stato: 400

KMSNotFoundException

La richiesta è stata rifiutata perché non è possibile trovare l'entità o la risorsa specificata.

HTTPCodice di stato: 400

KMSOptInRequired

L'ID della chiave di AWS accesso richiede un abbonamento al servizio.

HTTPCodice di stato: 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 .

HTTPCodice di stato: 400

Configurazione della crittografia per SNS argomenti Amazon con crittografia lato server

Con la crittografia lato server (SSE), puoi archiviare dati sensibili in argomenti crittografati. SSEprotegge il contenuto dei messaggi negli SNS argomenti di Amazon utilizzando chiavi gestite in AWS Key Management Service (AWS KMS). Per ulteriori informazioni sulla crittografia lato server con AmazonSNS, consulta. Protezione dei SNS dati Amazon con la crittografia lato server Per ulteriori informazioni sulla creazione di AWS KMS chiavi, consulta Creating keys nella AWS Key Management Service Developer Guide.

> **Important**
>
> Tutte le richieste relative agli argomenti con «must use» SSE abilitato HTTPS e Signature Version 4.

Abilita la crittografia lato server (SSE) per un SNS argomento Amazon utilizzando il AWS Management Console

1. Accedi alla SNSconsole Amazon.
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 AWS KMS chiave. Per ulteriori informazioni, consulta Termini chiave.

      Per ogni KMS tipo, KMSARNvengono visualizzati la Descrizione, l'Account e.

> **Important**
>
> Se non sei il proprietario diKMS, o se accedi con un account che non dispone delle `kms:DescribeKey` autorizzazioni `kms:ListAliases` e, non potrai visualizzare le informazioni in merito KMS sulla SNS console Amazon.
> Chiedi al proprietario di KMS concederti queste autorizzazioni. Per ulteriori informazioni, consulta la sezione AWS KMS APIAutorizzazioni: azioni e risorse di riferimento nella Guida per gli AWS Key Management Service sviluppatori.

- L'opzione AWS managed KMS for Amazon SNS (impostazione predefinita) alias/aws/sns è selezionata per impostazione predefinita.

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

- Per utilizzare una personalizzazione KMS del tuo AWS account, scegli il campo KMSchiave, quindi scegli la personalizzazione KMS dall'elenco.

  > **Note**
  >
  > Per istruzioni sulla creazione di chiavi personalizzateKMSs, consulta Creating Keys nella AWS Key Management Service Developer Guide

- Per utilizzare un codice personalizzato KMS ARN del tuo AWS account o di un altro AWS account, inseriscilo nel campo KMSchiave.

5. Scegli Save changes (Salva modifiche).

   SSEè abilitato per il tuo argomento e per il ***`MyTopic`***viene visualizzata la pagina.

Lo stato di crittografia, l' AWS account, la chiave principale del cliente (CMK) e la CMKARNdescrizione dell'argomento vengono visualizzati nella scheda Crittografia.

Imposta un SNS argomento Amazon con crittografia lato server

Quando crei la tua KMS chiave, utilizza la seguente politica KMS chiave:

```
{
    "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"
        }
    }
}
```

Impatto sui consumatori

Quando SSE è abilitato per un SNS argomento Amazon, il processo di consumo dei messaggi rimane invariato per gli abbonati. AWS gestisce il processo di crittografia e decrittografia utilizzando. KMS Pertanto, gli abbonati non devono apportare alcuna modifica alla configurazione esistente per gestire i messaggi crittografati. AWS assicura che i messaggi siano crittografati quando sono inattivi e decrittografati automaticamente prima della consegna agli abbonati. Ciò significa che gli abbonati continueranno a ricevere ed elaborare i messaggi come facevano prima dell'attivazione della crittografia, senza richiedere alcuna configurazione o logica di decrittografia aggiuntiva. Inoltre, ne AWS consiglia l'utilizzo HTTPS per garantire la trasmissione sicura dei messaggi.

Configurazione della crittografia degli SNS argomenti Amazon con abbonamento Amazon SQS Queue crittografato

Puoi abilitare la crittografia lato server (SSE) per un argomento per proteggerne i dati. Per consentire SNS ad Amazon di inviare messaggi a SQS code Amazon crittografate, la chiave gestita dal cliente associata alla SQS coda Amazon deve avere una dichiarazione politica che garantisca l'accesso da parte del responsabile del servizio SNS Amazon alle azioni e. AWS KMS API `GenerateDataKey` `Decrypt` Per ulteriori informazioni sull'utilizzo, consulta. SSE Protezione dei SNS dati Amazon con la crittografia lato server

Questa pagina mostra come abilitare SSE un SNS argomento Amazon a cui è sottoscritta una SQS coda Amazon crittografata, utilizzando il. AWS Management Console

Passaggio 1: crea una chiave personalizzata KMS

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 KMS chiave di crittografia simmetrica, per Tipo di chiave scegli Simmetrico.

   Per informazioni su come creare una chiave asimmetrica nella console, vedi Creazione di KMS chiavi asimmetriche (AWS KMS console). KMS
4. In Utilizzo della chiave, l'opzione Crittografa e decrittografa è selezionata per impostazione predefinita.

   Per informazioni su come creare KMS chiavi che generano e verificano MAC codici, vedi Creazione di chiavi. HMAC KMS

   Per informazioni sulle Opzioni avanzate, consultare Special-purpose keys (Chiavi per uso speciale).
5. Scegli Next (Successivo).
6. Digitate un alias per la KMS chiave. Un nome di alias non può iniziare con **`aws/`**. Il **`aws/`** prefisso è riservato da Amazon Web Services per essere rappresentato 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, consulta ABAC AWS KMS Fore and Using alias to control access to keys. KMS

Un alias è un nome visualizzato che è possibile utilizzare per identificare la KMS chiave. Ti consigliamo di scegliere un alias che indichi il tipo di dati che intendi proteggere o l'applicazione che intendi utilizzare con la KMS chiave.

Gli alias sono necessari quando si crea una KMS chiave in. AWS Management Console Sono facoltativi quando si utilizza l'CreateKeyoperazione.

7. (Facoltativo) Digitate una descrizione per la KMS chiave.

   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 in AWS Management Console o utilizza l'UpdateKeyDescriptionoperazione.

8. (Facoltativo) Digita tag per una chiave di un valore di tag facoltativo. Per aggiungere più di un tag alla KMS chiave, scegli Aggiungi tag.

   > **Note**
   >
   > L'aggiunta di tag o detag a una KMS chiave può consentire o negare l'autorizzazione alla chiave. KMS Per i dettagli, consulta ABACPer AWS KMS e come utilizzare i tag per controllare l'accesso alle chiavi. KMS

   Quando aggiungi tag alle tue AWS risorse, AWS genera un rapporto sull'allocazione dei costi con utilizzo e costi aggregati per tag. I tag possono essere utilizzati anche per controllare l'accesso a una KMS chiave. Per informazioni sull'etichettatura delle KMS chiavi, vedi Tagging keys and ABAC for. AWS KMS

9. Scegli Next (Successivo).
10. Seleziona gli IAM utenti e i ruoli che possono amministrare la chiave. KMS

> **Note**
>
> Questa politica chiave offre il Account AWS pieno controllo di questa KMS chiave. Consente agli amministratori degli account IAM di utilizzare le politiche per concedere ad altri responsabili il permesso di gestire la KMS chiave. Per informazioni dettagliate, consultare Default key policy (Policy della chiave predefinita).
>
> IAMle migliori pratiche scoraggiano l'uso di IAM utenti con credenziali a lungo termine. Quando possibile, utilizzate IAM ruoli che forniscono credenziali temporanee. Per i dettagli, consulta le migliori pratiche di sicurezza IAM nella Guida per l'IAMutente.

11. (Facoltativo) Per impedire agli IAM utenti e ai ruoli selezionati di eliminare questa KMS chiave, nella sezione Eliminazione della chiave nella parte inferiore della pagina, deseleziona la casella di controllo Consenti agli amministratori chiave di eliminare questa chiave.
12. Scegli Next (Successivo).
13. Seleziona gli IAM utenti e i ruoli che possono utilizzare la chiave nelle operazioni crittografiche. Scegli Next (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.

Passaggio 2: creare un SNS argomento Amazon crittografato

1. Accedi alla SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Scegli 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, ARNvengono visualizzate la Descrizione, l'Account e la chiave gestita dal cliente.

      > **Important**
      >
      > Se non sei il proprietario della chiave gestita dal cliente o se accedi con un account che non dispone delle `kms:DescribeKey` autorizzazioni `kms:ListAliases` and, non potrai visualizzare le informazioni sulla chiave gestita dal cliente sulla SNS console Amazon.
      > Richiedere al proprietario della chiave gestita dal cliente di concedere queste autorizzazioni. Per ulteriori informazioni, consulta il riferimento AWS KMS API Autorizzazioni: azioni e risorse nella Guida per gli AWS Key Management Service sviluppatori.

   c. Per la chiave gestita dal cliente, scegli MyCustomKeyquella che hai creato in precedenza, quindi scegli Abilita la crittografia lato server.
6. Scegli Save changes (Salva modifiche).

   SSEè abilitato per l'argomento e viene MyTopicvisualizzata la pagina.

   Gli argomenti Stato di crittografia, AWS Account, chiave gestita dal cliente ARN, chiave gestita dal cliente e Descrizione vengono visualizzati nella scheda Crittografia.

Il nuovo argomento crittografato viene visualizzato nell'elenco degli argomenti.

Fase 3: Creare e sottoscrivere SQS code Amazon criptate

1. Accedi alla SQSconsole Amazon.
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 Usa SSE.
   d. Per AWS KMS key, scegli quello MyCustomKeyche hai creato in precedenza, quindi scegli 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. Sulla SQS console Amazon, scegli `MyEncryptedQueue1` `MyEncryptedQueue2` e quindi scegli Queue Actions, Subscribe Queues to SNS Topic.
6. Nella finestra di dialogo Iscriviti a un argomento, per Scegli un argomento seleziona MyEncryptedTopic, quindi scegli 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 SNSconsole Amazon.
2. Nel pannello di navigazione, scegliere Argomenti.
3. Dall'elenco degli argomenti, scegli MyEncryptedTopice quindi scegli 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. Accedi alla SQSconsole Amazon.
2. Dall'elenco delle code, scegli MyEncryptedQueue1, quindi scegli Invia e ricevi messaggi.
3. Nella pagina Invia e ricevi messaggi in MyEncryptedQueue 1, scegli Sondaggio 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.

Protezione del SNS traffico Amazon con VPC endpoint

Un endpoint Amazon Virtual Private Cloud (AmazonVPC) per Amazon SNS è un'entità logica all'interno di un VPC che consente la connettività solo ad AmazonSNS. VPCIndirizza le richieste ad Amazon SNS e reindirizza le risposte aVPC. Le seguenti sezioni forniscono informazioni sull'utilizzo degli VPC endpoint e sulla creazione di policy per gli VPC endpoint.

Se utilizzi Amazon Virtual Private Cloud (AmazonVPC) per ospitare AWS le tue risorse, puoi stabilire una connessione privata tra te VPC e AmazonSNS. Con questa connessione, puoi pubblicare messaggi sui tuoi SNS argomenti Amazon senza inviarli tramite Internet pubblico.

Amazon VPC è un AWS servizio che puoi utilizzare per avviare AWS risorse in una rete virtuale definita dall'utente. Con aVPC, hai il controllo sulle impostazioni di rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. Per connetterti VPC ad AmazonSNS, definisci un VPCendpoint di interfaccia. Questo tipo di endpoint ti consente di connetterti ai serviziVPC. AWS L'endpoint fornisce una connettività affidabile e scalabile ad Amazon SNS senza richiedere un gateway Internet, un'istanza di traduzione degli indirizzi di rete (NAT) o VPN una connessione. Per ulteriori informazioni, consulta Interface VPC Endpoints nella Amazon VPC User Guide.

Le informazioni contenute in questa sezione sono destinate agli utenti di AmazonVPC. Per ulteriori informazioni e per iniziare a creare un fileVPC, consulta Getting Started With Amazon VPC nella Amazon VPC User Guide.

> **Note**
>
> VPCgli endpoint non consentono di sottoscrivere un SNS argomento Amazon a un indirizzo IP privato.

Argomenti

- Creazione di un VPC endpoint Amazon per Amazon SNS
- Creazione di una policy VPC sugli endpoint di Amazon per Amazon SNS
- Pubblicazione di un SNS messaggio Amazon da Amazon VPC

Creazione di un VPC endpoint Amazon per Amazon SNS

Per pubblicare messaggi sui tuoi SNS argomenti Amazon da un AmazonVPC, crea un VPC endpoint di interfaccia. Quindi, puoi pubblicare messaggi sui tuoi argomenti mantenendo il traffico all'interno della rete che gestisci con. VPC

Utilizza le seguenti informazioni per creare l'endpoint e testare la connessione tra il tuo VPC e AmazonSNS. In alternativa, per una procedura guidata che consente di iniziare da zero, consulta Pubblicazione di un SNS messaggio Amazon da Amazon VPC.

Creazione dell'endpoint

Puoi creare un SNS endpoint Amazon VPC utilizzando AWS Management Console, the AWS CLI, an AWS SDK SNSAPI, Amazon o AWS CloudFormation.

Per informazioni sulla creazione e configurazione di un endpoint utilizzando la VPC console Amazon o la AWS CLI, consulta Creating an Interface Endpoint nella Amazon VPC User Guide.

> **Important**
>
> Puoi utilizzare Amazon Virtual Private Cloud solo con gli SNS endpoint HTTPS Amazon. Quando crei un endpoint, specifica Amazon SNS come servizio a cui desideri VPC connetterti. Nella VPC console Amazon, 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 AmazonVPC, devi abilitare gli endpoint privati DNS e specificare gli endpoint nel formato`sns.us-east-2.amazonaws.com`.
Private DNS non supporta endpoint legacy 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 *AWS::EC2::VPCEndpoint*risorsa nella Guida per l'utente.AWS CloudFormation

Verifica della connessione tra il tuo VPC e Amazon SNS

Dopo aver creato un endpoint per AmazonSNS, puoi pubblicare messaggi dai tuoi VPC SNS argomenti su Amazon. Per testare questa connessione, procedi come indicato di seguito:

1. Connettiti a un'EC2istanza Amazon che risiede nel tuoVPC. Per informazioni sulla connessione, consulta Connect to Your Linux Instance o Connecting to Your Windows Instance nella EC2 documentazione di Amazon.

   Ad esempio, per connetterti a un'istanza Linux utilizzando un SSH client, esegui il seguente comando 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 fornita da Amazon EC2 al momento della creazione dell'istanza.
   - instance-hostname è il nome host pubblico dell'istanza. Per ottenere il nome host nella EC2console Amazon: scegli Istanze, scegli la tua istanza e trova il valore per Public DNS () IPv4.
2. Dalla tua istanza, usa il SNS `publish`comando Amazon con 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 scaricarlo ARN dalla SNSconsole Amazon: scegli Argomenti, trova l'argomento e trova il valore nella ARNcolonna.

Se il messaggio viene ricevuto con successo da AmazonSNS, il terminale stampa un ID del messaggio, come il seguente:

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Creazione di una policy VPC sugli endpoint di Amazon per Amazon SNS

Puoi creare una policy per Amazon VPC endpoints for Amazon SNS in cui specifichi quanto segue:

- Il principale che può eseguire azioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire azioni.

Per ulteriori informazioni, consulta Controlling Access to Services with VPC Endpoints nella Amazon VPC User Guide.

Il seguente esempio di politica VPC sugli endpoint specifica che l'IAMutente `MyUser` è autorizzato a pubblicare sull'argomento AmazonSNS. `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 SNS API azioni di Amazon, ad esempio `sns:Subscribe` `esns:Unsubscribe`.
- Altri IAM utenti e regole che tentano di utilizzare questo VPC endpoint.
- `MyUser`pubblicazione su un SNS argomento Amazon diverso.

> **Note**
>
> L'IAMutente può comunque utilizzare altre SNS API azioni Amazon dall'esterno diVPC.

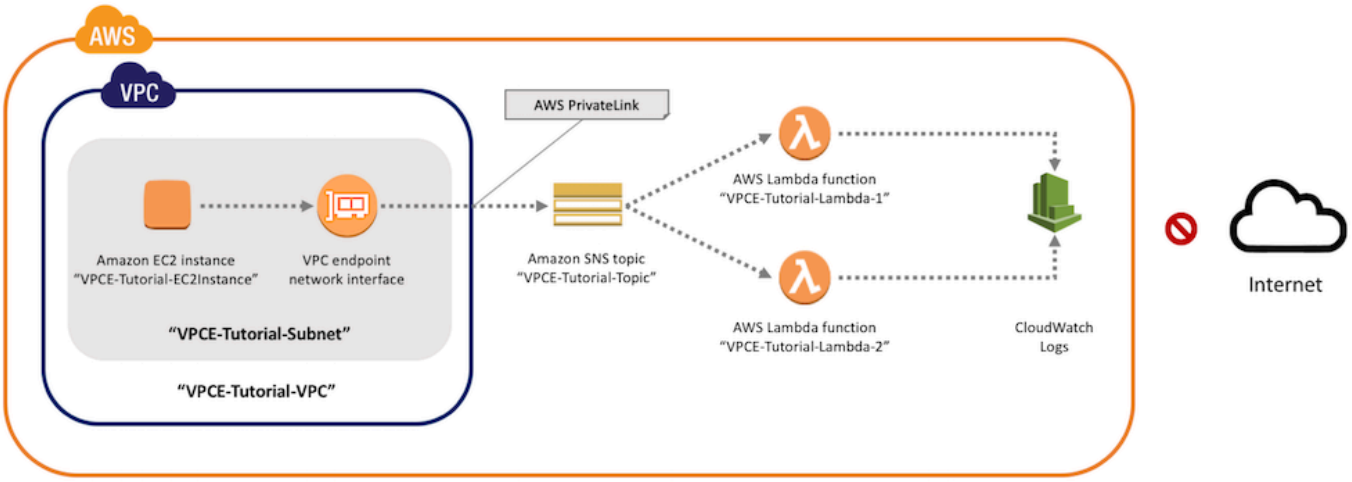
Pubblicazione di un SNS messaggio Amazon da Amazon VPC

Questa sezione descrive come pubblicare su un SNS argomento Amazon mantenendo i messaggi al sicuro in una rete privata. Pubblichi un messaggio da un'EC2istanza Amazon ospitata in Amazon Virtual Private Cloud (AmazonVPC). Il messaggio rimane all'interno della AWS rete senza viaggiare sulla rete Internet pubblica. Pubblicando messaggi in privato da unVPC, puoi migliorare la sicurezza del traffico tra le tue applicazioni e AmazonSNS. Questa sicurezza è importante quando pubblichi informazioni di identificazione personale (PII) sui tuoi clienti o quando la tua applicazione è soggetta alle normative di mercato. Ad esempio, la pubblicazione privata è utile se si dispone di un sistema sanitario che deve essere conforme all'Health Insurance Portability and Accountability Act (HIPAA) o 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:

- Utilizza un AWS CloudFormation modello per creare automaticamente una rete privata temporanea nel tuo. Account AWS
- Crea un VPC endpoint che li connetta VPC con AmazonSNS.
- Accedi a un'EC2istanza Amazon e pubblica un messaggio privato su un SNS argomento Amazon.
- Verificare che il messaggio sia stato consegnato correttamente.
- Elimina le risorse che hai creato durante questo processo in modo che non rimangano nelle tue Account AWS.

Il diagramma seguente illustra la rete privata che crei nel tuo AWS account durante il completamento di questi passaggi:



Questa rete è composta da una VPC che contiene un'EC2istanza Amazon. L'istanza si connette ad Amazon SNS tramite un VPCendpoint di interfaccia. Questo tipo di endpoint si connette a servizi alimentati da. AWS PrivateLink Una volta stabilita questa connessione, puoi accedere all'EC2istanza Amazon e pubblicare messaggi SNS sull'argomento Amazon, anche se la rete è disconnessa dalla rete Internet pubblica. L'argomento suddivide i messaggi che riceve in due funzioni di sottoscrizione AWS Lambda . Queste funzioni registrano i messaggi che ricevono 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 ti registri, il tuo account viene automaticamente registrato per tutti i servizi in AWS, inclusi Amazon SNS e

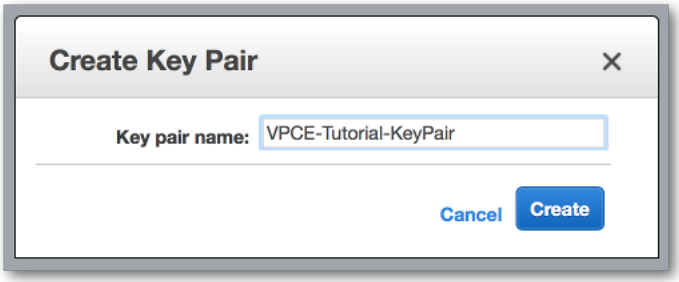
AmazonVPC. Se non hai già creato un account, vai a https://aws.amazon.com/, quindi scegli Crea un account gratuito.

