



Guida per gli sviluppatori

Amazon Simple Notification Service



Amazon Simple Notification Service: Guida per gli sviluppatori

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è Amazon SNS?	1
Come funziona	1
Accedere ad Amazon SNS	2
Scenari comuni di Amazon SNS	3
Integrazione delle applicazioni	3
Avvisi dall' applicazione	4
Notifiche all'utente	4
Notifiche push per dispositivi mobili	5
Prezzi per Amazon SNS	5
Caratteristiche e funzionalità di Amazon SNS	5
Servizi comunemente condivisi	7
Lavorare con AWS SDKs	10
Crea un argomento e pubblica messaggi	12
Configurazione	12
Creazione di un account e di un utente IAM	12
Fasi successive	14
Fase 1: Creare un argomento	14
Console di gestione AWS	15
AWS SDKs	18
Fase 2: creazione di un abbonamento a un argomento	35
Per iscrivere un endpoint a un argomento Amazon SNS	35
Fase 3: Pubblicazione di un messaggio	37
Console di gestione AWS	37
AWS SDKs	38
Payload di messaggi di grandi dimensioni	66
Attributi di messaggio	77
Batch di messaggi	81
Fase 4: Eliminazione di un abbonamento e di un argomento	85
Console di gestione AWS	86
AWS SDKs	87
Fasi successive	98
Ordinamento e deduplicazione dei messaggi utilizzando argomenti FIFO	99
Argomenti FIFO ad alta velocità	99
Casi d'uso	99

Partizioni e distribuzione dei dati	100
Partizioni e distribuzione dei dati	101
Abilita un throughput elevato	101
Abilita la modalità di throughput elevato per le code Amazon SQS	102
Caso d'uso dell'argomento FIFO	102
Dettagli dell'ordine dei messaggi	104
Raggruppamento di messaggi	107
Distribuzione dei dati per gruppo di messaggi per migliorare le prestazioni IDs	108
Consegna dei messaggi	109
Filtro dei messaggi	110
Deduplicazione messaggi	111
Sicurezza dei messaggi	114
Durabilità dei messaggi	115
Archiviazione e riproduzione dei messaggi	117
Che cosa è l'archiviazione e riproduzione dei messaggi	117
Per i proprietari di argomenti	118
Per gli abbonati all'argomento	123
Esempi di codice	128
Esempio FIFO (AWS SDKs)	128
Esempio FIFO (CloudFormation)	141
Filtro dei messaggi	145
Ambito delle policy di filtro per le sottoscrizioni	145
Policy di filtro per le sottoscrizioni	146
Esempi di politiche di filtro di Amazon SNS	147
Vincoli delle policy di filtro	150
Logica AND/OR	154
Corrispondenza di chiave	158
Corrispondenza dei valori numerici	160
Corrispondenza dei valori di stringa	163
Applicazione di una policy di filtro per le sottoscrizioni	171
Console di gestione AWS	172
AWS CLI	173
AWS SDKs	174
API Amazon SNS	179
AWS CloudFormation	179
Rimozione di una policy di filtro per le sottoscrizioni	180

Utilizzando il Console di gestione AWS	180
Utilizzando il AWS CLI	180
Utilizzo dell'API Amazon SNS	181
Protezione dei dati dei messaggi	182
Cos'è la protezione dei dati dei messaggi	182
Perché utilizzare la protezione dei dati dei messaggi	183
Modifica della disponibilità della protezione dei dati dei messaggi Amazon SNS	183
Architettura alternativa	184
Visualizzazione delle politiche di protezione dei dati dei messaggi esistenti	185
Disattivazione della protezione dei dati dei messaggi Amazon SNS	185
policy di protezione dei dati	186
Cosa sono le policy di protezione dei dati?	186
Panoramica della struttura di una policy di protezione dei dati	187
Come faccio a determinare i principali IAM	189
Operazioni delle policy di protezione dei dati	190
Esempi di policy di protezione dei dati	198
Creazione di policy di protezione dei dati	205
Eliminazione di policy di protezione dei dati	216
Identificatori di dati	217
identificatori di dati gestiti	218
Identificatori di dati personalizzati	258
Consegna dei messaggi	261
Consegna di messaggi non elaborati	261
Abilitare il recapito di messaggi non elaborati utilizzando il Console di gestione AWS	262
Esempi di formati di messaggi	262
Attributi dei messaggi e consegna di messaggi non elaborati per gli abbonamenti Amazon SQS	263
Distribuzione tra più account	263
Creazione della sottoscrizione da parte del proprietario della coda	264
Creazione di una sottoscrizione da parte di un utente non proprietario della coda	266
Come faccio a forzare una sottoscrizione a richiedere l'autenticazione per le richieste di annullamento della sottoscrizione?	269
Consegna tra regioni	269
Regioni con consenso esplicito	269
Stato di consegna dei messaggi	273
Prerequisiti per la registrazione dello stato della consegna	273

Configurazione della registrazione dello stato di consegna utilizzando il Console di gestione AWS	274
Configurazione della registrazione dello stato della consegna utilizzando il AWS SDKs	275
AWS Esempi SDK per configurare gli attributi degli argomenti	278
Configurazione della registrazione dello stato di consegna tramite CloudFormation	286
Nuovi tentativi di consegna dei messaggi	288
Protocolli e policy di consegna	288
Fasi della policy di consegna	289
Creazione di una politica di consegna HTTP/S	290
Code di lettere morte	296
Perché le consegne dei messaggi non riescono?	297
Come funzionano le code DLQ?	298
Come vengono spostati i messaggi in una coda DLQ?	298
Come posso spostare i messaggi fuori da una coda DLQ?	298
Come posso monitorare e registrare code DLQ?	299
Configurazione di una coda DLQ	299
Archiviazione e analisi dei dati dei messaggi	305
Gestione e ottimizzazione delle risorse	306
Assegnazione di tag	306
Assegnazione di tag per l'allocazione dei costi	306
Assegnazione di tag per il controllo degli accessi	307
Assegnazione di tag per la ricerca e il filtro delle risorse	308
Configurare i tag	309
Sorgenti e destinazioni degli eventi Amazon SNS	315
Origini eventi	315
Analisi	315
Integrazione delle applicazioni	316
Gestione di costi e fatturazione	317
Applicazioni aziendali	318
Calcolo	318
Container	319
Coinvolgimento dei clienti	320
Database	321
Strumenti per sviluppatori	322
Web e dispositivi mobili front-end	323
Sviluppo di videogiochi	324

Internet of Things	325
Machine learning	325
Gestione e governance	327
Media	329
Migrazione e trasferimento	329
Reti e distribuzione di contenuti	330
Sicurezza, identità e conformità	331
Serverless	333
Archiviazione	333
Origini eventi aggiuntivi	335
Destinazioni degli eventi	336
A2A destinazioni	336
Destinazioni A2P	338
Application-to-application messaggistica	340
Flussi di distribuzione da Fanout a Firehose	340
Prerequisiti	341
Iscrizione di un flusso di distribuzione a un argomento	343
Gestione dei messaggi su più destinazioni con flussi di consegna	344
Esempio d'uso di archiviazione e analisi dei messaggi	358
Funzioni da Fanout a Lambda	371
Prerequisiti	371
Sottoscrizione di una funzione a un argomento	372
Fan-out a code Amazon SQS	373
Iscrizione di una coda a un argomento	374
Automatizza la messaggistica da Amazon SNS ad Amazon SQS con AWS CloudFormation	381
Notifiche Fanout sugli endpoint HTTPS	389
Iscrizione di un endpoint a un argomento	391
Verifica delle firme dei messaggi	400
Analisi dei formati di messaggi	405
Eventi Fanout su Event Fork Pipelines AWS	415
Come funziona AWS Event Fork Pipelines	416
Distribuzione di Event Fork AWS Pipelines	420
Distribuzione e test dell'applicazione di esempio Event Fork Pipelines	421
Sottoscrizione di una pipeline di eventi a un argomento	430
Usare Scheduler EventBridge	439

Impostazione del ruolo di esecuzione	440
Creare una pianificazione.	440
Risorse correlate	445
Application-to-person messaggistica	446
Messaggi di testo mobili	446
In che modo Amazon SNS consegna i miei messaggi SMS?	448
Nozioni di base	448
Identità di origine	455
Configurazioni	456
Le migliori pratiche per la messaggistica SMS	538
Invio di notifiche push per dispositivi mobili	554
Come funzionano le notifiche utente di Amazon SNS	555
Configurazione delle notifiche push con Amazon SNS	555
Configurare un'app mobile	556
Utilizzo di Amazon SNS per notifiche push su dispositivi mobili	575
Attributi per app	590
Eventi app per dispositivi mobili	593
Operazioni API push per dispositivi mobili	596
Errori comuni dell'API push mobile	598
TTL push per dispositivi mobili	612
Regioni supportate	614
Le migliori pratiche per le notifiche push su dispositivi mobili	615
Configurazione e gestione dell'abbonamento e-mail	616
Console di gestione AWS	617
AWS SDKs	618
Esempi di codice	652
Nozioni di base	653
Hello Amazon SNS	654
Azioni	666
Scenari	850
Costruisci un'app per inviare dati a una tabella DynamoDB	851
Costruzione di un'applicazione Amazon SNS	852
Creazione di un endpoint di piattaforma per notifiche push	854
Creazione di un'applicazione serverless per gestire foto	857
Creazione di un'applicazione Amazon Textract explorer	861
Crea un argomento Amazon SNS e pubblica messaggi	863

Creazione e pubblicazione su un argomento FIFO	869
Rilevamento di persone e oggetti in un video	881
Guida introduttiva a IoT Device Defender	882
Guida introduttiva a Config	903
Pubblicazione di un messaggio in un argomento	914
Pubblicazione di un messaggio di grandi dimensioni	920
Pubblicazione di un SMS	924
Pubblicazione di messaggi nelle code	933
Utilizzo di Gateway API per invocare una funzione Lambda	1070
Utilizzo degli eventi pianificati per invocare una funzione Lambda	1072
Esempi serverless	1074
Richiamo di una funzione Lambda da un trigger Amazon SNS	1075
Sicurezza	1085
Crittografia dei dati	1086
Protezione dei dati con la crittografia lato server	1086
Gestione delle chiavi	1089
Configurazione della crittografia degli argomenti con crittografia lato server	1096
Configurazione della crittografia degli argomenti con un abbonamento crittografato alla coda Amazon SQS	1098
Protezione del traffico con endpoint VPC	1103
Creazione di un endpoint VPC	1104
Creazione di una policy VPC	1106
Pubblicazione di un messaggio da un VPC	1107
Utilizzo di endpoint dual-stack per la connettività	1119
Sicurezza della protezione dei dati dei messaggi	1120
Gestione dell'identità e degli accessi	1120
Destinatari	1121
Autenticazione con identità	1121
Gestione dell'accesso tramite policy	1122
Controllo accessi	1124
Casi d'uso del controllo degli accessi	1125
Concetti chiave sulla politica di accesso	1125
Panoramica dell'architettura	1129
Utilizzo del linguaggio della policy di accesso	1130
Logica della valutazione	1131
Esempi di casi per il controllo degli accessi Amazon SNS	1136

Come funziona Amazon SNS con IAM	1148
AWS politiche gestite	1149
Operazioni di policy	1155
Risorse relative alle policy	1156
Chiavi di condizione delle policy	1156
ACLs	1157
ABAC	1157
Credenziali temporanee	1157
Autorizzazioni dell'entità principale	1158
Ruoli di servizio	1158
Ruoli collegati ai servizi	1158
Esempi di policy basate su identità	1159
Policy basate sull'identità	1162
Policy basate sulle risorse	1163
Utilizzo di policy basate su identità	1163
Gestione di politiche IAM personalizzate	1171
Utilizzo di credenziali temporanee	1172
Riferimento alle autorizzazioni API	1173
Registrazione di log e monitoraggio	1177
CloudTrail registri	1178
Argomenti di monitoraggio utilizzando CloudWatch	1188
Convalida della conformità	1205
Resilienza	1205
Sicurezza dell'infrastruttura	1206
Best practice di sicurezza	1206
Best practice di prevenzione	1207
Argomenti di risoluzione dei problemi AWS X-Ray	1212
Tracciamento attivo	1212
Permissions	1213
Abilitazione del tracciamento attivo	1213
Abilitazione del tracciamento attivo su un argomento Amazon SNS utilizzando l'SDK AWS	1214
Abilitazione del tracciamento attivo su un argomento Amazon SNS utilizzando la CLI	1214
AWS	1214
Abilitazione del tracciamento attivo su un argomento Amazon SNS utilizzando	1215
CloudFormation	1215
Verifica dell'abilitazione del tracciamento attivo	1215

Test in corso	1216
Cronologia della documentazione di Amazon SNS	1218
.....	mccxxix

Che cos'è Amazon SNS?

Amazon Simple Notification Service (Amazon SNS) è un servizio completamente gestito che fornisce il recapito dei messaggi dagli editori (produttori) agli abbonati (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.