Fase 1: creare una coppia di EC2 chiavi Amazon

Una coppia di key pair viene utilizzata per accedere a un'EC2istanza Amazon. 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, utilizzi la key pair per accedere a un'EC2istanza Amazon. 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. Accedi a AWS Management Console e apri la EC2 console Amazon 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 fornisce al file un'estensione di `.pem`.
6. (Facoltativo) Se utilizzi un SSH client su un computer Mac o Linux per connetterti alla tua istanza, usa il `chmod` comando per impostare le autorizzazioni del tuo file di chiave privata in modo che solo tu possa leggerlo:

   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: Creare le risorse AWS

Per configurare l'infrastruttura, si utilizza un AWS CloudFormation modello. Un modello è un file che funge da modello per la creazione di AWS risorse, come EC2 istanze Amazon e argomenti AmazonSNS. Il modello per questo processo è disponibile GitHub e può essere scaricato.

Fornisci il modello e fornisci AWS CloudFormation le risorse di cui hai bisogno come pila del tuo Account AWS. AWS CloudFormation Uno stack è una raccolta di risorse che puoi gestire come un'unità singola. Una volta completati questi passaggi, puoi AWS CloudFormation eliminare tutte le risorse dello stack contemporaneamente. Queste risorse non rimangono nelle tue mani Account AWS, a meno che tu non voglia che lo facciano.

Lo stack per questa procedura include le seguenti risorse:

- A VPC e le risorse di rete associate, tra cui una sottorete, un gruppo di sicurezza, un gateway Internet e una tabella di routing.
- Un'EC2istanza Amazon lanciata nella sottorete in. VPC
- Un SNS argomento di Amazon.
- Due AWS Lambda funzioni. Queste funzioni ricevono messaggi pubblicati SNS sull'argomento Amazon e registrano gli eventi in CloudWatch Logs.
- CloudWatch Metriche e log di Amazon.
- Un IAM ruolo che consente all'EC2istanza Amazon di utilizzare Amazon SNS e un IAM ruolo che consente alle funzioni Lambda di scrivere CloudWatch nei log.

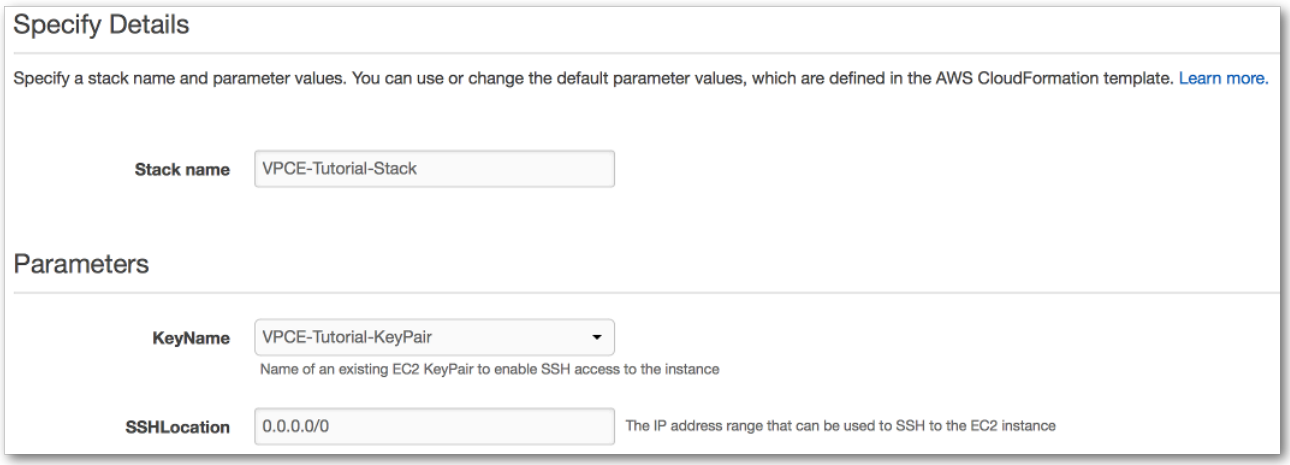
Per creare le risorse AWS

1. Scaricate il file modello dal GitHub sito Web.
2. Accedere alla console AWS CloudFormation.
3. Scegli Create Stack (Crea stack).
4. 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).
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, scegli VPCE-Tutorial-. KeyPair

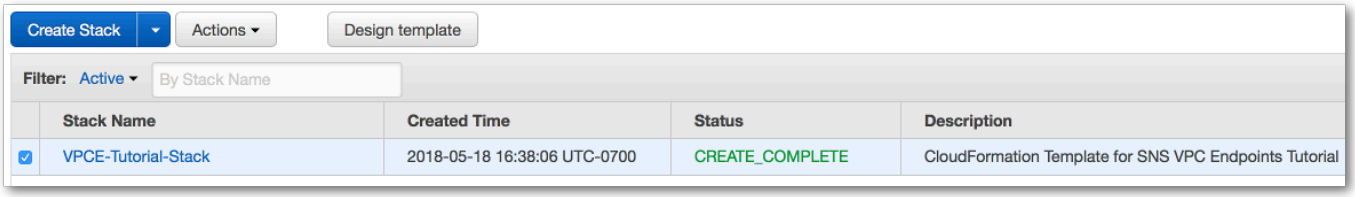
c. Per SSHLocation, mantieni il valore predefinito di. **`0.0.0.0/0`**



d. Scegli 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 Capacità, riconosci che AWS CloudFormation potrebbe creare IAM risorse con nomi personalizzati.
9. Scegli Create (Crea) .

   La AWS CloudFormation console apre la pagina Stacks. Lo VPCE-Tutorial-Stack stato è CREATEPROGRESS_IN_. Dopo pochi minuti, una volta completato il processo di creazione, lo stato diventa _. CREATE COMPLETE



> **Tip**
>
> Scegli il pulsante Refresh (Aggiorna) per visualizzare l'ultimo stato dello stack.

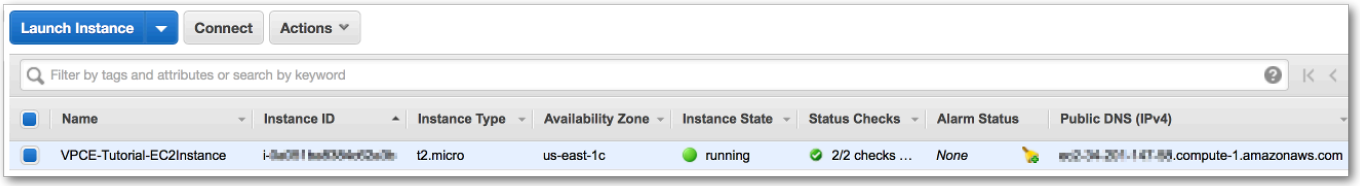
Passaggio 3: verifica che la tua EC2 istanza Amazon non abbia accesso a Internet

L'EC2istanza Amazon che è stata lanciata VPC nel passaggio precedente non ha accesso a Internet. Non consente il traffico in uscita e non è in grado di pubblicare messaggi su Amazon. SNS Verifica accedendo all'istanza. Quindi, prova a connetterti a un endpoint pubblico e prova a inviare un messaggio ad AmazonSNS.

A questo punto della procedura il tentativo di pubblicazione ha esito negativo. In una fase successiva, dopo aver creato un VPC endpoint per AmazonSNS, il tentativo di pubblicazione ha esito positivo.

Per connetterti alla tua EC2 istanza Amazon

1. Apri la EC2 console Amazon 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. Apri un terminale. Dalla directory che contiene la key pair, connettiti all'istanza utilizzando il seguente comando, dove *instance-hostname* è il nome host che hai copiato dalla console AmazonEC2:

```
$ 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 disponga di connettività ad Amazon SNS

1. Accedi alla SNSconsole Amazon.
2. Nel menu di navigazione a sinistra, scegli Topics (Argomenti).
3. Nella pagina Argomenti, copia 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: creare un VPC endpoint Amazon per Amazon SNS

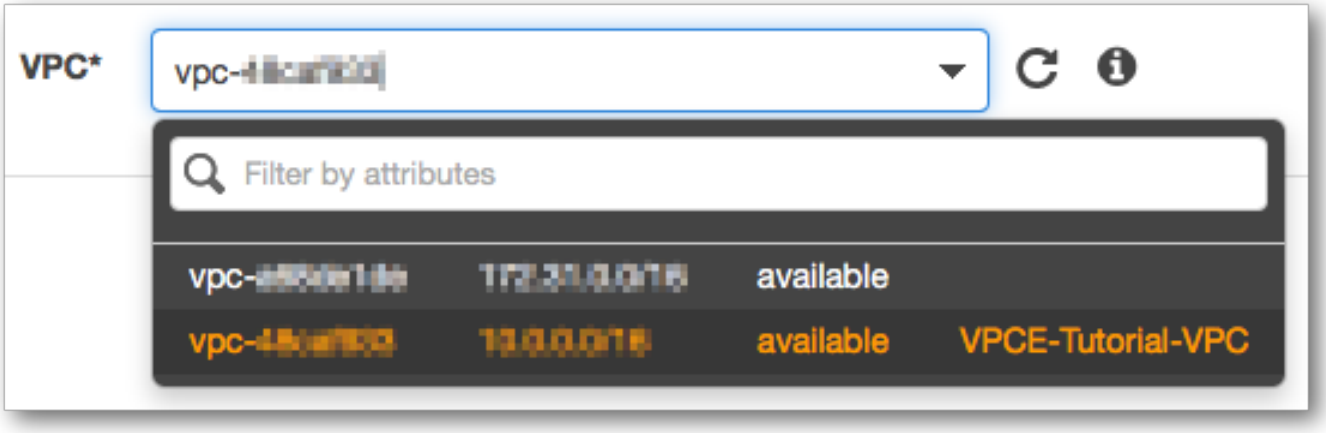
Per connetterlo VPC ad AmazonSNS, definisci un VPC endpoint di interfaccia. Dopo aver aggiunto l'endpoint, puoi accedere all'EC2istanza Amazon del tuo VPC dispositivo e da lì puoi utilizzare Amazon SNSAPI. È possibile pubblicare messaggi all'argomento che vengono pubblicati privatamente. Rimangono all'interno della AWS rete e non viaggiano sulla rete Internet pubblica.

> **Note**
>
> L'istanza non ha ancora accesso ad altri AWS servizi ed endpoint su Internet.

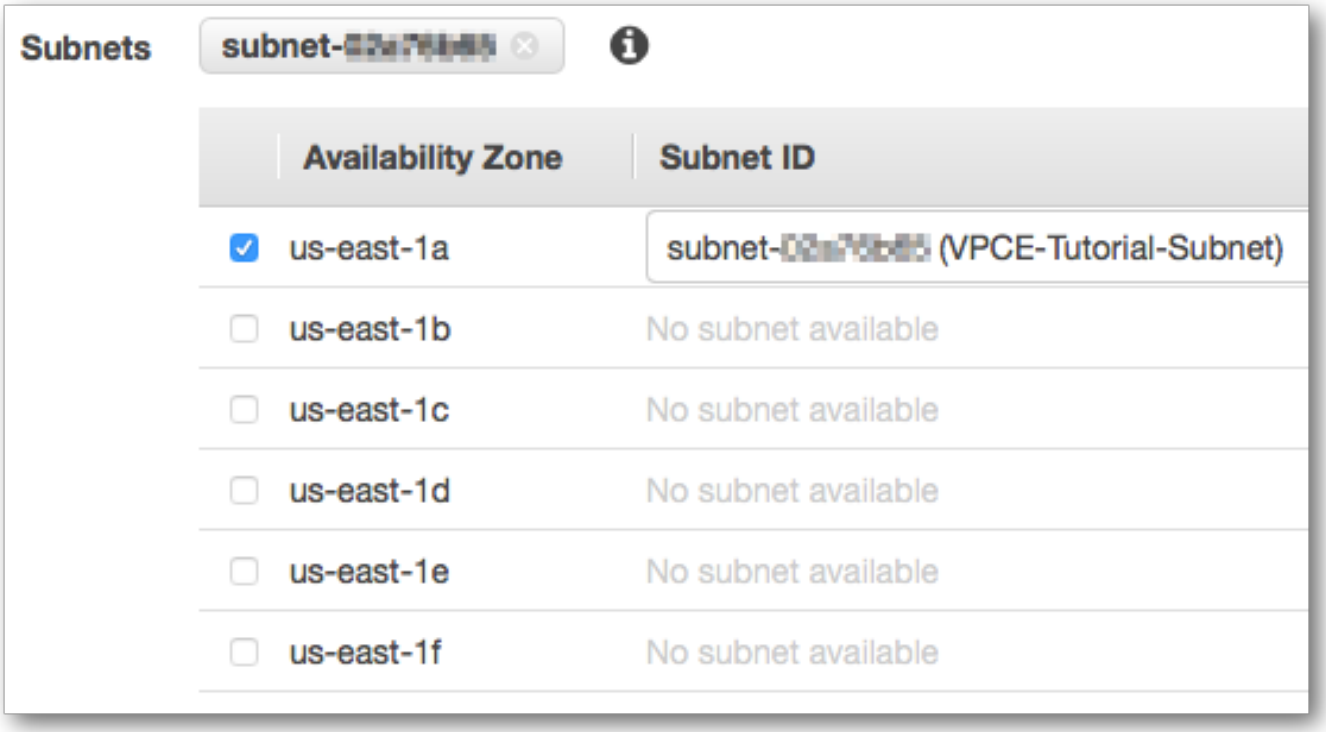
Per creare l'endpoint

1. Apri la VPC console Amazon 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 Service Name, scegli il nome del servizio per AmazonSNS.

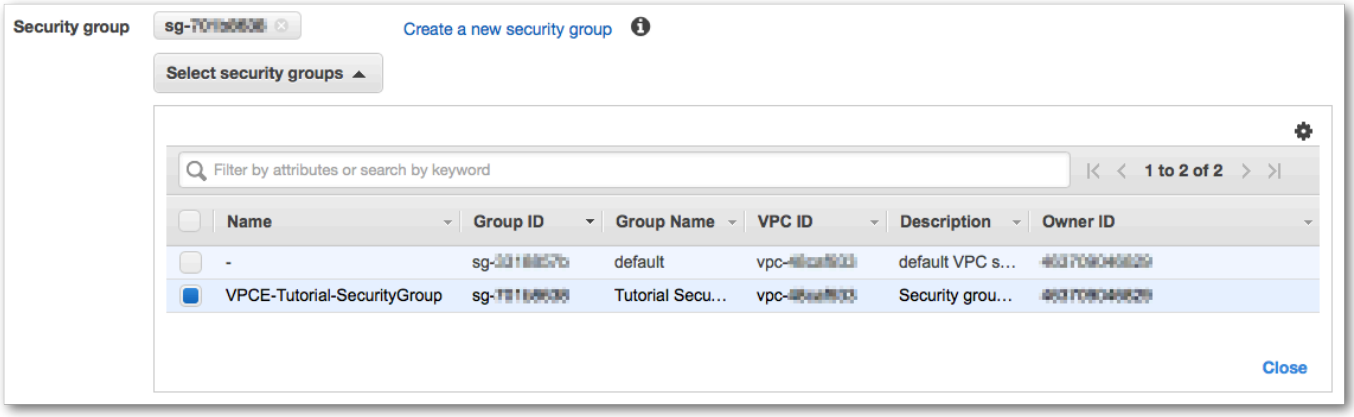
   I nomi dei servizi variano in base alla regione scelta. Ad esempio, se hai scelto Stati Uniti orientali (Virginia settentrionale), il nome del servizio è com.amazonaws.*us-east-1*.sns.
6. Per VPC, scegli VPC quello che ha il nome VPCE-Tutorial -. VPC



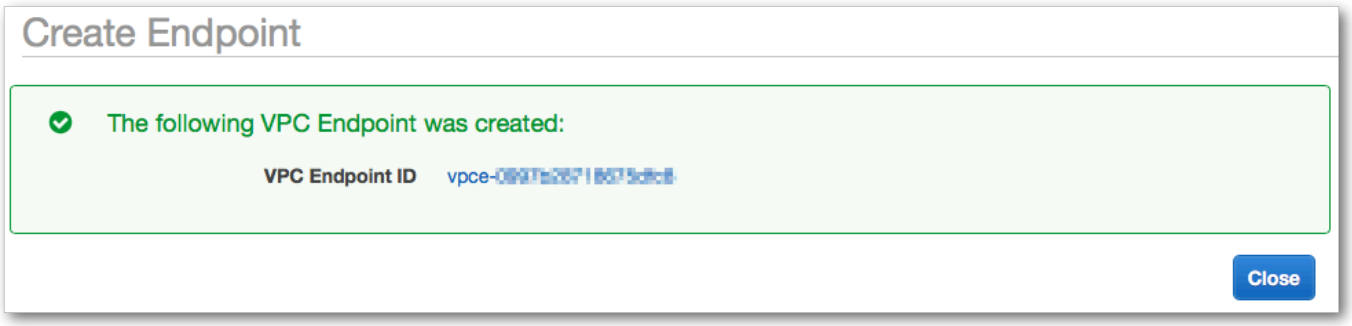
7. Per le sottoreti, scegli la sottorete che ha VPCE-Tutorial-Subnet nell'ID della sottorete.



8. Per Abilita nome privato, seleziona Abilita per questo endpoint. DNS
9. Per Gruppo di sicurezza, scegli Seleziona gruppo di sicurezza e scegli VPCE-Tutorial-. SecurityGroup

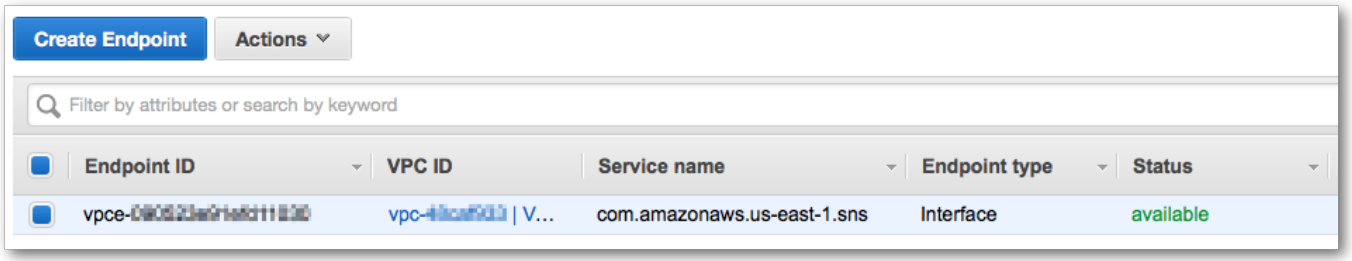


10. Seleziona Crea endpoint. La VPC console Amazon conferma che è stato creato un VPC endpoint.



11. Scegli Chiudi.

    La VPC console Amazon 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).



Passaggio 5: pubblica un messaggio sul tuo SNS argomento Amazon

Ora che hai VPC incluso un endpoint per AmazonSNS, puoi accedere all'EC2istanza Amazon e pubblicare messaggi sull'argomento.

Per pubblicare un messaggio

1. Se il tuo terminale non è più connesso alla tua EC2 istanza Amazon, riconnettiti:

   ```
   $ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
   ```

2. Esegui lo stesso comando che hai usato in precedenza per pubblicare un messaggio sul tuo SNS argomento Amazon. Questa volta, il tentativo di pubblicazione ha esito positivo 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"
   }
   ```

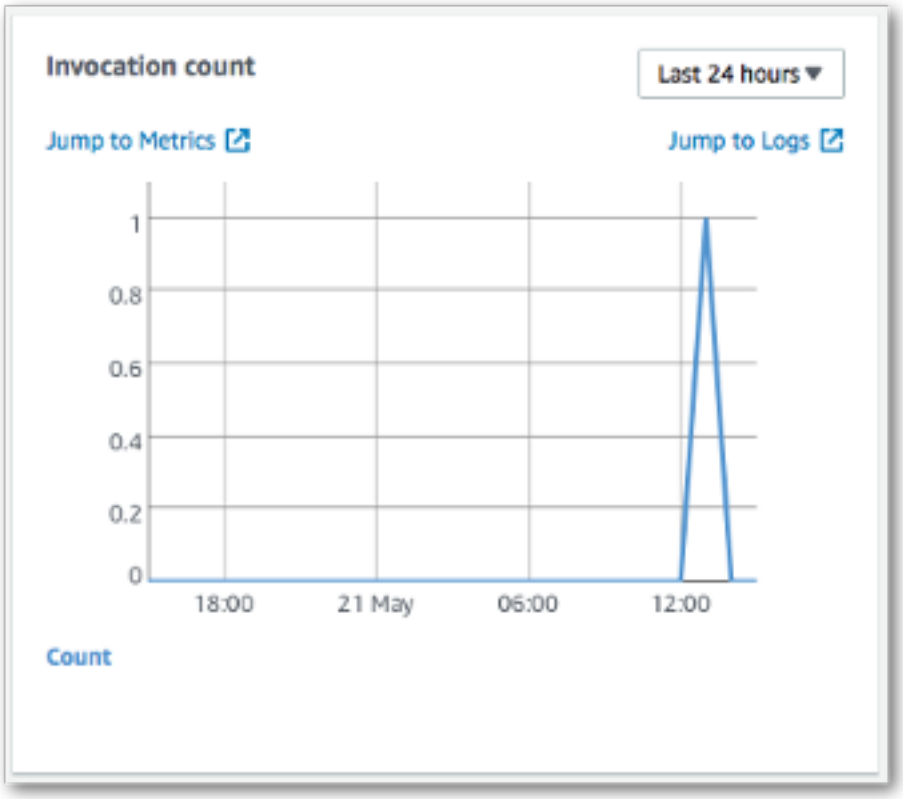
Fase 6: verifica delle consegne del messaggio

Quando l'SNSargomento Amazon riceve un messaggio, lo distribuisce inviandolo alle due funzioni Lambda sottoscritte. Quando queste funzioni ricevono il messaggio, registrano l'evento nei log. CloudWatch Per verificare che il recapito del messaggio sia avvenuto correttamente, verifica che le funzioni siano state richiamate e verifica che i CloudWatch registri siano stati aggiornati.

Per verificare che le funzioni Lambda; siano state richiamate

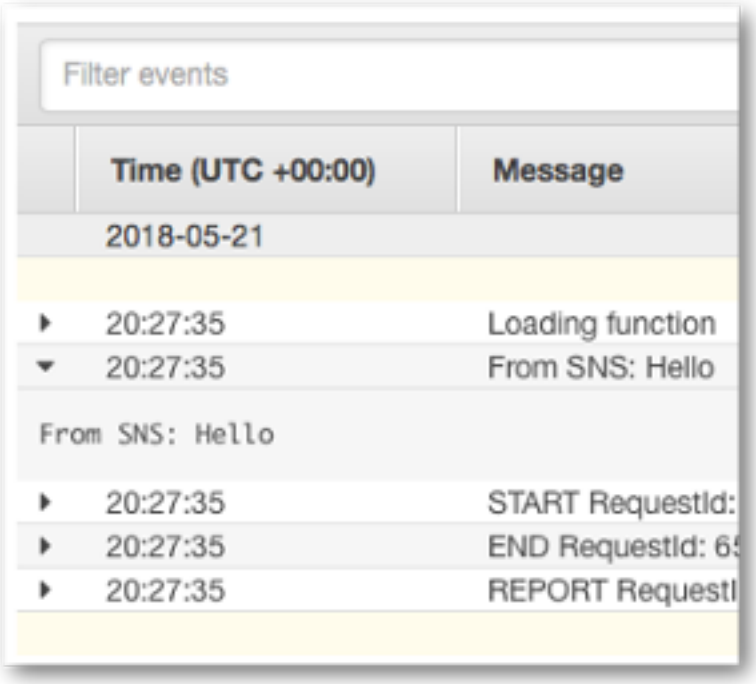
1. Apri la console all' AWS Lambda indirizzo. https://console.aws.amazon.com/lambda/
2. Nella pagina Funzioni, scegli VPCE-Tutorial-Lambda-1.
3. Selezionare Monitoring (Monitoraggio).
4. Controlla il grafico Invocation count (Conteggio invocazioni). Questo grafico mostra il numero di volte che la funzione 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 registri siano stati aggiornati CloudWatch

1. Apri la CloudWatch console 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 i passaggi precedenti per the /aws/lambda/VPCE - Tutorial-Lambda-2/ log group.

Complimenti! Aggiungendo un endpoint per Amazon SNS aVPC, sei stato in grado di pubblicare un messaggio su un argomento dall'interno della rete gestita da. VPC Il messaggio è stato pubblicato privatamente senza essere esposto sulla rete Internet pubblica.

Fase 7: pulire

Se non si desidera conservare le risorse create, è possibile eliminarle ora. Eliminando AWS le risorse che non utilizzi più, eviti addebiti inutili ai tuoi. Account AWS

Innanzitutto, elimina il tuo VPC endpoint utilizzando la VPC console Amazon. Quindi, elimina le altre risorse che hai creato eliminando lo stack nella console. AWS CloudFormation Quando elimini uno stack, AWS CloudFormation rimuove le risorse dello stack dal tuo. Account AWS

Per eliminare il tuo endpoint VPC

1. Apri la VPC console Amazon 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 il tuo AWS CloudFormation stack

1. Apri la AWS CloudFormation console all'indirizzo https://console.aws.amazon.com/cloudformazione.
2. Seleziona lo stack -Tutorial-Stack. VPCE
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 cambia in _IN_. DELETE PROGRESS Quando l'eliminazione viene completata, lo stack viene rimosso dalla pagina.

Risorse correlate

Per ulteriori informazioni, consulta le risorse seguenti.

- AWS Blog sulla sicurezza: Protezione dei messaggi pubblicati su Amazon SNS con AWS PrivateLink
- Che cos'è AmazonVPC?
- VPCEndpoint
- Che cos'è AmazonEC2?
- Concetti di AWS CloudFormation

Migliorare la SNS sicurezza di Amazon con Message Data Protection

- Message Data Protection è una funzionalità di Amazon SNS utilizzata per definire regole e politiche personalizzate per verificare e controllare il contenuto dei dati in movimento, anziché dei dati inattivi.
- Message Data Protection fornisce servizi di governance, conformità e controllo per applicazioni aziendali incentrate sui messaggi, in modo che l'ingresso e l'uscita dei dati possano essere

controllati dal proprietario dell'SNSargomento Amazon 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.

Gestione delle identità e degli accessi in Amazon SNS

L'accesso ad Amazon SNS richiede credenziali che AWS possono essere utilizzate per autenticare le tue richieste. Queste credenziali devono disporre delle autorizzazioni per accedere a AWS risorse, ad esempio SNS argomenti e messaggi di Amazon. Le seguenti sezioni forniscono dettagli su come utilizzare AWS Identity and Access Management (IAM) e Amazon SNS per proteggere le risorse controllandone l'accesso.

AWS Identity and Access Management (IAM) è un programma Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle AWS risorse. IAMgli amministratori controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (disporre delle autorizzazioni) a utilizzare le risorse Amazon. SNS IAMè un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Destinatari

Il modo in cui usi AWS Identity and Access Management (IAM) varia a seconda del lavoro che svolgi in AmazonSNS.

Utente del servizio: se utilizzi il SNS servizio Amazon per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più SNS funzionalità di Amazon per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di AmazonSNS, consultaRisoluzione dei problemi di identità e accesso ad Amazon Simple Notification Service.

Amministratore del servizio: se sei responsabile delle SNS risorse Amazon della tua azienda, probabilmente hai pieno accesso ad AmazonSNS. È tuo compito determinare a quali SNS funzionalità e risorse di Amazon devono accedere gli utenti del servizio. Devi quindi inviare richieste all'IAMamministratore per modificare le autorizzazioni degli utenti del servizio. Consulta

le informazioni contenute in questa pagina per comprendere i concetti di base diIAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM AmazonSNS, consultaCome SNS funziona Amazon con IAM.

IAMamministratore: se sei un IAM amministratore, potresti voler saperne di più su come scrivere politiche per gestire l'accesso ad AmazonSNS. Per visualizzare esempi di policy SNS basate sull'identità di Amazon che puoi utilizzare, consultaIAM. Esempi di policy basate su identità per Amazon Simple Notification Service

Autenticazione con identità

L'autenticazione è il modo in cui accedi AWS utilizzando le tue credenziali di identità. È necessario autenticarsi (accedere a AWS) come utente Account AWS root, come IAM utente o assumendo un ruolo. IAM

È possibile accedere AWS come identità federata utilizzando le credenziali fornite tramite una fonte di identità. AWS IAM Identity Center Gli utenti (IAMIdentity Center), l'autenticazione Single Sign-On della tua azienda e le tue credenziali di Google o Facebook sono esempi di identità federate. Quando accedi come identità federata, l'amministratore aveva precedentemente configurato la federazione delle identità utilizzando i ruoli. IAM Quando si accede AWS utilizzando la federazione, si assume indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere al AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente.

Se accedi a AWS livello di codice, AWS fornisce un kit di sviluppo software (SDK) e un'interfaccia a riga di comando () per firmare crittograficamente le tue richieste utilizzando le tue credenziali. CLI Se non utilizzi AWS strumenti, devi firmare tu stesso le richieste. Per ulteriori informazioni sull'utilizzo del metodo consigliato per firmare autonomamente le richieste, consulta AWS Signature Version 4 per API le richieste nella Guida per l'IAMutente.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, ti AWS consiglia di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'AWS IAM Identity Center utente e Autenticazione a AWS più fattori IAM nella Guida per l'IAMutente.

Account AWS utente root

Quando si crea un account Account AWS, si inizia con un'identità di accesso che ha accesso completo a tutte Servizi AWS le risorse dell'account. Questa identità è denominata utente Account AWS root ed è accessibile effettuando l'accesso con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per l'elenco completo delle attività che richiedono l'accesso come utente root, consulta Attività che richiedono le credenziali dell'utente root nella Guida per l'IAMutente.

Identità federata

Come procedura consigliata, richiedi agli utenti umani, compresi gli utenti che richiedono l'accesso come amministratore, di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente dell'elenco utenti aziendale, di un provider di identità Web AWS Directory Service, della directory Identity Center o di qualsiasi utente che accede utilizzando le Servizi AWS credenziali fornite tramite un'origine di identità. Quando le identità federate accedono Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. Puoi creare utenti e gruppi in IAM Identity Center oppure puoi connetterti e sincronizzarti con un set di utenti e gruppi nella tua fonte di identità per utilizzarli su tutte le tue applicazioni. Account AWS Per informazioni su IAM Identity Center, vedi Cos'è IAM Identity Center? nella Guida AWS IAM Identity Center per l'utente.

IAM users and groups

Un IAMutente è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Laddove possibile, consigliamo di fare affidamento su credenziali temporanee anziché creare IAM utenti con credenziali a lungo termine come password e chiavi di accesso. Tuttavia, se hai casi d'uso specifici che richiedono credenziali a lungo termine con IAM gli utenti, ti consigliamo di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta Ruotare regolarmente le chiavi di accesso per i casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente. IAM

Un IAMgruppo è un'identità che specifica un insieme di utenti. IAM Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio,

potresti avere un gruppo denominato IAMAdminse concedere a quel gruppo le autorizzazioni per IAM amministrare le risorse.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Casi d'uso per IAM gli utenti nella Guida per l'IAMutente.

IAMruoli

Un IAMruolo è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un IAM utente, ma non è associato a una persona specifica. Per assumere temporaneamente un IAM ruolo in AWS Management Console, puoi passare da un utente a un IAM ruolo (console). È possibile assumere un ruolo chiamando un' AWS APIoperazione AWS CLI or o utilizzando un'operazione personalizzataURL. Per ulteriori informazioni sui metodi di utilizzo dei ruoli, vedere Metodi per assumere un ruolo nella Guida per l'IAMutente.

IAMi ruoli 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 informazioni sui ruoli per la federazione, vedere Creazione di un ruolo per un provider di identità di terze parti nella Guida per l'IAMutente. Se utilizzi IAM Identity Center, configuri un set di autorizzazioni. Per controllare a cosa possono accedere le identità dopo l'autenticazione, IAM Identity Center correla il set di autorizzazioni a un ruolo in. IAM Per informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center .
- Autorizzazioni IAM utente temporanee: un IAM utente o un ruolo può assumere il IAM ruolo di assumere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso su più account: puoi utilizzare un IAM ruolo per consentire a qualcuno (un responsabile fidato) di un altro account di accedere alle risorse del tuo account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, con alcuni Servizi AWS, è possibile allegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per conoscere la differenza tra ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta la sezione Accesso alle risorse su più account IAM nella Guida per l'utente. IAM
- Accesso tra servizi: alcuni Servizi AWS utilizzano funzionalità in altri. Servizi AWS Ad esempio, quando effettui una chiamata in un servizio, è normale che quel servizio esegua applicazioni

in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.

- Sessioni di accesso diretto (FAS): quando utilizzi un IAM utente o un ruolo per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. FASle richieste vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.
- Ruolo di servizio: un ruolo di servizio è un IAMruolo che un servizio assume per eseguire azioni per conto dell'utente. Un IAM amministratore può creare, modificare ed eliminare un ruolo di servizio dall'internoIAM. Per ulteriori informazioni, vedere Creazione di un ruolo per delegare le autorizzazioni a un utente Servizio AWS nella Guida per l'IAMutente.
- Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un IAM amministratore può visualizzare, ma non modificare le autorizzazioni per i ruoli collegati al servizio.

- Applicazioni in esecuzione su Amazon EC2: puoi utilizzare un IAM ruolo per gestire le credenziali temporanee per le applicazioni in esecuzione su un'EC2istanza e che effettuano AWS CLI o effettuano AWS API richieste. Ciò è preferibile alla memorizzazione delle chiavi di accesso all'interno dell'EC2istanza. Per assegnare un AWS ruolo a un'EC2istanza e renderlo disponibile per tutte le sue applicazioni, create un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull'EC2istanza di ottenere credenziali temporanee. Per ulteriori informazioni, consulta Usare un IAM ruolo per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon nella Guida per l'IAMutente.

Gestione dell'accesso con policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy è un oggetto AWS che, se associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste politiche quando un principale (utente, utente root o sessione di ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La

maggior parte delle politiche viene archiviata AWS come JSON documenti. Per ulteriori informazioni sulla struttura e il contenuto dei documenti relativi alle JSON politiche, vedere Panoramica delle JSON politiche nella Guida per l'IAMutente.

Gli amministratori possono utilizzare AWS JSON le politiche per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un IAM amministratore può creare IAM politiche. L'amministratore può quindi aggiungere le IAM politiche ai ruoli e gli utenti possono assumerli.

IAMle politiche definiscono le autorizzazioni per un'azione indipendentemente dal metodo utilizzato per eseguire l'operazione. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale criterio può ottenere informazioni sul ruolo da AWS Management Console, da o da. AWS CLI AWS API

Policy basate su identità

I criteri basati sull'identità sono documenti relativi alle politiche di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un IAM utente, un gruppo di utenti o un ruolo. Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una politica basata sull'identità, consulta Definire le IAM autorizzazioni personalizzate con le politiche gestite dal cliente nella Guida per l'utente. IAM

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le politiche gestite sono politiche autonome che puoi allegare a più utenti, gruppi e ruoli all'interno del tuo. Account AWS Le politiche gestite includono politiche AWS gestite e politiche gestite dai clienti. Per informazioni su come scegliere tra una politica gestita o una politica in linea, consulta Scegliere tra politiche gestite e politiche in linea nella Guida per l'IAMutente.

Policy basate su risorse

Le politiche basate sulle risorse sono documenti di JSON policy allegati a una risorsa. Esempi di politiche basate sulle risorse sono le policy di trust dei IAM ruoli e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le 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 AWS gestite contenute IAM in una policy basata sulle risorse.

Elenchi di controllo degli accessi () ACLs

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLssono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy. JSON

Amazon S3 e Amazon VPC sono esempi di servizi che supportano. AWS WAF ACLs Per ulteriori informazioniACLs, consulta la panoramica di Access control list (ACL) nella Amazon Simple Storage Service Developer Guide.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite di autorizzazioni è una funzionalità avanzata in cui si impostano le autorizzazioni massime che una politica basata sull'identità può concedere a un'entità (utente o ruolo). IAM 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 nella Guida per l'utente. IAM IAM
- Politiche di controllo del servizio (SCPs): SCPs sono JSON politiche che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. SCPLimita le autorizzazioni per le entità negli account dei membri, incluso ogni utente Account AWS root. Per ulteriori informazioni su Organizations andSCPs, consulta le politiche di controllo dei servizi nella Guida AWS Organizations per l'utente.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un

utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta le politiche di sessione nella Guida IAM per l'utente.

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 determinare se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle politiche nella Guida per l'IAMutente.

Controllo accessi

Amazon SNS dispone di un proprio sistema di autorizzazioni basato sulle risorse che utilizza politiche scritte nello stesso linguaggio utilizzato per AWS Identity and Access Management le politiche (). IAM Ciò significa che puoi ottenere risultati simili con le SNS politiche e le politiche IAM di Amazon.

> **Note**
>
> È importante capire che tutti Account AWS possono delegare le proprie autorizzazioni agli utenti tramite i propri account. L'accesso tra account consente di condividere l'accesso a risorse AWS senza dover gestire utenti aggiuntivi. Per informazioni sull'utilizzo dell'accesso su più account, consulta Abilitazione dell'accesso su più account nella Guida per l'utente. IAM

Panoramica della 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

- Casi d'uso del controllo degli SNS accessi di Amazon
- Concetti chiave della politica di SNS accesso di Amazon

- Panoramica dell'architettura di controllo degli SNS accessi di Amazon
- Utilizzo dell'Access Policy Language in Amazon SNS
- Logica della valutazione
- Casi di esempio per il controllo degli SNS accessi di Amazon

Casi d'uso del controllo degli SNS accessi di Amazon

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:

- Vuoi concedere a Account AWS un altro un tipo particolare di azione sull'argomento (ad esempio, Pubblica). Per ulteriori informazioni, consulta Concedi Account AWS l'accesso a un argomento.
- Vuoi limitare le sottoscrizioni al tuo argomento solo al HTTPS protocollo. Per ulteriori informazioni, consulta Limita gli abbonamenti a HTTPS.
- Vuoi consentire ad Amazon SNS di pubblicare messaggi nella tua SQS coda Amazon. Per ulteriori informazioni, consulta Pubblica messaggi su una SQS coda Amazon.

Concetti chiave della politica di SNS accesso di Amazon

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

- Autorizzazione
- Dichiarazione
- Policy
- Emittente
- 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) è autorizzata a pubblicare (B) su Topica (C) purché utilizzi il HTTP protocollo (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.

Nei documenti relativi alle ASCII policy sono consentiti solo caratteri. È possibile `aws:SourceOwner` utilizzare `aws:SourceAccount` e aggirare lo scenario in cui è necessario collegare altri AWS servizi ARNs che non contengono caratteri. 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'emittente (per definizione) è sempre il proprietario della risorsa. AWS non consente agli utenti AWS del servizio di creare politiche per risorse che non possiedono. Se John è il proprietario della risorsa, AWS autentica l'identità di John quando invia la politica che ha scritto per concedere le autorizzazioni per quella 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'azione è solo l'operazione inclusa 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 CIDR intervallo particolare)

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. Il AWS servizio stesso (ad esempio Amazon SQS o AmazonSNS) potrebbe anche definire chiavi specifiche del servizio. Per ulteriori informazioni, consulta SNSAPIAutorizzazioni Amazon: riferimento ad azioni 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 AWS servizio per determinare se una richiesta in arrivo debba essere rifiutata o consentita in base alle politiche 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

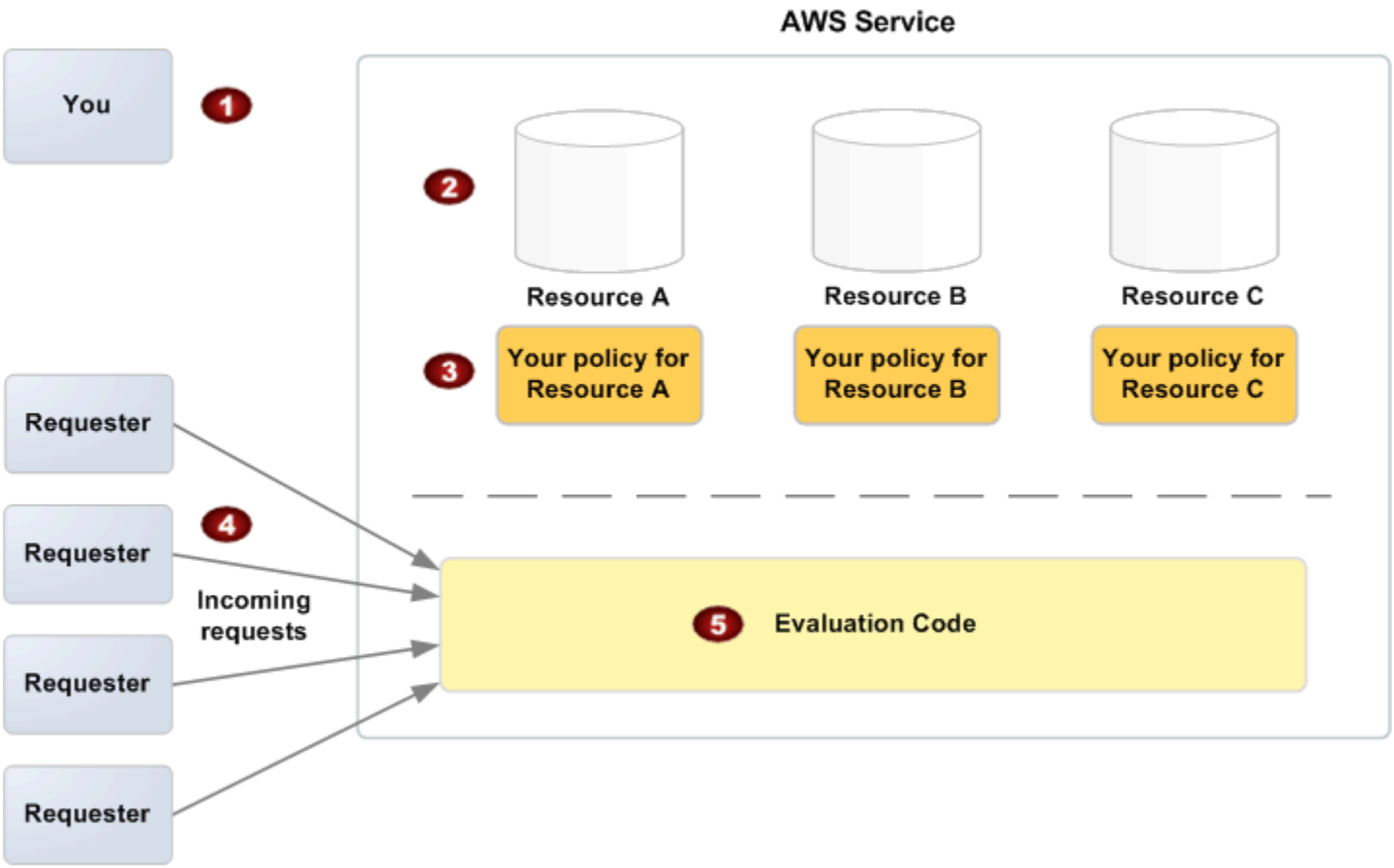
Un consenso viene restituito da una dichiarazione che ha effect=allow, presupponendo che tutte le condizioni dichiarate siano soddisfatte. Esempio: consenti le richieste se vengono ricevute prima delle 13:00 del 30 aprile 2010. Un consenso sovrascrive tutti i rifiuti per default, ma mai un rifiuto esplicito.