Come funziona

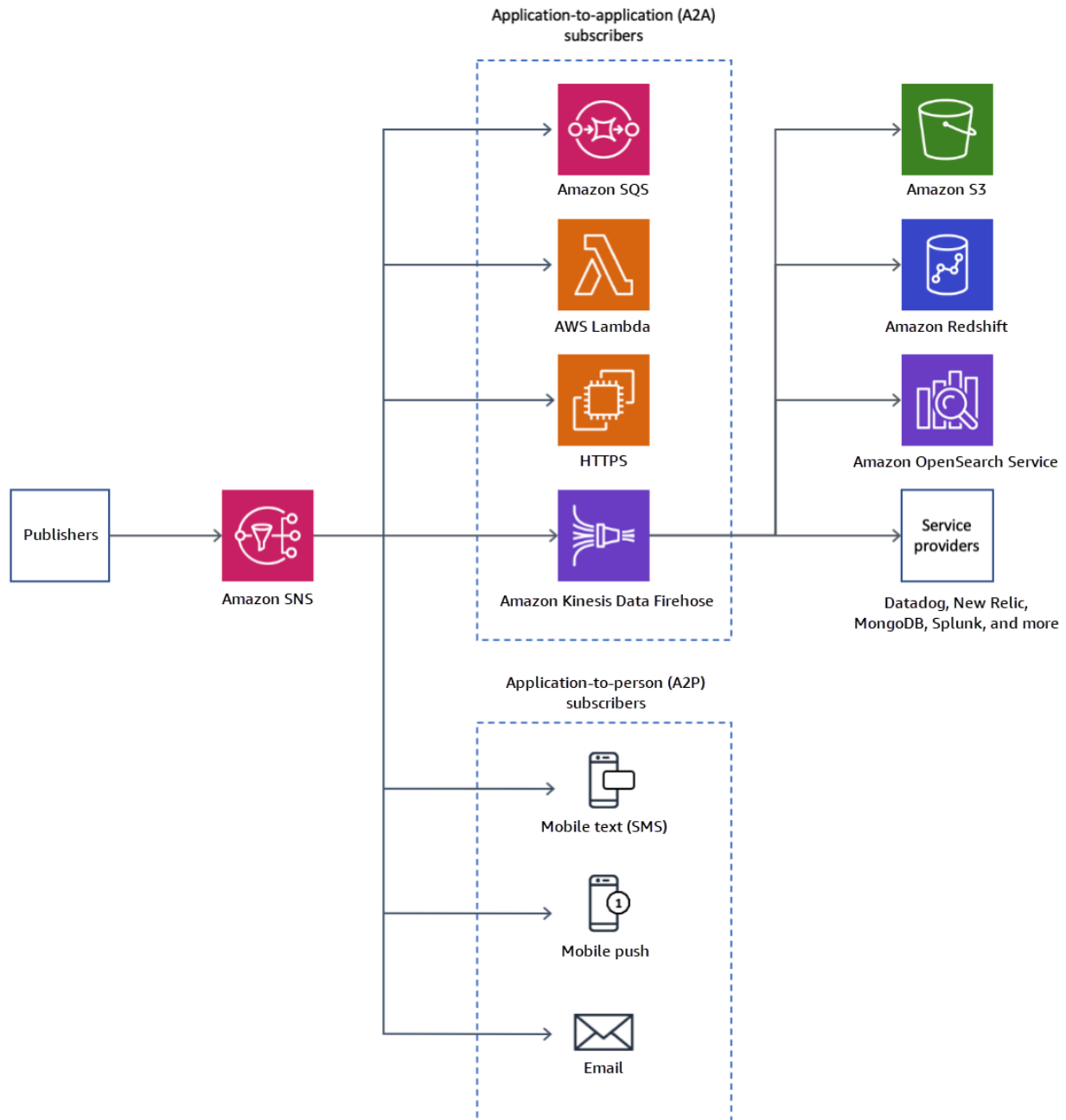
In SNS, gli editori inviano messaggi a un argomento, che funge da canale di comunicazione.

L'argomento funge da punto di accesso logico, garantendo che i messaggi vengano recapitati a più abbonati su piattaforme diverse.

Gli abbonati a un argomento SNS possono ricevere messaggi tramite endpoint diversi, a seconda del loro caso d'uso, ad esempio:

- Amazon SQS
- Lambda
- Endpoint HTTP (S)
- Email
- Notifiche push per dispositivi mobili
- Messaggi di testo mobili (SMS)
- Amazon Data Firehose
- Fornitori di servizi (ad esempio Datadog, MongoDB, Splunk)

SNS supporta la messaggistica Application-to-Application (A2A) e Application-to-Person (A2P), offrendo la flessibilità necessaria per inviare messaggi tra diverse applicazioni o direttamente a telefoni cellulari, indirizzi e-mail e altro ancora.



Accedere ad Amazon SNS

Puoi accedere e gestire Amazon SNS tramite la console o AWS CLI AWS SDKs, a seconda del metodo di interazione preferito. La console offre un'interfaccia grafica per le attività di base, mentre

SDKs fornisce funzionalità di configurazione AWS CLI e automazione avanzate per casi d'uso più complessi.

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

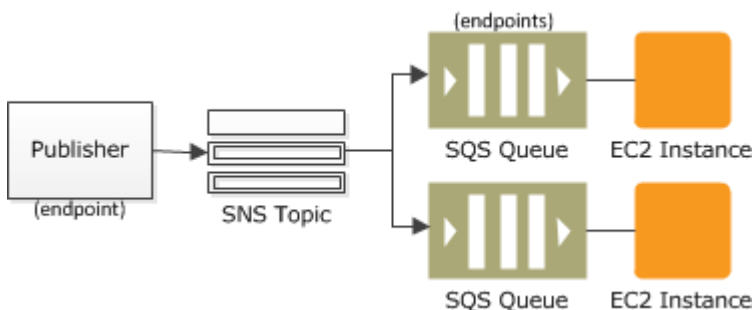
Scenari comuni di Amazon SNS

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

Integrazione delle applicazioni

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

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



Un altro modo per utilizzare il "fan-out" è replicare i dati inviati al tuo ambiente di produzione con il tuo ambiente di prova. Parlando dell'esempio precedente in modo più approfondito, è possibile

sottoscrivere un'altra coda SQS allo stesso argomento SNS per i nuovi ordini in entrata. Quindi, collegando questa nuova coda SQS al proprio ambiente di prova, è possibile continuare a migliorare e testare la propria applicazione utilizzando i dati ricevuti dall'ambiente di produzione.

Important

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

Per maggiori informazioni, consulta le seguenti risorse:

- [Flussi di distribuzione da Fanout a Firehose](#)
- [Fanout delle notifiche Amazon SNS alle funzioni Lambda per l'elaborazione automatizzata](#)
- [Esegui l'eliminazione delle notifiche Amazon SNS alle code di Amazon SQS per l'elaborazione asincrona](#)
- [Fanout delle notifiche Amazon SNS agli endpoint HTTPS](#)
- [Elaborazione basata sugli eventi con Amazon SNS AWS e servizi di elaborazione, storage, database e rete](#)

Avvisi dall' applicazione

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

Notifiche all'utente

Amazon SNS può inviare messaggi e-mail push e messaggi di testo (messaggi SMS) a singoli o gruppi. Ad esempio, è possibile inviare conferme di ordine e-commerce come notifiche utente. Per avere ulteriori informazioni su come usare Amazon SNS per inviare messaggi SMS, consultare [Messaggi di testo 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. Per esempio, puoi utilizzare Amazon SNS per inviare notifiche di aggiornamento a un'app. Il messaggio di notifica può includere un collegamento per eseguire il download e installare l'aggiornamento. Per avere ulteriori informazioni su come usare Amazon SNS per inviare messaggi SMS, consultare [Invio di notifiche push per dispositivi mobili con Amazon SNS](#).