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 di controllo degli SNS accessi di Amazon

Nella seguente figura e nella tabella vengono descritti i componenti principali che interagiscono per fornire il controllo accessi alle risorse.

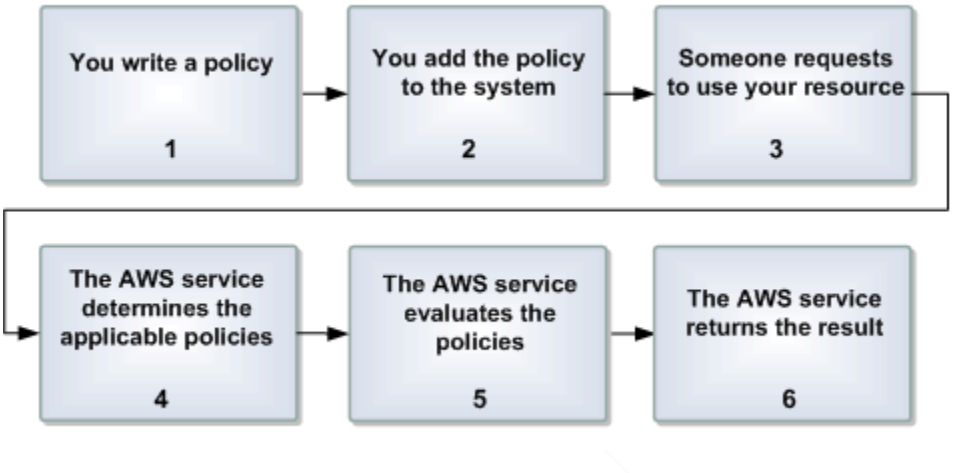


1	Tu, il proprietario delle risorse.
2	Le tue risorse (contenute all'interno del AWS servizio, ad esempio, le SQS code di Amazon).
3	Le tue policy. In genere hai una policy per risorsa, ma possono essere di più. Il AWS servizio stesso ti fornisce uno strumento API che puoi utilizzare per caricare e gestire le tue politiche.
4	Richiedenti e relative richieste in arrivo al AWS servizio.
5	Codice di valutazione del linguaggio della policy di accesso. Questo è l'insieme di codice all'interno del AWS servizio che valuta le richieste in entrata rispetto alle politiche applicabili e determina se al richiedente è consentito l'accesso alla

risorsa. Per informazioni su come il servizio prende le decisioni, vedi Logica della valutazione.

Utilizzo dell'Access Policy Language in Amazon SNS

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 SNS argomenti Amazon.
2	Carichi la tua politica su. AWS Il AWS servizio stesso fornisce un file API che utilizzi per caricare le tue politiche. Ad esempio, utilizzi l'SNS`SetTopicAttributes` azione Amazon per caricare una politica per un particolare SNS argomento di Amazon.
3	Qualcuno invia una richiesta per usare la tua risorsa. Ad esempio, un utente invia una richiesta ad Amazon SNS per utilizzare uno dei tuoi argomenti.
4	Il AWS servizio determina quali politiche sono applicabili alla richiesta.

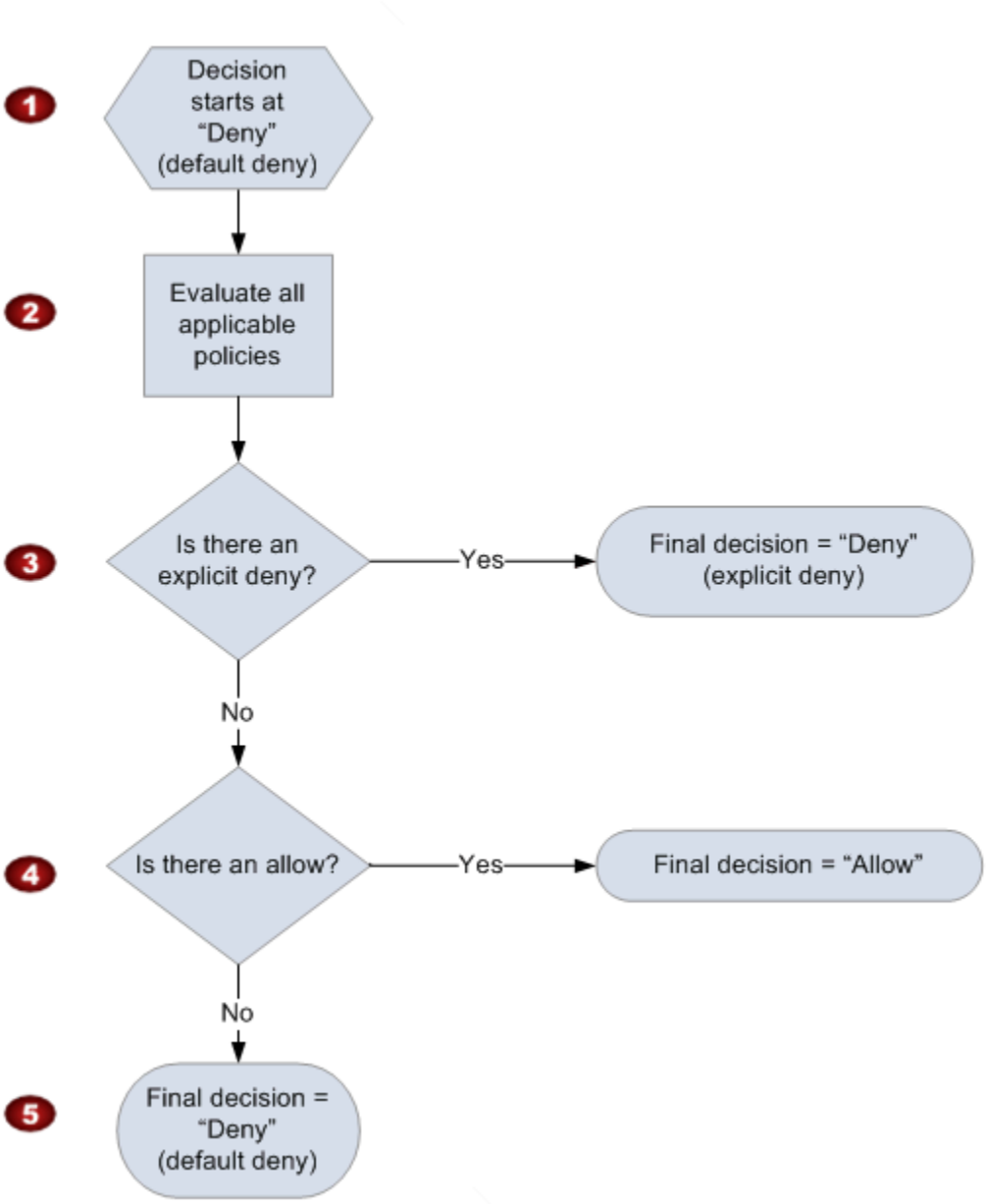
	Ad esempio, Amazon SNS esamina tutte le SNS politiche Amazon disponibili e determina quali sono applicabili (in base alla risorsa, a chi è il richiedente, ecc.).
5	Il AWS servizio valuta le politiche. Ad esempio, Amazon SNS valuta le politiche e determina se il richiedente è autorizzato a utilizzare il tuo argomento o meno. Per ulteriori informazioni sulla logica della decisione, consulta Logica della valutazione.
6	Il AWS servizio rifiuta la richiesta o continua a 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.

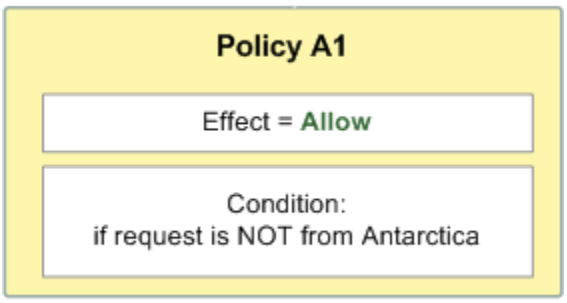
	Se ne trova anche uno, il codice di applicazione restituisce una decisione di "rifiuto" e il processo è finito (questo è un rifiuto esplicito; per ulteriori informazioni, vedi Rifiuto esplicito).
4	Se non viene trovato un rifiuto esplicito, il codice di applicazione cerca un'istruzione di "consenso" 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 AmazonSNS, ma la politica sull'argomento non si riferisce Account AWS affatto a quella dell'utente, tale politica comporta una negazione predefinita.

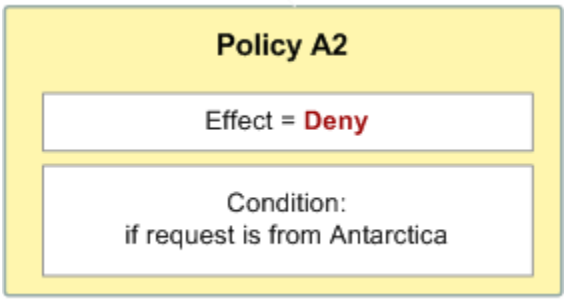
Una policy restituisce un rifiuto per default anche quando una condizione in una dichiarazione non viene soddisfatta. Se tutte le condizioni nella dichiarazione sono soddisfatte, la policy restituisce un consenso o un rifiuto esplicito, a secondo del valore dell'elemento Effetto nella policy. Le policy 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.



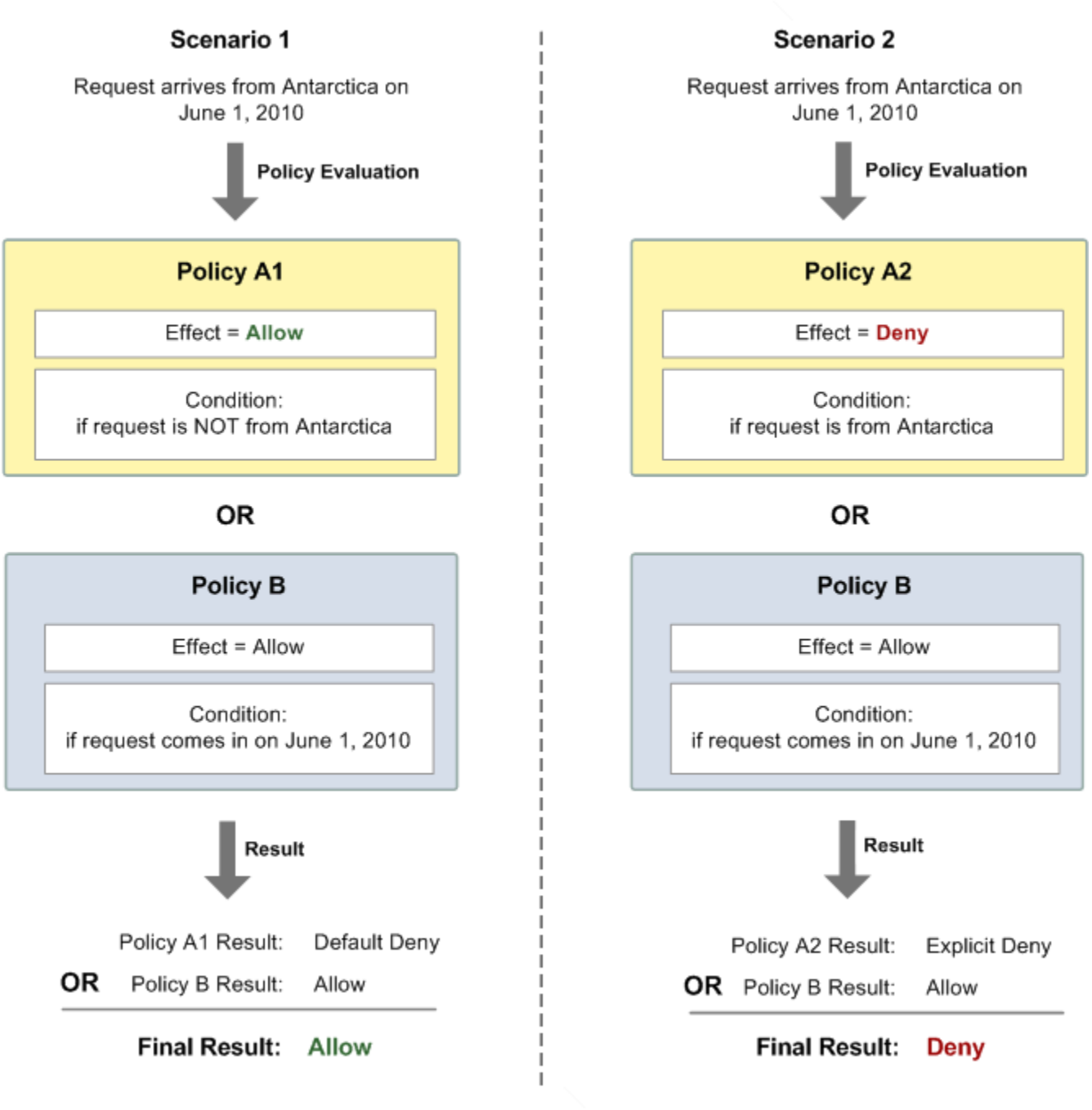
Se qualcuno invia una richiesta dagli Stati Uniti, la condizione è soddisfatta (la richiesta non proviene dall'Antartide). Pertanto, la richiesta è consentita. Tuttavia, se qualcuno invia una richiesta dall'Antartide, la condizione non viene soddisfatta e il risultato della policy è quindi un rifiuto 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.

Casi di esempio per il controllo degli SNS accessi di Amazon

Questa sezione include alcuni esempi di casi d'uso tipici per il controllo accessi.

Argomenti

Concedi Account AWS l'accesso a un argomento

Supponiamo che tu abbia un argomento in Amazon SNS e desideri consentire a uno o più di Account AWS eseguire un'azione specifica su quell'argomento, come la pubblicazione di messaggi. È possibile eseguire questa operazione utilizzando l'SNSAPIazione `AddPermission` Amazon.

L'`AddPermission`azione consente di specificare un argomento, un elenco di Account AWS IDs, un elenco di azioni e un'etichetta. Amazon SNS quindi genera e aggiunge automaticamente una nuova dichiarazione politica alla politica di controllo degli accessi dell'argomento. Non è necessario che tu scriva tu stesso la dichiarazione sulla politica: Amazon se ne SNS occupa per te. Se devi rimuovere la politica in un secondo momento, puoi farlo chiamando `RemovePermission` e fornendo l'etichetta che hai usato per aggiungere l'autorizzazione.

Ad esempio, se `AddPermission` chiami l'argomento arn:aws:sns:us-east-2:444455556666:MyTopic, specifichi l' Account AWS ID 1111-2222-3333, l'`Publish`azione e l'etichetta`grant-1234-publish`, SNS Amazon genererà e inserirà la seguente dichiarazione politica nella politica di controllo degli accessi dell'argomento:

```
{
  "Statement": [{
```

```
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Dopo aver aggiunto questa dichiarazione, il 1111-2222-3333 avrà l'autorizzazione a pubblicare messaggi sull'argomento. Account AWS

Informazioni aggiuntive:

- Gestione dei criteri personalizzati: sebbene `AddPermission` sia utile per concedere le autorizzazioni, è spesso utile gestire manualmente la politica di controllo degli accessi dell'argomento per scenari più complessi, come l'aggiunta di condizioni o la concessione di autorizzazioni a ruoli o servizi specifici. IAM È possibile farlo utilizzando l'attributo per aggiornare direttamente l'`SetTopicAttributes`APIattributo policy.
- Migliori pratiche di sicurezza: fai attenzione quando concedi le autorizzazioni per assicurarti che solo persone attendibili Account AWS o entità abbiano accesso ai tuoi argomenti Amazon. SNS Esamina e verifica regolarmente le politiche allegate ai tuoi argomenti per mantenere la sicurezza.
- Limiti delle politiche: tieni presente che esistono limiti alla dimensione e alla complessità delle SNS politiche di Amazon. Se devi aggiungere molte autorizzazioni o condizioni complesse, assicurati che la tua politica rimanga entro questi limiti.

Limita gli abbonamenti a HTTPS

Per limitare il protocollo di invio delle notifiche per il tuo SNS argomento Amazon aHTTPS, devi creare una politica personalizzata. L'`AddPermission`azione in Amazon SNS non ti consente di specificare restrizioni di protocollo quando concedi l'accesso al tuo argomento. Pertanto, devi scrivere manualmente una politica che applichi questa restrizione e quindi utilizzare l'`SetTopicAttributes`azione per applicare la politica al tuo argomento.

Ecco come puoi creare una politica che limiti gli abbonamenti a: HTTPS

1. Scrivi la politica. La politica deve specificare l' Account AWS ID a cui desideri concedere l'accesso e imporre la condizione che siano consentiti solo HTTPS gli abbonamenti. Di seguito è

riportato un esempio di politica che concede all' Account AWS ID 1111-2222-3333 il permesso di sottoscrivere l'argomento, ma solo se il protocollo utilizzato lo è. HTTPS