Prezzi per Amazon SNS

Amazon SNS non ha costi iniziali. I pagamenti vengono effettuati in base al numero di messaggi pubblicati, al numero di notifiche inviate e a eventuali chiamate API aggiuntive per la gestione di argomenti e abbonamenti. I prezzi di spedizione variano in base al tipo di endpoint. È possibile iniziare gratuitamente con il livello gratuito di Amazon SNS. Per informazioni, consulta i [prezzi degli SMS in tutto il mondo](#).

Caratteristiche e funzionalità di Amazon SNS

Amazon SNS offre un set completo di funzionalità progettate per migliorare la messaggistica tra applicazioni e utenti. Queste funzionalità consentono comunicazioni senza interruzioni, consegna sicura dei messaggi e 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 flussi di distribuzione, funzioni Lambda, code Amazon SQS, HTTP/S endpoint ed 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 utente agli abbonati come applicazioni mobili, numeri di cellulare e indirizzi e-mail.

Argomenti standard e FIFO

[Gli argomenti FIFO](#) garantiscono un ordine rigoroso dei messaggi, il raggruppamento e la deduplicazione dei messaggi, consentendo a FIFO e alle code standard di iscriversi all'elaborazione dei messaggi. [Gli argomenti standard](#) vengono utilizzati quando l'ordine 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 cooperano per garantire la durabilità dei messaggi:

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

Archiviazione, riproduzione e analisi dei messaggi

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

Attributi di messaggio

[Attributi messaggio di Amazon SNS](#) ti consentono 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 argomenti di Amazon SNS, 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, consulta [the section called “Protezione del traffico con endpoint VPC”](#).

AWS servizi comunemente usati con Amazon SNS

Integra Amazon SNS con più Cloud AWS servizi per potenziare la gestione dei messaggi, migliorare il controllo degli accessi, abilitare l'elaborazione basata sugli eventi e automatizzare le risorse. Questa integrazione ottimizza le prestazioni, rafforza la sicurezza e semplifica le operazioni.

Amazon CloudWatch

Amazon CloudWatch fornisce monitoraggio e osservabilità per Amazon SNS, aiutandoti a tracciare la consegna dei messaggi, rilevare anomalie e risolvere i problemi. Con il CloudWatch puoi:

- Monitora i parametri di Amazon SNS, come il numero di messaggi pubblicati, recapitati o non riusciti per argomenti e abbonamenti.
- Imposta CloudWatch allarmi per attivare azioni automatizzate quando i parametri di Amazon SNS superano le soglie predefinite, ad esempio problemi di consegna elevati o limitazioni.
- Usa CloudWatch Logs per acquisire lo stato di consegna di Amazon SNS per i messaggi inviati agli endpoint HTTP/S, Lambda e Amazon SQS per il debug e il controllo.

Per ulteriori informazioni, consulta [Monitoraggio degli argomenti di Amazon SNS tramite CloudWatch](#).

Amazon SQS

Amazon SQS è un servizio di accodamento dei messaggi completamente gestito che consente comunicazioni sicure, durature e scalabili tra componenti software distribuiti. Aiuta a disaccoppiare l'architettura delle applicazioni mediante il buffering dei messaggi, garantendo una consegna affidabile e prevenendo i guasti del sistema dovuti alla perdita dei messaggi. Amazon SQS si integra con Amazon SNS nei seguenti modi:

- Code di [lettere non recapitabili: Amazon SNS può indirizzare i messaggi non recapitabili a una coda](#) di lettere non recapitate di Amazon SQS per la risoluzione dei problemi e la rielaborazione.
- Abbonamenti per argomenti: puoi [abbonare una coda Amazon SQS a un argomento](#) Amazon SNS, permettendo ad Amazon SNS di inviare messaggi a più consumatori utilizzando Amazon SQS.
- [Supporto per le code FIFO: le code](#) FIFO di Amazon SQS possono essere sottoscritte agli argomenti FIFO di Amazon SNS, garantendo un ordine rigoroso dei messaggi e l'elaborazione esattamente una volta. Le [code standard di Amazon SQS](#) possono anche abbonarsi agli argomenti di Amazon SNS, ma non garantiscono la consegna ordinata dei messaggi o la deduplicazione.

CloudFormation

CloudFormation automatizza il provisioning e la gestione delle AWS risorse, inclusi gli argomenti e gli abbonamenti di Amazon SNS, utilizzando l'infrastruttura come codice (IaC). Con, puoi: CloudFormation

- Definisci argomenti, abbonamenti e autorizzazioni di Amazon SNS in un modello riutilizzabile e controllato dalla versione.
- Garantisci una distribuzione coerente delle risorse Amazon SNS su più regioni Account AWS .
- Aggiorna o modifica le configurazioni di Amazon SNS utilizzando set di modifiche senza intervento manuale.

Per ulteriori informazioni, consulta la [Guida per l'utente AWS CloudFormation](#).

AWS CloudTrail

CloudTrail offre visibilità sull'attività delle API per Amazon SNS, aiutandoti a monitorare e controllare l'accesso ad argomenti, abbonamenti e messaggi di Amazon SNS. Con CloudTrail, puoi:

- Tieni traccia delle chiamate API effettuate ad Amazon SNS, incluso chi ha avuto accesso o modificato argomenti, abbonamenti e autorizzazioni.
- Rileva attività non autorizzate o impreviste analizzando i log per scopi di sicurezza e conformità.
- Effettua l'integrazione con Amazon CloudWatch o AWS Security Hub CSPM crea avvisi basati su azioni insolite di Amazon SNS.

Per ulteriori informazioni, consulta [Registrazione delle chiamate API AWS SNS utilizzando AWS CloudTrail](#).

AWS Lambda

AWS Lambda è un servizio di elaborazione serverless che esegue automaticamente il codice in risposta agli eventi, eliminando la necessità di fornire o gestire i server. Consente di creare applicazioni basate sugli eventi, scalabili automaticamente ed eseguite in un ambiente di elaborazione ad alta disponibilità.

Amazon SNS si integra con Lambda consentendoti di sottoscrivere una funzione Lambda a un argomento di Amazon SNS. Quando un argomento Amazon SNS riceve un messaggio, può attivare la funzione Lambda, abilitando l'elaborazione, l'automazione e l'esecuzione della logica dell'applicazione in tempo reale. Questa integrazione viene comunemente utilizzata per:

- [Elaborazione basata sugli eventi](#): attiva automaticamente le funzioni in risposta ai messaggi Amazon SNS.
- [Trasformazione dei dati](#): modifica o filtra i messaggi Amazon SNS prima di inoltrarli ad altri servizi.
- Flussi di lavoro automatizzati: elabora le notifiche per gli avvisi delle applicazioni, il monitoraggio del sistema o l'orchestrazione degli eventi.