```
{
  "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"
      }
    }
  }]
}
```

2. Applica la politica. Usa l'`SetTopicAttributes`azione in Amazon SNS API per applicare questa politica al tuo argomento. Imposta l'`Policy`attributo dell'argomento sulla JSON politica che hai creato.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informazioni aggiuntive:

- Personalizzazione del controllo degli accessi. Questo approccio consente di applicare controlli di accesso più granulari, come la limitazione dei protocolli di abbonamento, cosa che non è possibile con la sola azione. `AddPermission` Le policy personalizzate offrono flessibilità per scenari che richiedono condizioni specifiche, come l'applicazione del protocollo o le restrizioni degli indirizzi IP.
- Le migliori pratiche di sicurezza. Limita gli abbonamenti per HTTPS migliorare la sicurezza delle notifiche assicurando che i dati in transito siano crittografati. Rivedi regolarmente le politiche relative agli argomenti per assicurarti che soddisfino i requisiti di sicurezza e conformità.

- Test delle politiche. Prima di applicare la policy in un ambiente di produzione, testatela in un ambiente di sviluppo per assicurarvi che si comporti come previsto. Questo aiuta a prevenire problemi di accesso accidentali o restrizioni non intenzionali.

Pubblica messaggi su una SQS coda Amazon

Per pubblicare messaggi dal tuo SNS argomento Amazon su una SQS coda Amazon, devi configurare le autorizzazioni corrette sulla coda AmazonSQS. Sebbene SNS sia Amazon che Amazon SQS utilizzino il linguaggio delle policy AWS di controllo degli accessi, devi impostare esplicitamente una policy sulla SQS coda Amazon per consentire l'invio di messaggi dall'argomento AmazonSNS.

Puoi raggiungere questo obiettivo utilizzando l'`SetQueueAttributes`azione per applicare una politica personalizzata alla SQS coda di Amazon. A differenza di AmazonSNS, Amazon SQS non supporta l'`AddPermission`azione volta a creare dichiarazioni politiche con condizioni. Pertanto, è necessario scrivere la politica manualmente.

Di seguito è riportato un esempio di SQS politica di Amazon che concede ad Amazon SNS l'autorizzazione a inviare messaggi alla tua coda. Tieni presente che questa politica è associata alla SQS coda Amazon, non all'SNSargomento Amazon. Le azioni specificate sono SQS azioni Amazon e la risorsa è l'Amazon Resource Name (ARN) della coda. Puoi recuperare le code ARN utilizzando l'azione. `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 politica utilizza la `aws:SourceArn` condizione per limitare l'accesso alla SQS coda in base alla fonte dei messaggi inviati. Ciò garantisce che solo i messaggi provenienti dall'SNSargomento specificato (in questo caso, arn:aws:sns:us-east- 2:444455556666:) possano essere recapitati alla coda. MyTopic

Informazioni aggiuntive:

- Coda. ARN Assicurati di recuperare la SQS coda corretta ARN di Amazon utilizzando l'azione`GetQueueAttributes`. Questo ARN è essenziale per impostare le autorizzazioni corrette.
- Le migliori pratiche di sicurezza. Quando imposti le politiche, segui sempre il principio del privilegio minimo. Concedi solo le autorizzazioni necessarie all'SNSargomento Amazon per interagire con la SQS coda Amazon e rivedi regolarmente le tue politiche per assicurarti che siano sicure up-to-date
- Politiche predefinite in AmazonSNS. Contrariamente ad alcuni malintesi, Amazon SNS non concede automaticamente una politica predefinita che consenta ad altri Servizi AWS l'accesso agli argomenti appena creati. Devi definire e allegare in modo esplicito le policy per controllare l'accesso ai tuoi SNS argomenti Amazon.
- Test e convalida. Dopo aver impostato la politica, verifica l'integrazione pubblicando i messaggi SNS sull'argomento Amazon e verificando che vengano recapitati correttamente alla SQS coda Amazon. Questo aiuta a confermare che la policy è configurata correttamente.

Consentire la pubblicazione delle notifiche degli eventi Amazon S3 su un argomento

Per consentire a un bucket Amazon S3 di un altro di Account AWS pubblicare notifiche di eventi sul tuo SNS argomento Amazon, devi configurare di conseguenza la politica di accesso dell'argomento. Ciò comporta la stesura di una policy personalizzata che conceda l'autorizzazione al servizio Amazon S3 dall'utente Account AWS specifico e quindi l'applicazione di questa politica all'argomento trattato su AmazonSNS.

Ecco come puoi configurarlo:

1. Scrivi la politica. La policy dovrebbe garantire il servizio Amazon S3 (s3.amazonaws.com) le autorizzazioni necessarie per pubblicare sul tuo SNS argomento Amazon. Utilizzerai la `SourceAccount` condizione per assicurarti che solo l'utente specificato Account AWS, che possiede il bucket Amazon S3, possa pubblicare notifiche sul tuo argomento.

   Di seguito è riportato un esempio di policy:

   ```
   {
   ```

```
  "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"
      }
    }
  }]
}
```

- Proprietario dell'argomento: 111122223333 è l' Account AWS ID proprietario dell'argomento AmazonSNS.
- Proprietario del bucket Amazon S3:444455556666 è l' Account AWS ID proprietario del bucket Amazon S3 per l'invio delle notifiche.

2. Applica la policy. Utilizza l'`SetTopicAttributes`azione per impostare questa politica sul tuo SNS argomento Amazon. Ciò aggiornerà il controllo degli accessi dell'argomento per includere le autorizzazioni specificate nella politica personalizzata.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informazioni aggiuntive:

- **`SourceAccount`**Condizione d'uso. La `SourceAccount` condizione garantisce che solo gli eventi provenienti da quanto specificato Account AWS (444455556666 in questo caso) possano attivare l'argomento Amazon. SNS Questa è una misura di sicurezza per impedire ad account non autorizzati di inviare notifiche al tuo argomento.
- Altri servizi di supporto**`SourceAccount`**. La `SourceAccount` condizione è supportata dai seguenti servizi. È fondamentale utilizzare questa condizione quando desideri limitare l'accesso al tuo SNS argomento Amazon in base all'account di origine.

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon EventBridge
- Amazon GameLift
- Amazon Pinpoint SMS e Voice API
- 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

- Test e convalida. Dopo aver applicato la policy, verifica la configurazione attivando un evento nel bucket Amazon S3 e confermando che sia stato pubblicato correttamente sul tuo argomento Amazon. SNS Ciò contribuirà a garantire che la policy sia configurata correttamente.
- Le migliori pratiche di sicurezza. Esamina e verifica regolarmente le politiche SNS tematiche di Amazon per assicurarti che siano conformi ai tuoi requisiti di sicurezza. Limitare l'accesso solo ad account e servizi affidabili è essenziale per mantenere operazioni sicure.

Consenti SES ad Amazon di pubblicare su un argomento di proprietà di un altro account

È possibile consentire Servizio AWS a un altro di pubblicare su un argomento di proprietà di un altro Account AWS. Supponiamo che tu abbia effettuato l'accesso all'account 111122223333SES, aperto Amazon e creato un'e-mail. Per pubblicare notifiche su questa e-mail su un SNS argomento Amazon di proprietà dell'account 444455556666, devi creare una politica come la seguente. A tale scopo, è necessario fornire informazioni sull'entità principale (l'altro servizio) e sulla proprietà di ciascuna risorsa. L'`Resource`istruzione fornisce l'argomentoARN, che include l'ID account 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 AmazonSNS, sono supportati i seguenti servizi`aws:SourceOwner`:

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon GameLift
- Amazon Pinpoint SMS e Voice API
- 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` `aws:SourceAccount` Amazon mantiene SNS ancora la compatibilità con le versioni precedenti per i servizi esistenti attualmente supportati`aws:SourceOwner`.

Le chiavi `aws:SourceAccount` e `aws:SourceOwner` condition vengono impostate ciascuna da alcuni Servizi AWS quando vengono pubblicate su un SNS argomento di Amazon. Se supportato, il valore sarà l'ID dell'account a 12 cifre per AWS conto del quale il servizio pubblica i dati. Alcuni servizi supportano uno e altri supportano l'altro.

- Scopri Consentire la pubblicazione delle notifiche degli eventi Amazon S3 su un argomento come vengono utilizzate le notifiche di Amazon S3 `aws:SourceAccount` e un elenco di AWS servizi che supportano tale condizione.
- Scopri Consenti SES ad Amazon di pubblicare su un argomento di proprietà di un altro account come Amazon SES utilizza `aws:SourceOwner` e un elenco di AWS servizi che supportano tale condizione.

Consenti agli account di un'organizzazione AWS Organizations di pubblicare su un argomento in un altro account

Il AWS Organizations servizio ti aiuta a gestire centralmente la fatturazione, a controllare l'accesso e la sicurezza e a 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 organizzazione `myOrgId` può pubblicare su Amazon l'SNSargomento `MyTopic` dell'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 account diverso

In questo caso, tutti gli CloudWatch allarmi `111122223333` presenti nell'account possono essere pubblicati su un SNS argomento Amazon nell'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:*"
                }
            }
        }
    ]
}
```

Limita la pubblicazione a un SNS argomento Amazon solo da un VPC endpoint specifico

In questo caso, l'argomento nell'account 444455556666 può essere pubblicato solo dall'VPCendpoint 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 SNS funziona Amazon con IAM

Prima di utilizzare IAM per gestire l'accesso ad AmazonSNS, scopri quali IAM funzionalità sono disponibili per l'uso con AmazonSNS.

IAMfunzionalità che puoi utilizzare con Amazon Simple Notification Service

IAMfunzionalità	SNSAssistenza Amazon
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
ACLs	No
ABAC(tag nelle politiche)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì

IAMfunzionalità	SNSAssistenza Amazon
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per avere una panoramica generale di come Amazon SNS e altri AWS servizi funzionano con la maggior parte delle IAM funzionalità, consulta AWS i servizi con cui funzionano IAM nella Guida per l'IAMutente.

AWS politiche gestite per Amazon Simple Notification Service

Una politica AWS gestita è una politica autonoma creata e amministrata da AWS. AWS le politiche gestite sono progettate per fornire autorizzazioni per molti casi d'uso comuni, in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Tieni presente che le policy AWS gestite potrebbero non concedere le autorizzazioni con il privilegio minimo per i tuoi casi d'uso specifici, poiché sono disponibili per tutti i clienti. AWS Ti consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. AWS Se AWS aggiorna le autorizzazioni definite in una politica AWS gestita, l'aggiornamento ha effetto su tutte le identità principali (utenti, gruppi e ruoli) a cui è associata la politica. AWS è più probabile che aggiorni una policy AWS gestita quando ne Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove API operazioni per i servizi esistenti.

Per ulteriori informazioni, consulta le politiche AWS gestite nella Guida IAM per l'utente.

AWS politica gestita: A mazonSNSFull Access

`AmazonSNSFullAccess`fornisce l'accesso completo ad Amazon SNS utilizzando AWS Management Console. Questa politica include anche le seguenti azioni di lettura e scrittura per AWS End User Messaging SMS le chiamate tramite AmazonSNS. Puoi allegare questa politica ai tuoi utenti, gruppi o ruoli.

Dettagli dell'autorizzazione

Le seguenti autorizzazioni si applicano solo quando si utilizza Amazon SNSAPIs:

- `sns:*`— Consente le autorizzazioni complete per eseguire qualsiasi azione relativa ad AmazonSNS. Questo carattere jolly (*) indica che l'utente può eseguire tutte le SNS azioni Amazon possibili.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Consente di recuperare un elenco di numeri di telefono che sono stati verificati per l'invio di SMS messaggi all'interno di. Account AWS
- `sms-voice:CreateVerifiedDestinationNumber`— Consente di verificare un nuovo numero di telefono da utilizzare con i servizi SMS di messaggistica di. AWS
- `sms-voice:SendDestinationNumberVerificationCode`— Consente di inviare un codice di verifica a un numero di telefono in fase di verifica per la SMS messaggistica interna AWS.
- `sms-voice:SendTextMessage`— Consente di creare un nuovo messaggio di testo e inviarlo al numero di telefono del destinatario. `SendTextMessage`invia un SMS messaggio a un solo destinatario ogni volta che viene richiamato.
- `sms-voice:DeleteVerifiedDestinationNumber`— Consente di rimuovere un numero di telefono dall'elenco dei numeri verificati all'interno di Account AWS
- `sms-voice:VerifyDestinationNumber`— Consente di avviare e completare il processo di verifica di un numero di telefono da utilizzare per i servizi di SMS messaggistica all'interno AWS di.
- `sms-voice:DescribeAccountAttributes`— Consente di recuperare informazioni dettagliate sugli attributi a livello di account relativi ai SMS servizi di messaggistica all'interno. AWS
- `sms-voice:DescribeSpendLimits`— Consente di recuperare informazioni sui limiti di spesa associati ai servizi di messaggistica all'interno di SMS Account AWS
- `sms-voice:DescribePhoneNumbers`— Consente di recuperare informazioni dettagliate sui numeri di telefono associati ai servizi di SMS messaggistica all'interno di Account AWS
- `sms-voice:SetTextMessageSpendLimitOverride`— Consente di impostare o ignorare il limite di spesa per i messaggi di SMS testo all'interno di Account AWS
- `sms-voice:DescribeOptedOutNumbers`— Consente di recuperare un elenco di numeri di telefono che hanno scelto di non ricevere SMS messaggi dal proprio account. AWS
- `sms-voice:DeleteOptedOutNumber`— Consente di rimuovere un numero di telefono dall'elenco dei numeri esclusi all'interno del Account AWS

`AmazonSNSFullAccess`esempio di politica

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSFullAccess",
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:CreateVerifiedDestinationNumber",
                "sms-voice:SendDestinationNumberVerificationCode",
                "sms-voice:SendTextMessage",
                "sms-voice:DeleteVerifiedDestinationNumber",
                "sms-voice:VerifyDestinationNumber",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:SetTextMessageSpendLimitOverride",
                "sms-voice:DescribeOptedOutNumbers",
                "sms-voice:DeleteOptedOutNumber"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Per visualizzare le autorizzazioni per questa politica, vedere A mazonSNSFull Access nel AWS Managed Policy Reference.

AWS politica gestita: A mazonSNSRead OnlyAccess

`AmazonSNSReadOnlyAccess`fornisce accesso in sola lettura ad Amazon SNS utilizzando. AWS Management Console Questa politica include anche le seguenti azioni di sola lettura per AWS End

User Messaging SMS le chiamate tramite Amazon. SNS Puoi allegare questa politica ai tuoi utenti, gruppi e ruoli.

Dettagli dell'autorizzazione

Le seguenti autorizzazioni si applicano solo quando si utilizza Amazon SNSAPIs:

- `sns:GetTopicAttributes`— Consente di recuperare gli attributi di un SNS argomento Amazon. Ciò include informazioni come l'argomento ARN (Amazon Resource Name), l'elenco degli abbonati, le politiche di consegna, le politiche di controllo degli accessi e qualsiasi altro metadato associato all'argomento.
- `sns:List*`— Consente di eseguire qualsiasi operazione a partire da `List` per SNS le risorse Amazon. Ciò include le autorizzazioni per elencare vari elementi relativi ad AmazonSNS, come:
  - `sns:ListTopics`— Consente di recuperare un elenco di tutti gli SNS argomenti Amazon presenti in. Account AWS
  - `sns:ListSubscriptions`— Consente di recuperare un elenco di tutti gli abbonamenti agli argomenti di AmazonSNS.
  - `sns:ListSubscriptionsByTopic`— Consente di elencare tutti gli abbonamenti per un SNS argomento Amazon specifico.
  - `sns:ListPlatformApplications`— Consente di elencare tutte le applicazioni della piattaforma create per le notifiche push mobili.
  - `sns:ListEndpointsByPlatformApplication`— Consente di elencare tutti gli endpoint associati a un'applicazione della piattaforma.
- `sns:CheckIfPhoneNumberIsOptedOut`— Consente di verificare se un numero di telefono specifico ha scelto di non ricevere SMS messaggi tramite AmazonSNS.
- `sns:GetEndpointAttributes`— Consente di recuperare gli attributi di un endpoint associato a un'applicazione della SNS piattaforma Amazon. Ciò potrebbe includere attributi come lo stato di attivazione dell'endpoint, i dati utente personalizzati e qualsiasi altro metadato associato all'endpoint.
- `sns:GetDataProtectionPolicy`— Consente di recuperare la politica di protezione dei dati associata a un SNS argomento di Amazon.
- `sns:GetPlatformApplicationAttributes`— Consente di recuperare gli attributi di un'applicazione della SNS piattaforma Amazon. Le applicazioni della piattaforma vengono utilizzate in Amazon SNS per inviare notifiche push ai dispositivi mobili tramite servizi come Apple Push Notification Service (APNS) o Firebase Cloud Messaging (FCM).

- `sns:GetSMSAttributes`— Consente di recuperare le SMS impostazioni predefinite per. Account AWS
- `sns:GetSMSSandboxAccountStatus`— Consente di recuperare lo stato corrente della SMS sandbox per il tuo. Account AWS
- `sns:GetSubscriptionAttributes`— Consente di recuperare gli attributi di un abbonamento specifico a un SNS argomento di Amazon.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Consente di visualizzare o recuperare un elenco di numeri di telefono che sono stati verificati per l'invio di SMS messaggi all'interno del Account AWS
- `sms-voice:DescribeAccountAttributes`— Consente di visualizzare o recuperare informazioni sugli attributi a livello di account relativi ai servizi di messaggistica all'interno diSMS. AWS
- `sms-voice:DescribeSpendLimits`— Consente di visualizzare o recuperare informazioni sui limiti di spesa associati SMS ai servizi di messaggistica all'interno del Account AWS
- `sms-voice:DescribePhoneNumbers`— Consente di visualizzare o recuperare informazioni sui numeri di telefono utilizzati per i servizi di SMS messaggistica all'interno di Account AWS
- `sms-voice:DescribeOptedOutNumbers`— Consente di visualizzare o recuperare un elenco di numeri di telefono che hanno scelto di non ricevere SMS messaggi dal Account AWS

`AmazonSNSReadOnlyAccess`esempio di politica

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "sns:GetTopicAttributes",
                "sns:List*",
                "sns:CheckIfPhoneNumberIsOptedOut",
                "sns:GetEndpointAttributes",
                "sns:GetDataProtectionPolicy",
                "sns:GetPlatformApplicationAttributes",
                "sns:GetSMSAttributes",
                "sns:GetSMSSandboxAccountStatus",
                "sns:GetSubscriptionAttributes"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:DescribeOptedOutNumbers"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Per visualizzare le autorizzazioni per questa politica, vedere A mazonSNSFull Access nel AWS Managed Policy Reference.

SNSAggiornamenti Amazon alle politiche AWS gestite

Visualizza i dettagli sugli aggiornamenti delle politiche AWS gestite per Amazon SNS da quando questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al RSS feed nella pagina della cronologia di Amazon SNS Document.

Modifica	Descrizione	Data
AmazonSNSFullAccess: aggiornamento a una policy esistente	Amazon SNS ha aggiunto nuove autorizzazioni per consentire l'accesso completo ad Amazon SNS utilizzando. AWS Management Console	24/09/2024

Modifica	Descrizione	Data
A mazonSNSRead OnlyAcces s — Aggiornamento a una politica esistente	Amazon SNS ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura ad Amazon SNS utilizzando. AWS Managemen t Console	24/09/2024
Amazon SNS ha iniziato a tracciare le modifiche	Amazon SNS ha iniziato a tracciare le modifiche alle sue politiche AWS gestite.	27/08/2024

Azioni politiche per Amazon SNS

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare AWS JSON le politiche per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'`Action`elemento di una JSON policy descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a una policy. Le azioni politiche in genere hanno lo stesso nome dell' AWS APIoperazione associata. Esistono alcune eccezioni, come le azioni basate solo sulle autorizzazioni che non hanno un'operazione corrispondente. API 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 delle SNS azioni di Amazon, consulta Resources Defined by Amazon Simple Notification Service nel Service Authorization Reference.

Le azioni politiche in Amazon SNS utilizzano il seguente prefisso prima dell'azione:

```
sns
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "sns:action1",
```

```
      "sns:action2"
         ]
```

Per visualizzare esempi di politiche SNS basate sull'identità di Amazon, consulta. Esempi di policy basate su identità per Amazon Simple Notification Service

Risorse relative alle policy per Amazon SNS

Supporta le risorse di policy: sì

Gli amministratori possono utilizzare AWS JSON le politiche per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Resource` JSON policy specifica l'oggetto o gli oggetti a cui si applica l'azione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il relativo Amazon Resource Name (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 dei tipi di SNS risorse Amazon e relativiARNs, consulta Actions Defined by Amazon Simple Notification Service nel Service Authorization Reference. Per sapere con quali azioni puoi specificare il tipo ARN di ciascuna risorsa, consulta Resources Defined by Amazon Simple Notification Service.

Per visualizzare esempi di politiche SNS basate sull'identità di Amazon, consulta. Esempi di policy basate su identità per Amazon Simple Notification Service

Chiavi relative alle condizioni delle politiche per Amazon SNS

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare AWS JSON le policy per specificare chi ha accesso a cosa. 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 si specificano più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione logica`OR`. 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, è possibile concedere a un IAM utente l'autorizzazione ad accedere a una risorsa solo se è contrassegnata con il suo nome IAM utente. Per ulteriori informazioni, consulta gli elementi IAM della politica: variabili e tag nella Guida IAM per l'utente.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche del servizio. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'IAMutente.

Per visualizzare un elenco di chiavi di SNS condizione di Amazon, consulta Condition Keys for Amazon Simple Notification Service nel Service Authorization Reference. 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 politiche SNS basate sull'identità di Amazon, consulta. Esempi di policy basate su identità per Amazon Simple Notification Service

ACLsin Amazon SNS

SupportiACLs: no

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri dell'account, utenti o ruoli) dispongono delle autorizzazioni per accedere a una risorsa. ACLssono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy. JSON

ABACcon Amazon SNS

Supporti ABAC (tag nelle politiche): parziale

Il controllo degli accessi basato sugli attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, questi attributi sono chiamati tag. È possibile allegare tag a IAM entità (utenti o ruoli) e a molte AWS risorse. L'etichettatura di entità e risorse è il primo passo diABAC. Quindi si progettano ABAC politiche per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa a cui sta tentando di accedere.

ABACè utile in ambienti in rapida crescita e aiuta in situazioni in cui la gestione delle politiche diventa complicata.

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 in meritoABAC, vedere Definizione delle autorizzazioni con ABAC autorizzazione nella Guida per l'IAMutente. Per visualizzare un tutorial con i passaggi per la configurazioneABAC, consulta Use Attribute-based access control (ABAC) nella Guida per l'utente. 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, incluse quelle che Servizi AWS funzionano con credenziali temporanee, consulta la sezione Servizi AWS relativa alla funzionalità IAM nella Guida per l'IAMutente.

Si utilizzano credenziali temporanee se si accede AWS Management Console utilizzando qualsiasi metodo tranne il nome utente e la password. Ad esempio, quando accedete AWS utilizzando il link Single Sign-on (SSO) della vostra azienda, tale processo crea automaticamente credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sul cambio di ruolo, consulta Passare da un utente a un IAM ruolo (console) nella Guida per l'IAMutente.

È possibile creare manualmente credenziali temporanee utilizzando AWS CLI o AWS API. È quindi possibile utilizzare tali credenziali temporanee per accedere. AWS AWS consiglia di generare

dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, vedere Credenziali di sicurezza temporanee in. IAM

Autorizzazioni principali multiservizio per Amazon SNS

Supporta sessioni di accesso diretto (FAS): Sì

Quando utilizzi un IAM utente o un ruolo per eseguire azioni AWS, sei considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FASutilizza le autorizzazioni del principale che chiama an Servizio AWS, in combinazione con la richiesta Servizio AWS per effettuare richieste ai servizi downstream. FASle richieste vengono effettuate solo quando un servizio riceve una richiesta che richiede interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli FAS delle politiche relative alle richieste, consulta Forward access sessions.

Ruoli di servizio per Amazon SNS

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un IAMruolo che un servizio assume per eseguire azioni per conto dell'utente. Un IAM amministratore può creare, modificare ed eliminare un ruolo di servizio dall'internoIAM. Per ulteriori informazioni, vedere Creazione di un ruolo per delegare le autorizzazioni a un utente Servizio AWS nella Guida per l'IAMutente.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la SNS funzionalità di Amazon. Modifica i ruoli di servizio solo quando Amazon SNS fornisce indicazioni in tal senso.

Ruoli collegati ai servizi per Amazon SNS

Supporta i ruoli collegati ai servizi: no

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un IAM amministratore può visualizzare, ma non modificare le autorizzazioni per i ruoli collegati al servizio.

Per informazioni dettagliate sulla creazione o la gestione di ruoli collegati ai servizi, consulta AWS Servizi compatibili con. 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 sono autorizzati a creare o modificare SNS risorse Amazon. Inoltre, non possono eseguire attività utilizzando AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS API. Per concedere agli utenti il permesso di eseguire azioni sulle risorse di cui hanno bisogno, un IAM amministratore può creare IAM policy. L'amministratore può quindi aggiungere le IAM politiche ai ruoli e gli utenti possono assumerli.

Per informazioni su come creare una politica IAM basata sull'identità utilizzando questi documenti di esempio, consulta Create JSON IAM policy (console) nella Guida per l'IAMutente.

Per dettagli sulle azioni e sui tipi di risorse definiti da AmazonSNS, incluso il formato di ARNs per ogni tipo di risorsa, consulta Actions, Resources and Condition Keys for Amazon Simple Notification Service nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Utilizzo della SNS console Amazon
- Altri tipi di policy
- Più tipi di policy
- Consentire agli utenti di visualizzare le loro autorizzazioni

Best practice per le policy

Le politiche basate sull'identità determinano se qualcuno può creare, accedere o eliminare SNS risorse Amazon nel tuo account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Inizia con le policy AWS gestite e passa alle autorizzazioni con privilegi minimi: per iniziare a concedere autorizzazioni a utenti e carichi di lavoro, utilizza le politiche gestite che concedono le autorizzazioni per molti casi d'uso comuni.AWS Sono disponibili nel tuo. Account AWS Ti consigliamo di ridurre ulteriormente le autorizzazioni definendo politiche gestite dai AWS clienti

specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta le politiche AWS gestite o le politiche AWS gestite per le funzioni lavorative nella Guida per l'IAMutente.

- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le IAM politiche, concedi solo le autorizzazioni necessarie 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 per applicare le autorizzazioni, consulta Politiche e autorizzazioni nella Guida IAM per l'utente. IAM IAM
- Utilizza le condizioni nelle IAM politiche per limitare ulteriormente l'accesso: puoi aggiungere una condizione alle tue politiche per limitare l'accesso ad azioni e risorse. Ad esempio, puoi scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzandoSSL. È inoltre possibile utilizzare condizioni per concedere l'accesso alle azioni di servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta Elementi IAM JSON della politica: Condizione nella Guida IAM per l'utente.
- Usa IAM Access Analyzer per convalidare IAM le tue policy e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio delle IAM policy () e alle best practice. JSON IAM IAMAccess Analyzer fornisce più di 100 controlli delle politiche e consigli pratici per aiutarti a creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle politiche con IAM Access Analyzer nella Guida per l'utente. IAM
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede l'utilizzo di IAM utenti o di un utente root Account AWS, attiva questa opzione MFA per una maggiore sicurezza. Per richiedere MFA quando vengono richiamate API le operazioni, aggiungi MFA delle condizioni alle tue politiche. Per ulteriori informazioni, consulta Secure API access with MFA nella Guida IAM per l'utente.

Per ulteriori informazioni sulle best practice inIAM, consulta la sezione Procedure consigliate in materia di sicurezza IAM nella Guida IAM per l'utente.

Utilizzo della SNS console Amazon

Per accedere alla console Amazon SNS, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle SNS risorse Amazon presenti 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 autorizzazioni minime per la console agli utenti che effettuano chiamate solo verso il AWS CLI o il. AWS API Consenti invece l'accesso solo alle azioni che corrispondono all'APIoperazione che stanno cercando di eseguire.

Per garantire che gli utenti e i ruoli possano continuare a utilizzare la SNS console Amazon, allega anche la policy Amazon SNS `ConsoleAccess` o la policy `ReadOnly` AWS gestita alle entità. Per ulteriori informazioni, consulta Aggiungere autorizzazioni a un utente nella Guida per l'IAMutente.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi e meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite di autorizzazioni è una funzionalità avanzata in cui si impostano le autorizzazioni massime che una politica basata sull'identità può concedere a un'entità (utente o ruolo). IAM 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 nella Guida per l'utente. IAM IAM
- Politiche di controllo del servizio (SCPs): SCPs sono JSON politiche che specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa (OU) in. AWS Organizations AWS Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se abiliti tutte le funzionalità di un'organizzazione, puoi applicare le politiche di controllo del servizio (SCPs) a uno o tutti i tuoi account. SCPLimita le autorizzazioni per le entità negli account dei membri, incluso ogni utente Account AWS root. Per ulteriori informazioni su Organizations andSCPs, consulta le politiche di controllo dei servizi nella Guida AWS Organizations per l'utente.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta le politiche di sessione nella Guida IAM per l'utente.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per sapere come si AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle politiche nella Guida per l'IAMutente.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra come è possibile creare una politica che consenta IAM agli utenti di visualizzare le politiche in linea e gestite allegate alla propria identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando o a livello di codice. AWS CLI AWS API

```
{
    "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": "*"
```

```
        }
    ]
}
```

Politiche basate sull'identità per Amazon SNS

Supporta le policy basate su identità: sì

Le politiche basate sull'identità sono documenti relativi alle politiche di JSON autorizzazione che è possibile allegare a un'identità, ad esempio un IAM utente, un gruppo di utenti o un ruolo. Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una politica basata sull'identità, consulta Definire le IAM autorizzazioni personalizzate con le politiche gestite dal cliente nella Guida per l'utente. IAM

Con le politiche IAM basate sull'identità, puoi specificare azioni e risorse consentite o negate, nonché le condizioni in base alle quali le azioni sono consentite o negate. 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 ulteriori informazioni su tutti gli elementi che è possibile utilizzare in una JSON politica, vedere il riferimento agli elementi IAM JSON della politica nella Guida per l'IAMutente.

Esempi di policy basate sull'identità per Amazon SNS

Per visualizzare esempi di politiche SNS basate sull'identità di Amazon, consulta. Esempi di policy basate su identità per Amazon Simple Notification Service

Politiche basate sulle risorse all'interno di Amazon SNS

Supporta le policy basate su risorse	Sì

Le politiche basate sulle risorse sono documenti di JSON policy allegati a una risorsa. Esempi di politiche basate sulle risorse sono le policy di trust dei IAM ruoli e le policy dei bucket di Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le 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 abilitare l'accesso tra più account, puoi specificare un intero account o IAM entità in un altro account come principale in una politica 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 il principale e la risorsa sono diversi Account AWS, un IAM amministratore dell'account fidato deve inoltre concedere all'entità principale (utente o ruolo) l'autorizzazione ad 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 la sezione Cross Account Resource Access IAM nella Guida IAM per l'utente.

Utilizzo di politiche basate sull'identità con Amazon SNS

Argomenti

- IAMe SNS le politiche di Amazon insieme
- ARNFormato di SNS risorse Amazon
- SNSAPIAzioni Amazon
- Chiavi SNS della policy di Amazon
- Politiche di esempio per Amazon SNS

Amazon Simple Notification Service si integra con AWS Identity and Access Management (IAM) in modo da poter specificare quali SNS azioni Amazon un utente Account AWS può eseguire con SNS le risorse Amazon. Puoi specificare un particolare argomento nella policy. Ad esempio, puoi utilizzare le variabili quando crei una IAM policy che conceda a determinati utenti della tua organizzazione il permesso di utilizzare l'`Publish`azione su argomenti specifici della tua organizzazione. Account AWS Per ulteriori informazioni, consulta Policy Variables nella IAM guida all'uso.

> **Important**
>
> L'utilizzo di Amazon SNS con IAM non modifica il modo in cui utilizzi AmazonSNS. Non sono state apportate modifiche alle SNS azioni di Amazon e non sono state apportate nuove SNS azioni Amazon relative agli utenti e al controllo degli accessi.

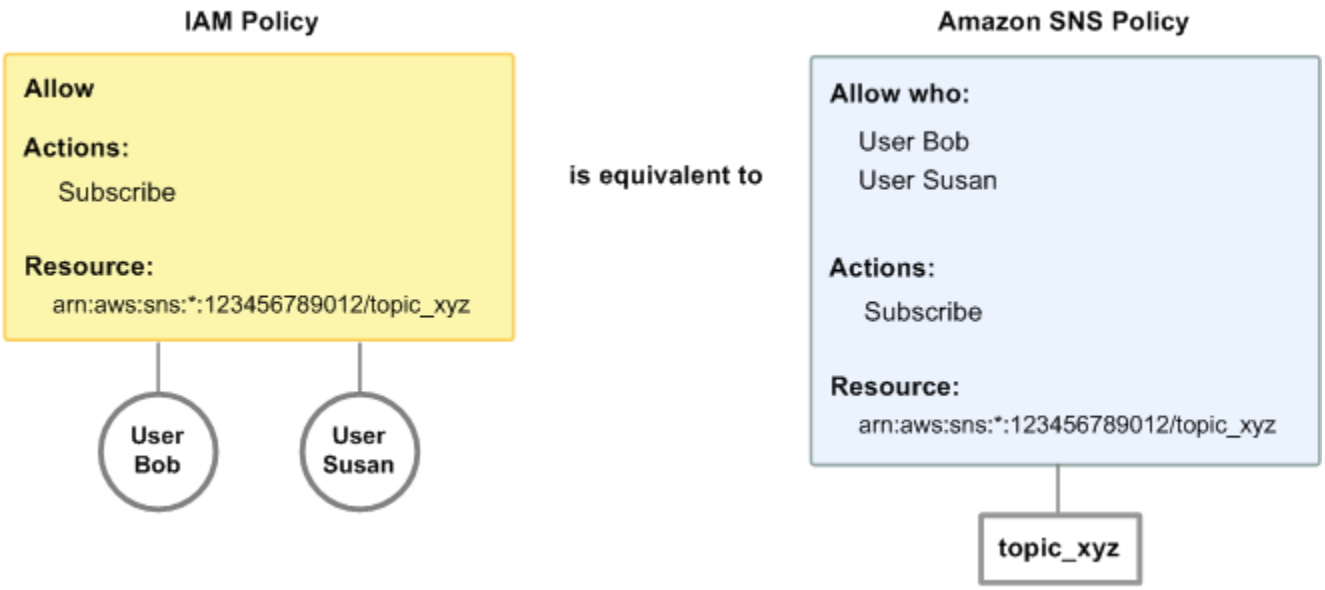
Per esempi di politiche che coprono le SNS azioni e le risorse di Amazon, consultaPolitiche di esempio per Amazon SNS.

IAMe SNS le politiche di Amazon insieme

Utilizzi una IAM policy per limitare l'accesso degli utenti alle SNS azioni e agli argomenti di Amazon. Una IAM policy può limitare l'accesso solo agli utenti del tuo AWS account, non ad altri Account AWS.

Utilizzi una SNS politica di Amazon con un argomento particolare per limitare chi può lavorare su quell'argomento (ad esempio, chi può pubblicare messaggi sull'argomento, chi può iscriversi, ecc.). SNSLe politiche di Amazon possono concedere l'accesso Account AWS ad altri utenti o a utenti interni al tuo Account AWS.

Per concedere ai tuoi utenti le autorizzazioni per i tuoi SNS argomenti Amazon, puoi utilizzare IAM le policy, le SNS politiche di Amazon o entrambe. Nella maggior parte dei casi, puoi ottenere gli stessi risultati con le une o le altre. Ad esempio, il diagramma seguente mostra una IAM politica e una SNS politica Amazon equivalenti. La IAM policy consente l'SNS`Subscribe`azione di Amazon per l'argomento chiamato topic_xyz nel tuo Account AWS La IAM policy è allegata agli utenti Bob e Susan (il che significa che Bob e Susan dispongono delle autorizzazioni indicate nella policy). Allo stesso modo, la SNS politica di Amazon concede a Bob e Susan il permesso di accedere a topic_xyz`Subscribe`.



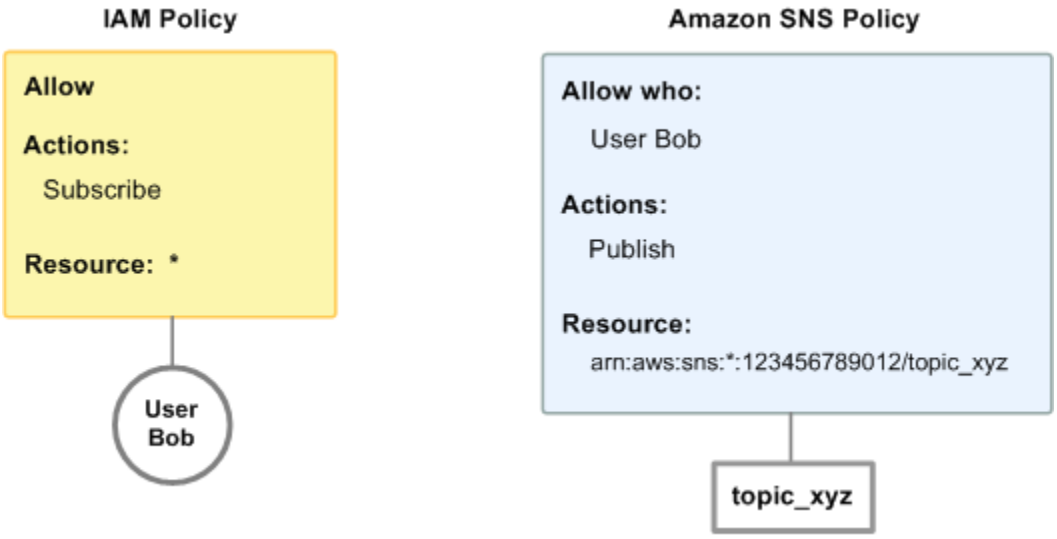
> **Note**
>
> L'esempio precedente mostra policy semplici senza condizioni. Puoi specificare una particolare condizione in una qualsiasi delle policy e ottenere lo stesso risultato.

C'è una differenza tra AWS IAM le SNS politiche di Amazon: il sistema di SNS policy di Amazon ti consente di concedere l'autorizzazione ad altri Account AWS, mentre la IAM politica no.

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, a Bob si applicano sia una IAM SNS policy che una policy Amazon. La IAM politica gli concede l'autorizzazione per `Subscribe` qualsiasi argomento, mentre la Account AWS SNS politica di Amazon gli concede il permesso di utilizzare `Publish` su un argomento specifico (topic_xyz). Il diagramma seguente illustra questo concetto.

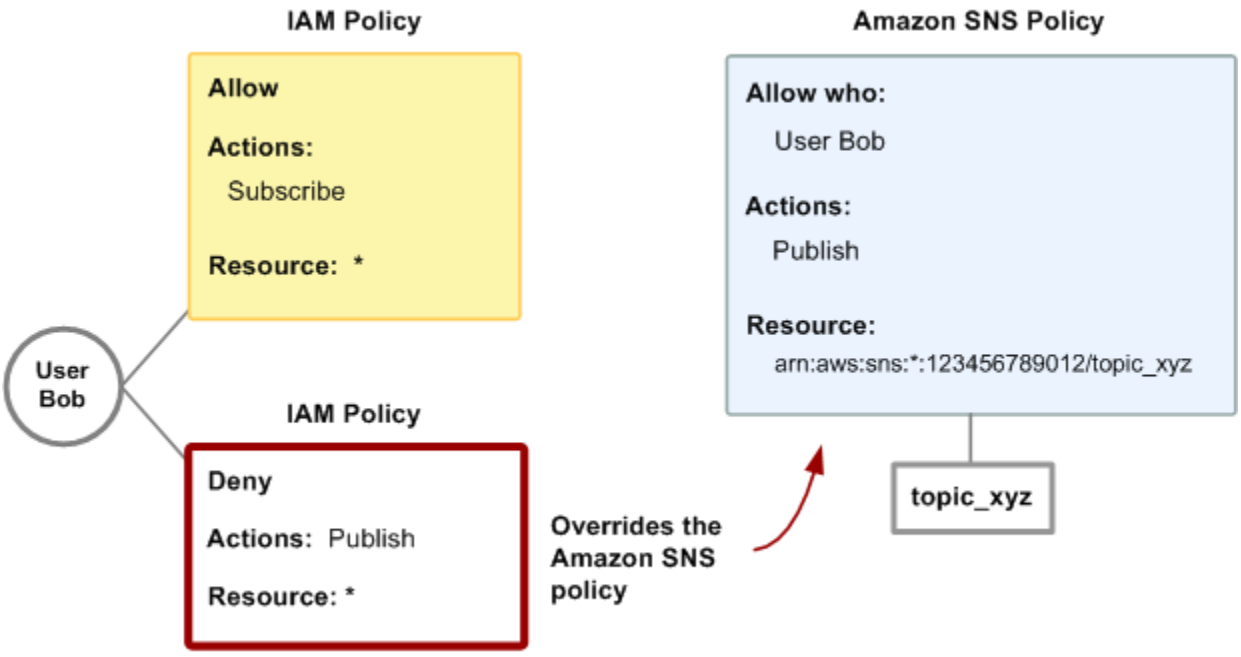


Se Bob dovesse inviare una richiesta di sottoscrizione a qualsiasi argomento dell' AWS account, la IAM politica consentirebbe l'azione. Se Bob dovesse inviare una richiesta per pubblicare un messaggio su topic_xyz, la SNS policy di Amazon consentirebbe l'azione.

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 IAM politica che gli neghi l'accesso all'azione su tutti gli argomenti. `Publish` Questa terza politica ha la precedenza sulla politica di Amazon che originariamente gli aveva dato il permesso di pubblicare su topic_xyz, perché un rifiuto esplicito ha

sempre la precedenza sull'autorizzazione (per ulteriori informazioni sulla logica di valutazione delle SNS politiche, consulta). Logica della valutazione Il diagramma seguente illustra questo concetto.



Per esempi di politiche che coprono le SNS azioni e le risorse di Amazon, consultaPolitiche di esempio per Amazon SNS. Per ulteriori informazioni sulla stesura SNS delle politiche di Amazon, consulta la documentazione tecnica di Amazon SNS.

ARNFormato di SNS risorse Amazon

Per AmazonSNS, gli argomenti sono l'unico tipo di risorsa che puoi specificare in una politica. Di seguito è riportato il formato Amazon Resource Name (ARN) per gli argomenti.

```
arn:aws:sns:region:account_ID:topic_name
```

Per ulteriori informazioni ARNsin meritoARNs, consulta la Guida per IAM l'utente.