AWS Identity and Access Management (IAM)

IAM fornisce un controllo sicuro degli accessi alle AWS risorse, consentendoti di gestire chi può accedere ai tuoi argomenti di Amazon SNS, quali azioni possono eseguire e in quali condizioni. Con IAM, puoi:

- Autentica utenti e servizi prima che possano interagire con gli argomenti di Amazon SNS.
- Definisci autorizzazioni dettagliate per specificare su quali argomenti o ruoli di Amazon SNS gli utenti o i ruoli possono pubblicare, sottoscrivere o gestire.
- Utilizza policy basate sull'identità per applicare le migliori pratiche di sicurezza, come la limitazione dell'accesso a indirizzi IP o condizioni specifici. Account AWS

Per ulteriori informazioni, consulta [Utilizzo di policy basate su identità con Amazon SNS](#).

AWS Key Management Service (AWS KMS)

AWS KMS migliora la sicurezza di Amazon SNS abilitando la crittografia lato server (SSE) per la riservatezza dei messaggi. Con, puoi: AWS KMS

- Crittografa i messaggi Amazon SNS inattivi AWS utilizzando chiavi di crittografia gestite o gestite dal cliente (). CMKs
- Controlla l'accesso agli argomenti di Amazon SNS definendo policy chiave granulari che limitano chi può pubblicare o abbonarsi.

- Garantisci la conformità ai requisiti di sicurezza e normativi controllando l'utilizzo delle chiavi tramite. AWS CloudTrail

Per ulteriori informazioni, consulta [Gestione delle chiavi e dei costi di crittografia di Amazon SNS](#).

AWS X-Ray

X-Ray fornisce funzionalità di tracciamento per Amazon SNS, aiutandoti ad analizzare ed eseguire il debug del flusso di messaggi attraverso la tua architettura basata sugli eventi. Con X-Ray puoi:

- Tieni traccia della consegna dei messaggi Amazon SNS su più endpoint Servizi AWS, come Lambda, Amazon SQS e HTTP/S.
- Identifica i problemi di latenza visualizzando il tempo impiegato dai messaggi per essere pubblicati, recapitati ed elaborati.
- Rileva errori e nuovi tentativi nei flussi di messaggi di Amazon SNS per risolvere problemi di consegne non riuscite o tempi di elaborazione lenti.

Per ulteriori informazioni, consulta [Tracciamento attivo in Amazon SNS](#).

Utilizzo di Amazon SNS con un SDK AWS

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

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice

Documentazione sugli SDK	Esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice
AWS Strumenti per PowerShell	AWS Strumenti per PowerShell esempi di codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK per Rust	AWS SDK per Rust esempi di codice
AWS SDK per SAP ABAP	AWS SDK per SAP ABAP esempi di codice
AWS SDK per Swift	AWS SDK per Swift esempi di codice

Per esempi specifici di Amazon SNS, consultare [Esempi di codice per l'utilizzo di Amazon SNS AWS SDKs](#).

 Esempio di disponibilità

Non riesci a trovare quello che ti serve? Richiedi un esempio di codice utilizzando il link
Fornisci un feedback nella parte inferiore di questa pagina.

Crea un argomento Amazon SNS e pubblica messaggi

Questo argomento fornisce i passaggi fondamentali per la gestione delle risorse Amazon SNS, concentrandosi in particolare su argomenti, abbonamenti e pubblicazione di messaggi. Innanzitutto, configurerai le autorizzazioni di accesso necessarie per Amazon SNS, assicurandoti di disporre delle autorizzazioni corrette per creare e gestire risorse Amazon SNS. Successivamente, creerai un nuovo argomento Amazon SNS, 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 modo Amazon SNS distribuisce efficacemente il messaggio a tutti gli endpoint sottoscritti. Infine, imparerai come eliminare sia l'abbonamento che l'argomento, completando il ciclo di vita delle risorse Amazon SNS che hai gestito. Questo approccio ti offre una chiara comprensione delle operazioni fondamentali di Amazon SNS, fornendoti le competenze pratiche necessarie per gestire i flussi di lavoro di messaggistica utilizzando la console Amazon SNS.

Configurazione degli accessi per Amazon SNS

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

Crea un utente Account AWS e un utente IAM

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

Iscriviti per un Account AWS

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

Per iscriverti a un Account AWS

1. Apri la <https://portal.aws.amazon.com/billing/registrazione>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata o un messaggio di testo e ti verrà chiesto di 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 le [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 [Console di gestione AWS](#) come proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Signing in as the root user](#) della Guida per l'utente di Accedi ad AWS .

2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

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

Crea un utente con accesso amministrativo

1. Abilita il Centro identità IAM.

Per istruzioni, consulta [Abilitazione del AWS IAM Identity Center](#) nella Guida per l'utente di AWS IAM Identity Center .

2. Nel Centro identità IAM, assegna 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 l'impostazione predefinita IAM Identity Center directory](#) nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere come utente del Centro identità IAM, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente del Centro identità IAM.

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

Assegnazione dell'accesso ad altri utenti

1. Nel Centro identità IAM, crea un set di autorizzazioni conforme alla best practice per l'applicazione di autorizzazioni con il privilegio minimo.

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 .

Fasi successive

Ora che sei pronto a lavorare con Amazon SNS, inizia con:

1. [Creare un argomento Amazon SNS](#)
2. [Creazione di un abbonamento a un argomento di Amazon SNS](#)
3. [Pubblicazione di un messaggio Amazon SNS](#)
4. [Eliminazione di un argomento e di un abbonamento Amazon SNS](#)

Creare un argomento Amazon SNS

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

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

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

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

Important

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

Per creare un argomento utilizzando il Console di gestione AWS

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 [console Amazon SNS](#).
2. Esegui una delle seguenti operazioni:
 - Se non è mai stato creato alcun argomento in Account AWS precedenza, leggi la descrizione di Amazon SNS nella home page.
 - Se gli argomenti sono già stati creati sotto il tuo Account AWS nome, nel pannello di navigazione, scegli Argomenti.
3. Nella pagina Topics (Argomenti), seleziona Create new topic (Crea nuovo argomento).
4. Nella pagina Create topic (Crea argomento), nella sezione Details (Dettagli), eseguire queste operazioni:
 - a. Per Type (Tipo), scegli un tipo di argomento (standard o FIFO).

- b. Immetti un nome per l'argomento. Per un [Argomento FIFO](#), aggiungere .fifo alla fine del nome.
- c. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.