Example

Quanto segue è un ARN argomento denominato MyTopic nella regione us-east-2, appartenente a 123456789012. Account AWS

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

Example

Se hai un argomento denominato MyTopic in ciascuna delle diverse regioni SNS supportate da Amazon, puoi specificare gli argomenti con quanto segueARN.

```
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 tua comodità, quando crei un argomento, Amazon SNS restituisce l'argomento ARN nella risposta.

SNSAPIAzioni Amazon

In una IAM politica, puoi specificare tutte le azioni SNS offerte da Amazon. Tuttavia, le `Unsubscribe` azioni `ConfirmSubscription` and non richiedono l'autenticazione, il che significa che, anche se specifichi tali azioni in una policy, IAM non limiteranno l'accesso degli utenti a tali azioni.

Ogni operazione che specifichi in una policy deve essere preceduta dalla stringa in minuscolo `sns:`. Per specificare tutte le SNS azioni di Amazon, ad esempio, dovresti usare`sns:*`. Per un elenco delle azioni, consulta Amazon Simple Notification Service API Reference.

Chiavi SNS della policy di Amazon

Amazon SNS implementa le seguenti chiavi di policy AWS estese, oltre ad alcune chiavi specifiche del servizio.

Per un elenco delle chiavi di condizione supportate da ciascuna di esse Servizio AWS, consulta Azioni, risorse e chiavi di condizione Servizi AWS nella Guida per l'IAMutente. Per un elenco delle chiavi di condizione che possono essere utilizzate in più modi Servizi AWS, consulta le chiavi di contesto delle condizioni AWS globali nella Guida per l'IAMutente.

Amazon SNS utilizza le seguenti chiavi specifiche del servizio. Utilizza queste chiavi nelle policy che limitano l'accesso alle richieste `Subscribe`.

- sns:endpoint: l'indirizzo e-mail URL fornito ARN da una `Subscribe` richiesta o da un abbonamento precedentemente confermato. Utilizza questa chiave con le condizioni di stringa

(vedi Politiche di esempio 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 Politiche di esempio per Amazon SNS) per limitare la pubblicazione a specifici protocolli di consegna (ad esempio, https).

Politiche di esempio per Amazon SNS

Questa sezione mostra diverse semplici politiche per il controllo dell'accesso degli utenti ad AmazonSNS.

> **Note**
>
> In futuro, Amazon SNS potrebbe aggiungere nuove azioni che dovrebbero essere incluse logicamente in una delle seguenti politiche, in base agli obiettivi dichiarati della politica.

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 3: Offri agli utenti la Account AWS possibilità di iscriversi agli 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 SNS policy di Amazon o una IAM policy per consentire a un partner di pubblicare su un argomento specifico. Se il tuo partner dispone di una polizza Amazon Account AWS, potrebbe essere più semplice utilizzare una SNS politica Amazon. Tuttavia, chiunque nell'azienda del partner possieda le credenziali AWS di sicurezza può pubblicare messaggi sull'argomento. Questo esempio presuppone che tu voglia limitare l'accesso a una determinata persona (o applicazione). Per fare ciò devi trattare il partner come un utente all'interno della tua azienda e utilizzare una IAM politica anziché una SNS politica Amazon.

Per questo esempio, creiamo un gruppo chiamato WidgetCo che rappresenta l'azienda partner; creiamo un utente per la persona (o l'applicazione) specifica presso l'azienda partner che deve accedere; quindi inseriamo l'utente nel gruppo.

Quindi alleghiamo una politica che concede al gruppo l'`Publish`accesso all'argomento specifico denominato WidgetPartnerTopic.

Vogliamo anche impedire al WidgetCo gruppo di fare altro con gli argomenti, quindi aggiungiamo una dichiarazione che nega l'autorizzazione a qualsiasi SNS azione di Amazon diversa da quella `Publish` su argomenti diversi WidgetPartnerTopic da. Ciò è necessario solo se esiste un'ampia politica altrove nel sistema che garantisce agli utenti un ampio accesso ad AmazonSNS.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Utilizzo di credenziali di sicurezza temporanee con Amazon SNS

AWS Identity and Access Management (IAM) consente di concedere credenziali di sicurezza temporanee agli utenti e alle applicazioni che devono accedere alle AWS risorse. Queste credenziali di sicurezza temporanee vengono utilizzate principalmente per IAM i ruoli e l'accesso federato tramite protocolli standard di settore come OpenID SAML Connect (). OIDC

Per gestire efficacemente l'accesso alle AWS risorse, è essenziale comprendere i seguenti concetti chiave:

- IAMRuoli: i ruoli vengono utilizzati per delegare l'accesso alle AWS risorse. I ruoli possono essere assunti da entità come EC2 istanze Amazon, funzioni Lambda o utenti di altri. Account AWS
- Utenti federati: si tratta di utenti autenticati tramite provider di identità esterni () IdPs utilizzando o. SAML OIDC L'accesso federato è consigliato per gli utenti umani, mentre IAM i ruoli devono essere utilizzati per le applicazioni software.
- Roles Anywhere: per le applicazioni esterne che richiedono AWS l'accesso, puoi utilizzare IAM Roles Anywhere per gestire in modo sicuro l'accesso senza creare credenziali a lungo termine.

Puoi utilizzare credenziali di sicurezza temporanee per effettuare richieste ad AmazonSNS. Le API librerie SDKs and calcolano la firma necessaria utilizzando queste credenziali per autenticare le tue richieste. Le richieste con credenziali scadute verranno rifiutate da Amazon. SNS

Per ulteriori informazioni sulle credenziali di sicurezza temporanee, consulta la sezione Utilizzo IAM dei ruoli e Fornitura dell'accesso agli utenti autenticati esternamente (federazione delle identità) nella Guida per l'utente. IAM

Example HTTPSesempio di richiesta

L'esempio seguente dimostra come autenticare una SNS richiesta Amazon utilizzando credenziali di sicurezza temporanee ottenute da AWS Security Token Service (). STS

```
https://sns.us-east-2.amazonaws.com/
?Action=CreateTopic
&Name=My-Topic
&SignatureVersion=4
&SignatureMethod=AWS4-HMAC-SHA256
&Timestamp=2023-07-05T12:00:00Z
&X-Amz-Security-Token=SecurityTokenValue
&X-Amz-Date=20230705T120000Z
&X-Amz-Credential=<your-access-key-id>/20230705/us-east-2/sns/aws4_request
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Passaggi per autenticare la richiesta

1. Ottenere credenziali di sicurezza temporanee: consente AWS STS di assumere un ruolo o ottenere credenziali utente federate. Ciò ti fornirà un ID della chiave di accesso, una chiave di accesso segreta e un token di sicurezza.
2. Crea la richiesta: includi i parametri richiesti per la tua SNS azione su Amazon (ad esempio CreateTopic) e assicurati di utilizzarli HTTPS per comunicazioni sicure.
3. Firma la richiesta: utilizza la procedura AWS Signature Version 4 per firmare la richiesta. Ciò comporta la creazione di una richiesta canonica e string-to-sign il calcolo della firma. Per ulteriori informazioni sulla versione 4 di AWS Signature, consulta Usare la firma Signature Version 4 nella Amazon EBS User Guide.
4. Invia la richiesta: includi X-Amz-Security-Token nell'intestazione della richiesta per passare le credenziali di sicurezza temporanee ad Amazon. SNS

SNSAPIAutorizzazioni Amazon: riferimento ad azioni e risorse

L'elenco seguente fornisce informazioni specifiche sull'SNSimplementazione Amazon del controllo degli accessi:

- 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).
Risorsa	1 (0) non è valido. Il valore deve corrispondere all'argomento ARN della politica.)

Azioni SNS politiche Amazon valide

Amazon SNS supporta le azioni mostrate 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

Azione	Descrizione
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.
sns:Subscribe	Autorizza la sottoscrizione a un argomento.

Chiavi specifiche del servizio

Amazon SNS utilizza le seguenti chiavi specifiche del servizio. Puoi utilizzare queste chiavi nelle policy che limitano l'accesso alle richieste `Subscribe`.

- sns:endpoint: l'indirizzo e-mail URL fornito ARN da una `Subscribe` richiesta o da un abbonamento precedentemente confermato. Utilizza con le condizioni di stringa (vedi Politiche di esempio 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 questa chiave con le condizioni di stringa (vedi Politiche di esempio per Amazon SNS) per limitare la pubblicazione a specifici protocolli di consegna (ad esempio, https).

> **Important**
>
> Quando utilizzi una policy per controllare l'accesso tramite SNS:Endpoint, tieni presente che i DNS problemi potrebbero influire sulla risoluzione dei nomi dell'endpoint in futuro.

Risoluzione dei problemi di identità e accesso ad Amazon Simple Notification Service

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare lavorando con Amazon SNS eIAM.

Argomenti

Non sono autorizzato a eseguire un'azione 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`*.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo ad AmazonSNS.

Alcuni Servizi AWS consentono di trasferire un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente errore di esempio si verifica quando un IAM utente denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in AmazonSNS. 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`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne Account AWS a me di accedere alle mie SNS risorse Amazon

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali politiche per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon SNS supporta queste funzionalità, consultaCome SNS funziona Amazon con IAM.
- Per sapere come fornire l'accesso alle tue risorse su Account AWS un sito di tua proprietà, consulta Fornire l'accesso a un IAM utente di un altro Account AWS utente di tua proprietà nella Guida per l'IAMutente.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta Fornire l'accesso a persone Account AWS di proprietà di terzi nella Guida per l'IAMutente.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso agli utenti autenticati esternamente (federazione delle identità) nella Guida per l'IAMutente.

- Per conoscere la differenza tra l'utilizzo di ruoli e politiche basate sulle risorse per l'accesso tra account diversi, consulta la sezione Accesso alle risorse tra account nella Guida per l'utente. IAM IAM

Registrazione e monitoraggio in Amazon SNS

Amazon ti SNS consente di tracciare e monitorare l'attività di messaggistica registrando API le chiamate CloudTrail e monitorando gli argomenti con CloudWatch. Questi strumenti ti aiutano a ottenere informazioni dettagliate sulla consegna dei messaggi, a risolvere i problemi e a garantire lo stato dei tuoi flussi di lavoro di messaggistica. Questo argomento comprende quanto segue:

- Registrazione delle SNS API chiamate Amazon tramite CloudTrail. Questa registrazione ti consente di tenere traccia delle azioni eseguite sui tuoi SNS argomenti Amazon, come la creazione di argomenti, la gestione degli abbonamenti e la pubblicazione di messaggi. Analizzando CloudTrail i log, puoi identificare chi ha fatto API richieste specifiche e quando sono state fatte, aiutandoti a controllare e risolvere i problemi del tuo utilizzo di Amazon. SNS
- Monitoraggio SNS degli argomenti di Amazon tramite CloudWatch. CloudWatch fornisce metriche che ti consentono di osservare le prestazioni e lo stato dei tuoi SNS argomenti Amazon in tempo reale. Imposta allarmi in base a queste metriche, per consentirti di rispondere prontamente a eventuali anomalie, come errori di consegna o elevata latenza dei messaggi. Questa funzionalità di monitoraggio garantisce la possibilità di mantenere l'affidabilità del sistema di messaggistica SNS basato sulla soluzione proattiva dei potenziali problemi.

Registrazione delle SNS API chiamate Amazon tramite CloudTrail

Amazon SNS è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, un ruolo o un AWS servizio in AmazonSNS. CloudTrail acquisisce le API chiamate per Amazon SNS come eventi. Le chiamate acquisite includono chiamate dalla SNS console Amazon e chiamate in codice verso le SNS API operazioni Amazon. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Amazon. SNS Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare la richiesta che è stata effettuata ad AmazonSNS, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per saperne di più CloudTrail, incluso come configurarlo e abilitarlo, consulta la Guida per AWS CloudTrail l'utente.

SNSInformazioni su Amazon in CloudTrail

CloudTrail è abilitata sul tuo Account AWS quando crei l'account. Quando si verifica un'attività di evento supportata in AmazonSNS, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nel tuo Account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi del tuo sito Account AWS, compresi gli eventi per AmazonSNS, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutte le AWS regioni. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un trail
- CloudTrail Servizi e integrazioni supportati
- Configurazione di Amazon SNS Notifications per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Eventi del piano di controllo in CloudTrail

Amazon SNS supporta la registrazione delle seguenti azioni come eventi nei file di CloudTrail registro:

- `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**
>
> Se non hai effettuato l'accesso ad Amazon Web Services (modalità non autenticata) e vengono richiamate le azioni ConfirmSubscriptiono Unsubscribe, tali azioni non verranno registrate. CloudTrail 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'`ConfirmSubscription`azione non verrebbe registrata. 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 AWS servizio.

Per ulteriori informazioni, consulta l'CloudTrail userIdentity elemento.

Eventi del piano dati in CloudTrail

Per abilitare la registrazione delle seguenti API azioni nei CloudTrail file, dovrai abilitare la registrazione delle attività del piano API dati. 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 per tipo di risorsa, per un controllo granulare su quali SNS API chiamate Amazon desideri registrare e pagare in modo selettivo. CloudTrail Ad esempio, specificando `AWS::SNS::Topic` come tipo di risorsa, puoi registrare le chiamate `Publish` e `PublishBatch` API le azioni relative agli argomenti. Allo stesso modo, specificando `AWS::SNS::PlatformEndpoint` come tipo di risorsa, è possibile registrare le chiamate all'APIazione di pubblicazione per gli endpoint della piattaforma. Per ulteriori informazioni, consulta la sezione `AdvancedEventSelector`Reference. AWS CloudTrail API

> **Note**
>
> `AWS::SNS::PhoneNumber`Il tipo di SNS risorsa Amazon non è registrato da CloudTrail.

Piano SNS dati Amazon APIs

- `Publish`
- `PublishBatch`

Esempio: voci dei file di SNS log di Amazon

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta da un'fonte e include informazioni sull'azione richiesta, data e ora dell'azione, parametri richiesti e così via. CloudTrail i file di registro non sono una traccia ordinata delle API chiamate pubbliche, quindi non appaiono in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che mostra le azioni `ListTopicsCreateTopic`, `eDeleteTopic`.

```
{
  "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 le voci di CloudTrail registro che illustrano `PublishBatch` le azioni `Publish` and.

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 SNS degli argomenti di Amazon tramite CloudWatch

Amazon SNS e Amazon CloudWatch sono integrati in modo da poter raccogliere, visualizzare e analizzare i parametri per ogni SNS notifica Amazon attiva. Dopo aver configurato CloudWatch AmazonSNS, puoi ottenere informazioni più dettagliate sulle prestazioni dei tuoi SNS argomenti, delle notifiche push e delle SMS consegne di Amazon. Ad esempio, puoi impostare un allarme per inviarti una notifica e-mail se viene raggiunta una soglia specificata per un SNS parametro Amazon, ad `NumberOfNotificationsFailed` esempio. Per un elenco di tutte le metriche SNS inviate da Amazon CloudWatch, consultaSNSMetriche Amazon. Per ulteriori informazioni sulle notifiche SNS push di Amazon, consultaInvio di notifiche push per dispositivi mobili con Amazon SNS.

> **Note**
>
> Le metriche con cui configuri CloudWatch i tuoi SNS argomenti Amazon vengono raccolte automaticamente e inserite a CloudWatch intervalli di 1 minuto. Queste metriche vengono raccolte su tutti gli argomenti che soddisfano le linee guida per l' CloudWatch attività.
> Un argomento è considerato attivo fino a sei ore dall'ultima attività (ovvero qualsiasi API chiamata) sull'argomento. CloudWatch
> Non sono previsti costi per i SNS parametri Amazon riportati in CloudWatch; sono forniti come parte del SNS servizio Amazon.

Visualizza i CloudWatch parametri per Amazon SNS

Puoi monitorare i parametri per Amazon SNS utilizzando la CloudWatch console, l'interfaccia a riga CloudWatch di comando (CLI) di Amazon o utilizzando programmaticamente il. CloudWatch API Le procedure seguenti mostrano come accedere ai parametri mediante la AWS Management Console.

Per visualizzare le metriche utilizzando la console CloudWatch

1. Accedi alla CloudWatchconsole.
2. Nel pannello di navigazione, scegli Metrics (Parametri).
3. Nella scheda Tutte le metriche, scegli SNS, quindi scegli una delle seguenti dimensioni:
   - 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 scegli Topic Metrics e poi scegli NumberOfMessagesPublished, viene visualizzato il numero medio di SNS messaggi Amazon pubblicati per un periodo di 1 minuto nell'intervallo di tempo di 6 ore.
5. Per visualizzare i parametri di SNS utilizzo di Amazon, nella scheda Tutti i parametri, scegli Utilizzo e seleziona il metrico di SNS utilizzo Amazon di destinazione (ad esempio,). `NumberOfMessagesPublishedPerAccount`

Imposta CloudWatch allarmi per i parametri di Amazon SNS

CloudWatch consente inoltre di impostare allarmi quando viene raggiunta una soglia per una metrica. Ad esempio, è possibile impostare un allarme per la metrica NumberOfNotificationsFailed, in modo che quando il numero di soglia specificato viene raggiunto entro il periodo di campionamento, venga inviata una notifica via e-mail per informarti dell'evento.

Per impostare allarmi utilizzando la console CloudWatch

1. Accedi a AWS Management Console e apri la CloudWatch console 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 le SNS metriche di Amazon per individuare la metrica su cui desideri inserire 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 desideri CloudWatch inviarti un'e-mail quando viene raggiunto lo stato di allarme, scegli un SNS argomento Amazon esistente o scegli 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 Crea nuovo argomento e-mail per creare un nuovo SNS argomento Amazon, 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 CloudWatch e sugli allarmi, consulta la CloudWatchdocumentazione.

SNSMetriche Amazon

Amazon SNS invia le seguenti metriche a. CloudWatch

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfMessagesPublished`	Il numero di messaggi pubblicati sui tuoi SNS argomenti Amazon. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: Sum
AWS/SNS	`NumberOfNotificationsDelivered`	Il numero di messaggi inviati con successo dai tuoi SNS argomenti Amazon agli endpoint abbonati. 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: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName

Spazio dei nomi	Parametro	Descrizione
		Statistiche valide: Sum
AWS/SNS	`NumberOfNotificationsFailed`	Il numero di messaggi che Amazon SNS non è riuscito a recapitare. Per AmazonSQS, e-mail o endpoint push mobiliSMS, la metrica aumenta di 1 quando Amazon SNS interrompe i tentativi di recapito dei messaggi. Per i HTTP nostri HTTPS endpoint, la metrica include ogni tentativo di recapito fallito, compresi i nuovi tentativi successivi al 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. È possibile controllare il numero di tentativi per gli endpoint. HTTP Per ulteriori informazioni, consulta Tentativi di recapito dei SNS messaggi Amazon. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

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: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Il numero di messaggi rifiutati dalle policy di filtro delle sottoscrizioni per il filtro basato su attributi. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Il numero di messaggi rifiutati dalle policy di filtro delle sottoscrizioni per il filtro basato sul payload. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Il numero di messaggi che sono stati rifiutati dai criteri di filtro degli abbonamenti perché gli attributi dei messaggi non sono validi, ad esempio perché l'attributo non JSON è formattato correttamente. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media
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: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Il numero di messaggi che sono stati rifiutati dai criteri di filtro degli abbonamenti perché il corpo del messaggio non è valido per il filtraggio, ad esempio corpo del messaggio non validoJSON. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Numero di messaggi che sono stati spostati in una coda dead-letter. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Numero di messaggi che non possono essere spostati in una coda dead-letter. Unità: conteggio Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`PublishSize`	Dimensione dei messaggi pubblicati. Unità: byte Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: minimo, massimo, medio e conteggio

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`SMSMonthToDateSpentUSD`	Gli addebiti maturati dall'inizio del mese solare corrente per l'invio SMS di messaggi. Puoi impostare un allarme per questa metrica per sapere quando gli month-to-date addebiti si avvicinano alla quota di SMS spesa mensile del tuo account. Quando Amazon SNS determina che l'invio di un SMS messaggio comporterebbe un costo superiore a tale quota, interrompe la pubblicazione dei SMS messaggi nel giro di pochi minuti. Per informazioni sull'impostazione della quota di SMS spesa mensile o per informazioni su come richiedere un aumento della quota di spesa con, consulta. AWSImpostazione delle preferenze di SMS messaggistica in Amazon SNS Unità: USD Dimensioni valide: nessuna Statistiche valide: somma

Spazio dei nomi	Parametro	Descrizione
AWS/SNS	`SMSSuccessRate`	La percentuale di recapiti riusciti dei SMS messaggi. Unità: conteggio Dimensioni valide: PhoneNumber Statistiche valide: somma, media, campioni di dati

Dimensioni per i SNS parametri di Amazon

Amazon Simple Notification Service invia le seguenti dimensioni a CloudWatch.

Dimensione	Descrizione
`Application`	Filtri sugli oggetti dell'applicazione, che rappresentano un'app e un dispositivo registrati con uno dei servizi di notifica push supportati, come APNs eFCM.
`Application,Platform`	Filtri sugli oggetti dell'applicazione e della piattaforma, dove gli oggetti della piattaforma si trovano per i servizi di notifica push supportati, ad esempio APNs eFCM.
`Country`	Filtri in base al paese o alla regione di destinazione di un SMS messaggio. Il paese o la regione sono rappresentati dal codice ISO alfa-2 3166-1.
`PhoneNumber`	Filtra in base al numero di telefono quando pubblichi SMS direttamente su un numero di telefono (senza argomento).
`Platform`	Filtri sugli oggetti della piattaforma per i servizi di notifica push, come APNs eFCM.
`TopicName`	Filtri sui nomi degli SNS argomenti di Amazon.

Dimensione	Descrizione
SMSType	Filtri in base al tipo di SMS messaggio. Può essere promozionale o transazionale.

Metriche SNS di utilizzo di Amazon

Amazon Simple Notification Service invia i seguenti parametri di utilizzo a CloudWatch.

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
AWS/Usage	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	Risorsa	• Il numero di messaggi pubblicati sui tuoi SNS argomenti Amazon nel tuo AWS account. • Unità: nessuna • Statistiche valide: Sum
AWS/Usage	SNS	ResourceCount	ApproximateNumberOfTopics	Risorsa	• Il numero approssimativo di argomenti nel tuo AWS account.

Spazio dei nomi	Servizio	Parametro	Risorsa	Tipo	Descrizione
					• Unità: nessuna • Statistiche valide: Average (Media), Minimum (Minimo), Maximum (Massimo), Sum (Somma)
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 API chiamate per l'Amazon seleziona to SNS API sul tuo AWS account. • Unità: nessuna • Statistic he 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 nell'SNSambito di diversi programmi di AWS conformità, tra cui l'Health Insurance Portability and Accountability Act (). HIPAA

Per un elenco dei AWS servizi che rientrano nell'ambito di specifici programmi di conformità, consulta AWS Services in Scope by Compliance Program by Compliance Program. Per informazioni generali, vedere Programmi di AWS conformità Programmi di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Scaricamento dei report in AWS Artifact .

La tua responsabilità di conformità quando usi Amazon SNS è determinata dalla sensibilità dei tuoi dati, dagli obiettivi di conformità della tua azienda e dalle leggi e dai regolamenti applicabili. AWS fornisce le seguenti risorse per contribuire alla conformità:

- Guide rapide su sicurezza e conformità Guide introduttive implementazione illustrano considerazioni sull'architettura e forniscono passaggi per implementare ambienti di base incentrati sulla sicurezza e la conformità. AWS
- Whitepaper sull'architettura per HIPAA la sicurezza e la conformità: questo white paper descrive come le aziende possono utilizzare per creare applicazioni conformi. AWS HIPAA
- AWS Risorse per la conformità Risorse per AWS : questa raccolta di cartelle di lavoro e guide potrebbe riguardare il settore e la località in cui operi.
- Valutazione delle risorse con le regole nella Guida per gli AWS Config sviluppatori: il AWS Config servizio valuta la conformità delle configurazioni delle risorse alle pratiche interne, alle linee guida del settore e alle normative.
- AWS Security Hub— Questo AWS servizio offre una visione completa dello stato di sicurezza dell'utente e consente di verificare la conformità agli standard e alle best practice del settore della sicurezza. AWS

Resilienza in Amazon SNS

La resilienza in Amazon SNS è garantita sfruttando l'infrastruttura AWS globale, che ruota attorno alle Regioni AWS 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, SNS gli abbonati Amazon 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 SNS argomenti di Amazon possono essere configurati con nuovi tentativi di consegna e code di lettera morta, abilitando la gestione automatica degli errori temporanei e garantendo che i messaggi raggiungano in modo affidabile le destinazioni previste.

Amazon supporta SNS anche il filtraggio dei messaggi 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.

Usa AWS API le azioni per accedere ad Amazon SNS tramite la rete. I client devono supportare Transport Layer Security (TLS) 1.2 o versione successiva. I client devono inoltre supportare suite di crittografia con Perfect Forward Secrecy (PFS), come Ephemeral Diffie-Hellman () o Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE

È necessario firmare le richieste utilizzando sia un ID chiave di accesso che una chiave di accesso segreta associata a un 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 API azioni da qualsiasi posizione di rete, ma Amazon SNS supporta politiche di accesso basate sulle risorse, che possono includere restrizioni basate sull'indirizzo IP di origine. Puoi anche utilizzare SNS le policy di Amazon per controllare l'accesso da VPC endpoint Amazon specifici o specificiVPCs. Ciò isola efficacemente l'accesso alla rete a un determinato SNS argomento di Amazon solo a quello specifico all'VPCinterno della AWS rete. Per ulteriori informazioni, consulta Limita la pubblicazione a un SNS argomento Amazon solo da un VPC endpoint specifico.

Risoluzione dei problemi relativi ad SNS Amazon

Scopri come utilizzare per AWS X-Ray risolvere SNS gli argomenti di Amazon tracciando e analizzando i messaggi, identificando i problemi e ottimizzando le prestazioni attraverso dati dettagliati di richiesta e risposta.

Risoluzione dei problemi SNS relativi ad Amazon utilizzando AWS X-Ray

AWS X-Ray raccoglie dati sulle richieste servite dalla tua applicazione e ti consente di visualizzare e filtrare i dati per identificare potenziali problemi e opportunità di ottimizzazione. Per ogni richiesta tracciata all'applicazione, è possibile visualizzare informazioni dettagliate sulla richiesta, sulla risposta e sulle chiamate effettuate dall'applicazione verso AWS risorse a valle, microservizi, database e Web. HTTP APIs

Puoi usare X-Ray con Amazon SNS per tracciare e analizzare i messaggi che viaggiano attraverso la tua applicazione. Puoi utilizzare il AWS Management Console per visualizzare la mappa delle connessioni tra Amazon SNS e altri servizi utilizzati dalla tua applicazione. È inoltre possibile utilizzare la console per visualizzare i parametri come la latenza media e le percentuali di errore. Per ulteriori informazioni, consulta Amazon SNS e AWS X-Ray la AWS X-Ray Developer Guide.

Tracciamento attivo in Amazon SNS

Puoi utilizzarlo AWS X-Ray per tracciare e analizzare le richieste degli utenti mentre passano dai tuoi SNS argomenti Amazon agli abbonamenti Amazon Data FirehoseSQS, AWS LambdaAmazon e HTTP/S endpoint. Poiché X-Ray ti offre una end-to-end visualizzazione dell'intera richiesta, puoi visualizzare ciò che chiama il tuo SNS argomento Amazon 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**
>
> SNSGli argomenti Amazon con numerosi abbonamenti possono raggiungere un limite di dimensione e non essere tracciati completamente. Per informazioni sui limiti di dimensione

dei documenti di traccia, consulta la sezione X-ray service quotas nella sezione General Reference. AWS

Se chiami un Amazon SNS API da un servizio che è già stato tracciato, Amazon SNS trasmette la traccia, anche se il tracciamento a raggi X non è abilitato su. API

Amazon SNS supporta il tracciamento a raggi X sia per gli standard che per gli argomenti. FIFO Puoi abilitare X-Ray per un SNS argomento Amazon utilizzando la SNSconsole Amazon, Amazon, AmazonSimple Notification Service CLI Reference o. SNS `SetTopicAttributes` API AWS CloudFormation

Per ulteriori informazioni sull'uso di Amazon SNS con X-Ray, consulta Amazon SNS e AWS X-Ray la AWS X-Ray Developer Guide.

Argomenti

- Autorizzazioni per il tracciamento attivo
- Abilitazione del tracciamento attivo su un SNS argomento Amazon utilizzando la console AWS
- Abilitazione del tracciamento attivo su un SNS argomento Amazon utilizzando il AWS SDK
- Abilitazione del tracciamento attivo su un SNS argomento Amazon utilizzando il AWS CLI
- Abilitazione del tracciamento attivo su un SNS argomento Amazon utilizzando AWS CloudFormation
- Verifica dell'abilitazione del tracciamento attivo per l'argomento
- Test del tracciamento attivo

Autorizzazioni per il tracciamento attivo

Quando utilizza la SNS console Amazon, Amazon SNS tenta di creare le autorizzazioni necessarie per l'SNSargomento Amazon per chiamare X-Ray. Il tentativo può essere rifiutato se non disponi di autorizzazioni sufficienti per utilizzare la SNS console Amazon. Per ulteriori informazioni, consulta Gestione delle identità e degli accessi in Amazon SNS e Casi di esempio per il controllo degli SNS accessi di Amazon.

Quando si utilizzaCLI, 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 SNS argomento Amazon utilizzando la console AWS

Quando il tracciamento attivo è abilitato su un SNS argomento Amazon, legge l'ID di traccia, invia i dati al cliente in base all'ID di traccia e propaga l'ID di traccia ai servizi downstream.

1. Accedi alla SNSconsole Amazon.
2. Scegli un argomento o creane uno nuovo. Per maggiori dettagli sulla creazione di argomenti, consulta Creazione di un SNS argomento Amazon.
3. Nella pagina Crea argomento, nella sezione Dettagli, scegli un tipo di argomento: FIFOo 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 SNS argomento Amazon, puoi utilizzare la mappa dei servizi X-Ray per visualizzare end-to-end le tracce e le mappe dei servizi per l'argomento.

Abilitazione del tracciamento attivo su un SNS argomento Amazon utilizzando il AWS SDK

Il seguente esempio di codice mostra come abilitare l'active tracing su un SNS argomento Amazon utilizzando AWS SDK for 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 SNS argomento Amazon utilizzando il AWS CLI

Il seguente esempio di codice mostra come abilitare il tracciamento attivo su un SNS argomento Amazon 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 SNS argomento Amazon utilizzando AWS CloudFormation

Lo AWS CloudFormation stack seguente mostra come abilitare il tracciamento attivo su un argomento AmazonSNS.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

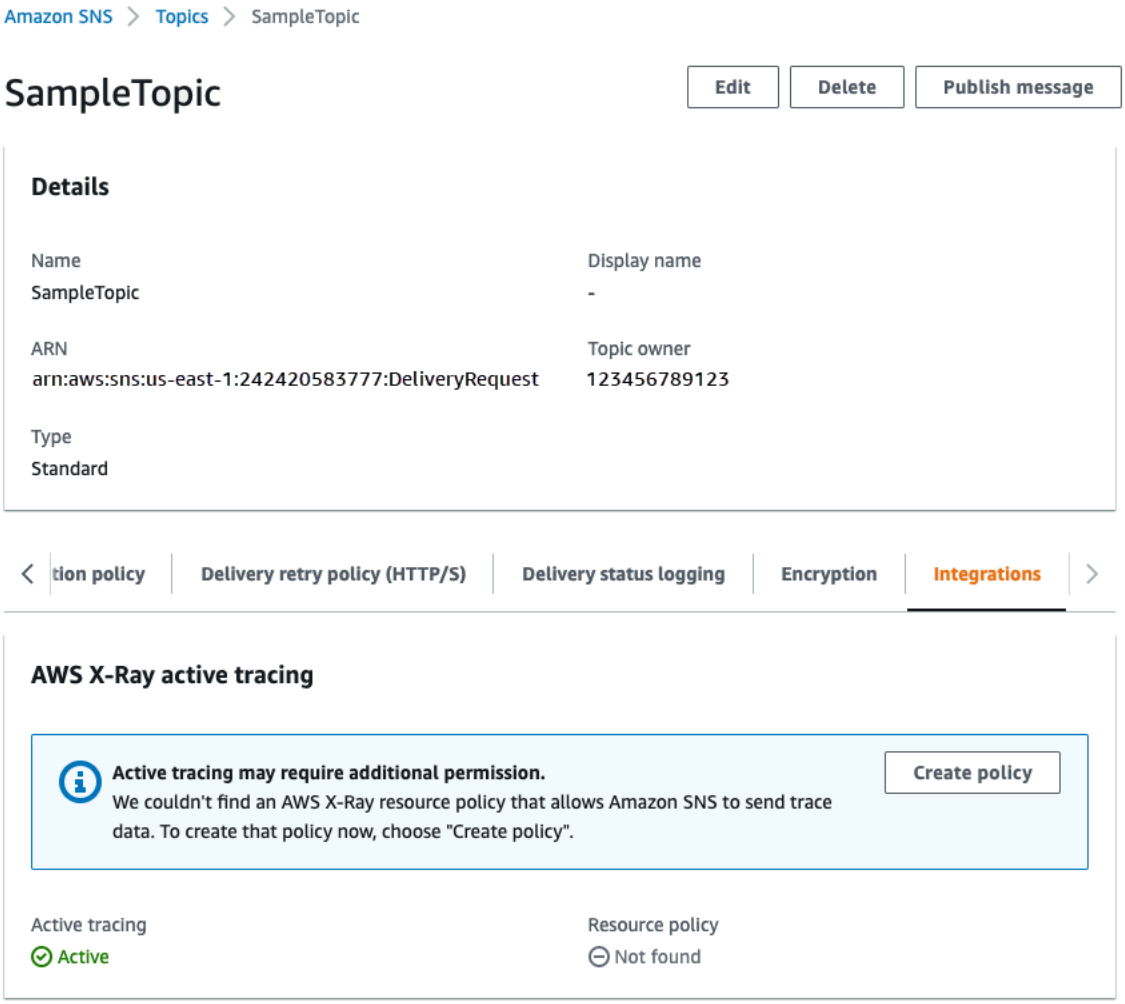
Verifica dell'abilitazione del tracciamento attivo per l'argomento

Puoi utilizzare la SNS console Amazon per verificare se il tracciamento attivo è abilitato per il tuo argomento o se la politica delle risorse non è stata aggiunta.

1. Accedi alla SNSconsole Amazon.
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 SNSconsole Amazon.
2. Crea un SNS argomento Amazon. 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 sull'SNSargomento Amazon. Per informazioni dettagliate su come eseguire questa operazione, consulta Per pubblicare messaggi su SNS argomenti di Amazon 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.

Cronologia SNS della documentazione di Amazon

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
`AmazonSNSFullAccess` e aggiornamenti delle politiche `AmazonSNSReadOnlyAccess` gestiti	Amazon SNS ha aggiunto nuove autorizzazioni AmazonSNSFullAccess e AmazonSNSReadOnlyAccess politiche gestite, che consentono un accesso aggiuntivo ad Amazon SNS tramite AWS Management Console.	24 settembre 2024
SNSIntegrazione con Amazon AWS End User Messaging SMS per la consegna dei SMS messaggi	SNSI clienti Amazon possono utilizzare nuove funzionalità come la gestione SMS delle risorse, la messaggistica bidirezionale, le autorizzazioni granulari per le risorse, le regole locali e la fatturazione centralizzata per tutti i AWS SMS messaggi senza apportare modifiche alle configurazioni o alla rete globale utilizzata da Amazon. AWS SMS SNS	24 settembre 2024

Supporto per argomenti relativi a Canada West (Calgary) FIFO	Amazon SNS supporta l'FIFOargomento in Canada occidentale (Calgary).	28 marzo 2024
SNSSMSSupporto Amazon in cinque nuove regioni	Amazon SNS ha aggiunto il SMS supporto alle seguenti regioni: Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Medio Oriente (UAE), Europa (Zurigo) ed Europa (Spagna).	8 febbraio 2024
Supporto per Firebase Cloud Messaging () v1 FCM HTTP	Amazon SNS supporta le credenziali FCM v1.	18 gennaio 2024
Amazon è SNS SMS supportato nell'area Asia-Pacifico (Giacarta)	Amazon SNS supporta la SMS messaggistica in Asia Pacifico (Giacarta).	14 dicembre 2023
AWS CloudFormation supporto per la configurazione `DeliveryStatusLogging` per argomenti Amazon SNS	AWS CloudFormation è disponibile il supporto per la configurazione `DeliveryStatusLogging` durante la creazione o l'aggiornamento di SNS argomenti Amazon.	7 dicembre 2023
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	Puoi sottoscrivere una SQS FIFO coda Amazon o una coda standard a un argomento Amazon SNSFIFO. Solo Amazon SQS FIFO Queues garantisce che i messaggi vengano ricevuti in ordine e senza duplicati.	14 settembre 2023
SMSè stato aggiunto il supporto per Israele (Tel Aviv)	Amazon SNS SMS è ora supportato nella regione di Israele (Tel Aviv).	28 agosto 2023
Support per il tracciamento attivo a raggi X aggiunto agli argomenti FIFO	In precedenza supportat a solo con gli argomenti SNS standard di Amazon, AWS X-Ray ora traccia e analizza le richieste degli utenti mentre passano dai tuoi FIFO argomenti agli abbonamenti Amazon Data Firehose AWS Lambda, SQS Amazon HTTP e /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 passano dagli argomenti SNS standard di Amazon agli abbonamenti Amazon Data Firehose AWS Lambda, SQS Amazon HTTP e /S endpoint.	8 febbraio 2023
Registrazione dell'ID mittente di Singapore	Sono state aggiunte istruzion i per la registrazione del mittente a Singapore. IDs	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
SHA256algoritmo hash aggiunto per la firma dei SNS messaggi Amazon	Support aggiunto per l'algorit mo SHA256 hash quando si utilizza Amazon SNS Message Signing.	15 settembre 2022
Regioni aggiuntive aggiunte alla messaggistica SMS	Amazon SNS supporta la SMS messaggistica 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 sui tuoi SNS argomenti Amazon utilizzando politiche di protezione dei dati per controllare e bloccare le informazioni sensibili che si spostano tra applicazioni o AWS servizi.	8 settembre 2022
Nuova procedura di registrazione per i numeri verdi	Support aggiunto per l'invio di SNS messaggi per Amazon utilizzando numeri di telefono gratuiti (TFN) a destinatari degli Stati Uniti.	1 agosto 2022
Support per i controlli di accesso basati sugli attributi () ABAC	È stato aggiunto il supporto per il controllo degli accessi basato sugli attributi () ABAC per azioni che includono e. API `Publish` `PublishBatch` ABACè una strategia di autorizzazione che definisce le autorizzazioni di accesso in base a tag che possono essere allegati a IAM risorse, come IAM utenti e ruoli, e a AWS risorse, come gli SNS argomenti di Amazon, 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 alla tua app iOS o macOS fornendo informazioni che ti identificano come sviluppatore dell'app.	28 ottobre 2021

I nuovi mittenti di SMS messaggi vengono inseriti nella sandbox SMS	La SMS sandbox serve a prevenire frodi e abusi e a proteggere la tua reputazione di mittente. Mentre il tuo AWS account è nella SMS sandbox, puoi inviare SMS messaggi solo a numeri di telefono di destinazione verificati.	1 giugno 2021
I nuovi mittenti di SMS messaggi vengono inseriti nella sandbox SMS	La SMS sandbox serve a prevenire frodi e abusi e a proteggere la tua reputazione di mittente. Mentre il tuo AWS account è nella SMS sandbox, puoi inviare SMS messaggi solo a numeri di telefono di destinazione verificati.	1 giugno 2021
Nuovi attributi per l'invio SMS di messaggi a destinatari in India	Due nuovi attributi, Entity ID e Template ID, sono ora necessari per l'invio di SMS messaggi 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 verificar e 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. Puoi invece utilizzare codici brevi, numeri verdi o un nuovo tipo di numero di origine chiamato 10. DLC	16 febbraio 2021
Support per i parametri Amazon CloudWatch di 1 minuto	La CloudWatch metrica di 1 minuto per Amazon SNS è ora disponibile in tutte le AWS regioni.	28 gennaio 2021
Supporto per gli endpoint Amazon Data Firehose	Puoi 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	Puoi utilizzare i numeri di origine per inviare messaggi di testo (). SMS	23 ottobre 2020

Support per SNS FIFO argomenti di Amazon	Per integrare applicazioni distribuite che richiedono la coerenza dei dati quasi in tempo reale, puoi utilizzare gli FIFO argomenti Amazon SNS first-in, first-out () con Amazon queues. SQS FIFO	22 ottobre 2020
La Amazon SNS Extended Client Library per Java è disponibile	Puoi usare questa libreria per pubblicare SNS messaggi Amazon di grandi dimensioni.	25 agosto 2020
SSEè disponibile nelle regioni della Cina	La crittografia lato server (SSE) per Amazon SNS è disponibile nelle regioni della Cina.	20 gennaio 2020
Support per l'utilizzo DLQs per l'acquisizione di messaggi non recapitabili	Per acquisire messaggi non recapitabili, puoi utilizzar e un Amazon SQS dead-letter queue () DLQ con un abbonamento Amazon. SNS	14 novembre 2019
Support per specificare valori di APNs intestazione personalizzati	È possibile specificare un valore di APNs intestazione personalizzato.	18 ottobre 2019
Support per il campo di intestazione `apns-push-type` " per APNs	Puoi utilizzare il campo di `apns-push-type` intestazione per le notifiche mobili inviate tramite. APNs	10 settembre 2019
Support per la risoluzione dei problemi tramite AWS X-Ray	È possibile utilizzare X-Ray per risolvere i problemi relativi ai messaggi che passano attraverso gli argomenti. SNS	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 a stringhe multiple, Amazon SNS consente qualsiasi cosa tranne la corrispondenza di più valori numerici.	5 luglio 2019
Le note SNS di rilascio di Amazon sono disponibili come RSS feed	Dopo il titolo di questa pagina (Cronologia della documentazione), scegli RSS.	22 giugno 2019

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.