 Important

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

 Note

Per gli abbonamenti e-mail ed email-json, il nome visualizzato dell'argomento viene utilizzato come nome del mittente solo per i normali messaggi di notifica. Le e-mail di conferma dell'iscrizione e dell'annullamento dell'iscrizione utilizzano sempre "AWS Notifiche» come nome del mittente, indipendentemente dall'impostazione del nome visualizzato dell'argomento.

- d. (Facoltativo) Per un argomento FIFO, puoi scegliere Deduplicazione dei messaggi basata sul contenuto per abilitare la deduplicazione predefinita dei messaggi. Per ulteriori informazioni, consulta [Deduplicazione dei messaggi Amazon SNS per argomenti FIFO](#).
5. (Facoltativo) Espandere la sezione Encryption (Crittografia) e procedere come segue. Per ulteriori informazioni, consulta [Protezione dei dati di Amazon SNS con crittografia lato server](#).
- a. Scegliere Enable encryption (Abilita crittografia).
 - b. Specificare la chiave. AWS KMS Per ulteriori informazioni, consulta [Termini chiave](#).

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

 Important

Se non sei il proprietario della KMS, oppure se effettui l'accesso con un account che non dispone delle autorizzazioni `kms:ListAliases` e `kms:DescribeKey`, non puoi visualizzare le informazioni sulla KMS sulla console Amazon SNS.

Chiedi al proprietario della KMS di concederti queste autorizzazioni. Per esempi e ulteriori informazioni consulta [AWS KMS Autorizzazioni API: e Documentazione su operazioni e risorse](#) nella AWS Key Management Service Guida per sviluppatori.

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

 Note

Ricorda quanto segue:

- La prima volta che usi Console di gestione AWS per specificare il KMS AWS gestito per Amazon SNS per un argomento AWS KMS , crea AWS il KMS gestito per Amazon SNS.
- In alternativa, la prima volta che utilizzi l'Publishazione su un argomento con SSE abilitato, AWS KMS crea il KMS AWS gestito per Amazon SNS.

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

 Note

Per istruzioni sulla creazione di chiavi personalizzate KMSs, consulta [Creating Keys](#) nella Developer Guide AWS Key Management Service

- Per utilizzare un ARN KMS personalizzato dal AWS tuo account o da AWS un altro account, inseriscilo nel campo della chiave KMS.
6. (Facoltativo) Per impostazione predefinita, solo il proprietario dell'argomento può pubblicare o sottoscrivere l'argomento. Per configurare autorizzazioni di accesso aggiuntive, espandi la sezione Access policy (Policy di accesso) . Per ulteriori informazioni, consultare [Identity and](#)

[Access Management in Amazon SNS](#) e [Esempi di casi per il controllo degli accessi Amazon SNS](#).

Note

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

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

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

Il nome dell'argomento, l'ARN, (opzionale) il nome visualizzato e l'ID AWS account del proprietario dell'argomento vengono visualizzati nella sezione Dettagli.

11. Copiare l'ARN dell'argomento negli appunti, ad esempio:

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

Per creare un argomento utilizzando un SDK AWS

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

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

.NET

SDK per .NET

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 con un nome di specifico.

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

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

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

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

    /// <summary>
    /// Creates a new SNS topic using the supplied topic name.
    /// </summary>
    /// <param name="client">The initialized SNS client object used to
    /// create the new topic.</param>
    /// <param name="topicName">A string representing the topic name.</param>
    /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
```

```
public static async Task<string>
CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
{
    var request = new CreateTopicRequest
    {
        Name = topicName,
    };

    var response = await client.CreateTopicAsync(request);

    return response.TopicArn;
}
```

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

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

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

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

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

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

if (outcome.IsSuccess()) {
    topicARNResult = outcome.GetResult().GetTopicArn();
    std::cout << "Successfully created an Amazon SNS topic " << topicName
                << " with topic ARN '" << topicARNResult
                << "'." << std::endl;
}
else {
    std::cerr << "Error creating topic " << topicName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
    topicARNResult.clear();
}

return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Creazione di un argomento SNS

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

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

Output:

```
{
```

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

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

- Per i dettagli sull'API, consulta AWS CLI Command [CreateTopic](#) Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sns"  
    "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// 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 i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }
}
```



```
public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

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

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

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

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

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

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

$SnsClient = new SnsClient([
```

```

        'profile' => 'default',
        'region' => 'us-east-1',
        'version' => '2010-03-31'
    ]);

    $topicname = 'myTopic';

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

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

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

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

```

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

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

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def create_topic(
        self,
```

```

        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
        deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
        try:
            # Add .fifo suffix for FIFO topics
            if is_fifo and not topic_name.endswith('.fifo'):
                topic_name += '.fifo'

            attributes = {}
            if is_fifo:
                attributes['FifoTopic'] = 'true'
                if content_based_deduplication:
                    attributes['ContentBasedDeduplication'] = 'true'

            response = self.sns_client.create_topic(
                Name=topic_name,
                Attributes=attributes
            )

            topic_arn = response['TopicArn']
            logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
            return topic_arn

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')
            logger.error(f"Error creating topic {topic_name}: {error_code} -
            {e}")
            raise

```

- Per i dettagli sull'API, consulta [CreateTopic AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

  puts "Creating the topic '#{topic_name}'..."
end
```

```
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 per Ruby](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per Ruby API Reference.

Rust

SDK per 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 i dettagli sulle API, consulta il riferimento [CreateTopic](#) all'API AWS SDK for Rust.

SAP ABAP

SDK per 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 i dettagli sulle API, [CreateTopic](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS  
  
let config = try await SNSClient.SNSClientConfiguration(region: region)  
let snsClient = SNSClient(config: config)  
  
let output = try await snsClient.createTopic(  
    input: CreateTopicInput(name: name)  
)
```

```
guard let arn = output.topicArn else {  
    print("No topic ARN returned by Amazon SNS.")  
    return  
}
```

- Per i dettagli sull'API, consulta la [CreateTopic](#) guida di riferimento all'API AWS SDK for Swift.

Creazione di un abbonamento a un argomento di Amazon SNS

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

Note

Gli endpoint HTTP (S), gli indirizzi e-mail e altre AWS risorse Account AWS richiedono la conferma dell'abbonamento prima di poter ricevere messaggi.

Per iscrivere un endpoint a un argomento Amazon SNS

La sottoscrizione di un endpoint a un argomento Amazon SNS 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 e-mail o altri servizi, e consente una comunicazione senza interruzioni tra i sistemi.

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione sinistro scegli Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
 - a. Per argomento ARN, scegli l'Amazon Resource Name (ARN) di un argomento. Questo valore è l' AWS ARN generato quando hai creato l'argomento Amazon SNS, ad esempio.
`arn:aws:sns:us-east-2:123456789012:your_topic`
 - b. Per Protocollo, scegli un tipo di endpoint. I tipi di endpoint disponibili sono:

- [HTTP/HTTPS](#)
- [Email/Email-JSON](#)
-
- [Amazon SQS](#)

 Note

Per sottoscrivere ad un [argomento SNS FIFO](#) scegli questa opzione.

- [AWS Lambda](#)
 - [Endpoint applicazione piattaforma](#)
 - [SMS](#)
- c. Per Endpoint, inserisci il valore dell'endpoint, ad esempio un indirizzo e-mail o l'ARN di una coda Amazon SQS.
 - d. Solo endpoint Firehose: per il ruolo di sottoscrizione ARN, specifica l'ARN del ruolo IAM che hai creato per la scrittura nei flussi di distribuzione Firehose. Per ulteriori informazioni, consulta [Prerequisiti per la sottoscrizione dei flussi di distribuzione di Firehose agli argomenti di Amazon SNS](#).
 - e. (Facoltativo) Per gli endpoint Firehose, Amazon SQS HTTP/S , puoi anche abilitare il recapito di messaggi non elaborati. Per ulteriori informazioni, consulta [Consegna di messaggi non elaborati Amazon SNS](#).
 - f. (Facoltativo) Per configurare un criterio di filtro, espandere la sezione policy di filtro della sottoscrizione. Per ulteriori informazioni, consulta [Policy di filtro degli abbonamenti Amazon SNS](#).
 - g. (Facoltativo) Per abilitare il filtro basato sul payload, imposta Filter Policy Scope su MessageBody. Per ulteriori informazioni, consulta [Ambito delle policy di filtro per le sottoscrizioni Amazon SNS](#).
 - h. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta [Code di lettere non ricevute di Amazon SNS](#).
 - i. Scegli Create Subscription (Crea sottoscrizione).

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

Pubblicazione di un messaggio Amazon SNS

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

Per pubblicare messaggi su argomenti di Amazon SNS utilizzando Console di gestione AWS

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

La console apre la pagina Pubblica messaggio in argomento.

4. Nella sezione Basic details (Dettagli di base), eseguire le seguenti:
 - a. (Facoltativo) Immettere un messaggio Oggetto.
 - b. Per un [Argomento FIFO](#), immettere un ID gruppo di messaggi. L'ID del gruppo di messaggi è obbligatorio per gli argomenti FIFO. I messaggi nello stesso gruppo di messaggi vengono recapitati nell'ordine in cui vengono pubblicati.
 - c. (Facoltativo) Per un argomento standard, inserisci un ID del gruppo di messaggi. Questo ID viene inoltrato a tutti gli abbonamenti standard Amazon SQS e non viene utilizzato o inviato a nessun altro tipo di endpoint.
 - d. Per un argomento FIFO, immettere un ID deduplicazione messaggi. Questo ID è facoltativo se è stata attivata l'impostazione Deduplicazione dei messaggi basata sul contenuto per l'argomento.
 - e. (Facoltativo) Per [Notife push per dispositivi](#), immettere 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. Scegliere Custom payload for each delivery protocol (Payload personalizzato per ogni protocollo di consegna) e inserire l'oggetto JSON per definire il messaggio da inviare a ogni protocollo.

Per ulteriori informazioni, consulta [Pubblicazione di notifiche Amazon SNS con payload specifici della piattaforma](#).

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

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

 Note

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

- b. Immettere un attributo Nome, ad esempio customer_interests.
- c. Immettere un attributo Valore, ad esempio ["soccer", "rugby", "hockey"].

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

Per ulteriori informazioni, consulta [Attributi messaggio di Amazon SNS](#).

7. Seleziona Publish message (Pubblica messaggio).

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

Per pubblicare un messaggio su un argomento utilizzando un SDK AWS

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

Gli esempi di codice seguenti mostrano come utilizzare Publish.

.NET

SDK per .NET

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 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($"{i + 1}. {_tones[i]}");
            }

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

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

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

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

```



```

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

```

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

```

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

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>

```

```

        {
            { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
        };
    }

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

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

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

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

```

```

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

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

return outcome.IsSuccess();
}

```

Pubblica un messaggio con un attributo.

```

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

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

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

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

if (filteringMessages && askYesNoQuestion(
    "Add an attribute to this message? (y/n) ")) {
    for (size_t i = 0; i < TONES.size(); ++i) {
        std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
    }
    int selection = askQuestionForIntRange(
        "Enter a number for an attribute. ",
        1, static_cast<int>(TONES.size()));
}

```

```

        Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
        messageAttributeValue.SetDataType("String");
        messageAttributeValue.SetStringValue(TONES[selection - 1]);
        request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
    }

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

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

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

        return false;
    }
}

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

CLI

AWS CLI

Esempio 1: pubblicazione di un messaggio in un argomento

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

```

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

```

Contenuto di `message.txt`:

```
Hello World
Second Line
```

Output:

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

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

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

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

Output:

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

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dei comandi della AWS CLI .

Go

SDK per Go V2

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// 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 informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Go .

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.PublishRequest;  
import software.amazon.awssdk.services.sns.model.PublishResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-  
started.html  
 */  
public class PublishTopic {  
    public static void main(String[] args) {
```

```
final String usage = ""

    Usage:    <message> <topicArn>

    Where:
        message - The message text to send.
        topicArn - The ARN of the topic to publish.
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String message = args[0];
String topicArn = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();
pubTopic(snsClient, message, topicArn);
snsClient.close();
}

public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.getMessageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

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


- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

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

```

    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
console.log(response);
// {
//   '$metadata': {
//     httpStatusCode: 200,
//     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
//     extendedRequestId: undefined,
//     cfId: undefined,
//     attempts: 1,
//     totalRetryDelay: 0
//   },
//   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx'
// }
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 per JavaScript](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per JavaScript .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

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

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Kotlin.

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

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

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

```
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

PowerShell

Strumenti per PowerShell V4

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

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -  
Message "Hello" -MessageAttribute  
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{'DataType='String'  
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 informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dei cmdlet di AWS Strumenti per PowerShell (V4).

Strumenti per PowerShell V5

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

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -  
Message "Hello" -MessageAttribute  
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String'  
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 informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dei cmdlet di AWS Strumenti per PowerShell (V5).

Python

SDK per Python (Boto3)

Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Pubblicare un messaggio 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

```

Publicare 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

```

```

class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod

```

```
def from_client(cls) -> 'SnsWrapper':
    """
    Create an SnsWrapper instance using a default boto3 client.

    :return: An instance of this class.
    """
    sns_client = boto3.client('sns')
    return cls(sns_client)

def publish_message(
    self,
    topic_arn: str,
    message: str,
    tone_attribute: Optional[str] = None,
    deduplication_id: Optional[str] = None,
    message_group_id: Optional[str] = None
) -> str:
    """
    Publish a message to an SNS topic.

    :param topic_arn: The ARN of the SNS topic.
    :param message: The message content to publish.
    :param tone_attribute: Optional tone attribute for message filtering.
    :param deduplication_id: Optional deduplication ID for FIFO topics.
    :param message_group_id: Optional message group ID for FIFO topics.
    :return: The message ID of the published message.
    :raises ClientError: If the message publication fails.
    """
    try:
        publish_args = {
            'TopicArn': topic_arn,
            'Message': message
        }

        # Add message attributes if tone is specified
        if tone_attribute:
            publish_args['MessageAttributes'] = {
                'tone': {
                    'DataType': 'String',
                    'StringValue': tone_attribute
                }
            }
    
```

```

        # Add FIFO-specific parameters
        if message_group_id:
            publish_args['MessageGroupId'] = message_group_id

        if deduplication_id:
            publish_args['MessageDeduplicationId'] = deduplication_id

        response = self.sns_client.publish(**publish_args)

        message_id = response['MessageId']
        logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
        return message_id

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error publishing message to topic: {error_code} -
{e}")
        raise

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API SDK for Python (Boto3)AWS .

Ruby

SDK per Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

# Service class for sending messages using Amazon Simple Notification Service
(SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client

```

```

def initialize(sns_client)
  @sns_client = sns_client
  @logger = Logger.new($stdout)
end

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

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

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

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

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

Rust

SDK per 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 informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per 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.
```

Pubblica un messaggio con attributi relativi a un argomento.

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

Pubblica un messaggio multiformato su un argomento.

```
" Build JSON message structure for multi-format message
DATA(lv_json_message) = |\{ "default": "{ iv_default_message }", "sms":
"{ iv_sms_message }", "email": "{ iv_email_message }" \}|.
```

```

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

```

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

let output = try await snsClient.publish(
    input: PublishInput(
        message: message,
        topicArn: arn
    )
)

guard let messageId = output.messageId else {
    print("No message ID received from Amazon SNS.")
    return
}

```



```
}  
  
print("Published message with ID \"(messageId)")
```

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dell'API AWS SDK per Swift.

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

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

Note

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

Libreria client estesa Amazon SNS per Java

Prerequisiti

Di seguito sono indicati i prerequisiti per l'utilizzo di [Libreria client ampia di Amazon SNS per Java](#):

- Un SDK. AWS L'esempio in questa pagina utilizza l'SDK AWS Java. Per installare e configurare l'SDK, consulta [Configurare l' AWS SDK for Java](#) nella AWS SDK per Java Developer Guide.
- 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 per Java sviluppatori.

- Java 8 o versione successiva.
- La libreria client ampia di Amazon SNS per Java (disponibile anche da [Maven](#)).

Configurazione dello storage dei messaggi

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

- Soglia di dimensioni dei messaggi personalizzate: i messaggi con payload e attributi che superano questa dimensione vengono archiviati automaticamente in Amazon S3.
- **alwaysThroughS3**flag — Imposta questo valore per `true` forzare l'archiviazione di tutti i payload dei messaggi in Amazon S3. Ad esempio:

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

- Chiave KMS personalizzata: la chiave da utilizzare per la crittografia lato server nel tuo bucket Amazon S3.
- Nome del bucket: il nome del bucket Amazon S3 per l'archiviazione dei payload dei messaggi.

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

L'esempio di codice seguente mostra come:

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

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

SDK per Java 1.x

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

public class Example {

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

        // Message threshold controls the maximum message size that will be
        allowed to
        // be published
```

```

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

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

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

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

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

        // Initialize SNS extended client
        // PayloadSizeThreshold triggers message content storage in S3 when
the
        // threshold is exceeded
        // To store all messages content in S3, use AlwaysThroughS3 flag

```

```
        final SNSExtendedClientConfiguration snsExtendedClientConfiguration
= new SNSExtendedClientConfiguration()
        .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

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

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

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

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

Altri protocolli per endpoint

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

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

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

Libreria client estesa Amazon SNS per Python

Prerequisiti

Di seguito sono indicati i prerequisiti per l'utilizzo della [Libreria client ampia di Amazon SNS per Python](#):

- Un AWS SDK. L'esempio in questa pagina utilizza AWS Python SDK Boto3. Per installare e configurare l'SDK, consultare la documentazione [SDK AWS per Python](#).
- E Account AWS con le credenziali appropriate. Per crearne uno Account AWS, vai alla [AWS home page](#), quindi scegli Crea un AWS account. Segui le istruzioni.

Per ulteriori informazioni sulle credenziali, consultare [Credenziali](#) nella Guida per sviluppatori di SDK AWS per Python.

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

Configurazione dello storage dei messaggi

Gli attributi seguenti sono disponibili su Boto3 Amazon [SNS](#) Client, Topic [PlatformEndpoint](#) e Objects per configurare le opzioni di storage dei messaggi di Amazon S3.

- **large_payload_support**— Il nome del bucket Amazon S3 che memorizzerà messaggi di grandi dimensioni.
- **use_legacy_attribute**— Se True, allora tutti i messaggi pubblicati utilizzano l'attributo Legacy reserved message (SQSLargePayloadSize) anziché l'attuale attributo riservato del messaggio (ExtendedPayloadSize).
- **message_size_threshold** – La soglia per l'archiviazione del messaggio nel bucket dei messaggi di grandi dimensioni. Non può essere minore 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_client**— L'oggetto Boto3 Amazon client S3 da utilizzare per archiviare oggetti su Amazon S3. Usalo se desideri controllare il client Amazon S3 (ad esempio, configurazioni o credenziali Amazon S3 personalizzate). L'impostazione predefinita è al primo utilizzo se non è stata `boto3.client("s3")` impostata in precedenza.

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

L'esempio di codice seguente mostra come:

- Crea un argomento Amazon SNS di esempio e una coda Amazon SQS.
- Allega la policy alla coda di Amazon SQS per ricevere il messaggio dall'argomento Amazon SNS.
- Iscriviti alla coda per ricevere messaggi dall'argomento.
- Pubblica un messaggio di prova utilizzando il client esteso Amazon SNS, la risorsa Topic e PlatformEndpoint la risorsa.
- Il payload del messaggio è memorizzato in Amazon S3 e il riferimento ad esso è pubblicato.
- Stampa il messaggio pubblicato dalla coda insieme al messaggio originale recuperato da Amazon S3.

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

```
import boto3
from sns_extended_client import SNSExtendedClientSession
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store" # S3 bucket with the
    given bucket name is a resource which is created and accessible with the given AWS
    credentials
TOPIC_NAME = "---TOPIC-NAME---"
QUEUE_NAME = "---QUEUE-NAME---"

def allow_sns_to_write_to_sqs(topicarn, queuearn):
    policy_document = """{{
        "Version": "2012-10-17",
        "Statement":[
```

```

        {{
            "Sid":"MyPolicy",
            "Effect":"Allow",
            "Principal" : {{ "AWS" : "*" }},
            "Action":"SQS:SendMessage",
            "Resource": "{}",
            "Condition":{{
                "ArnEquals":{{
                    "aws:SourceArn": "{}"
                }}
            }}
        }}
    ]
    }}"".format(queuearn, topicarn)

return policy_document

def get_msg_from_s3(body,sns_extended_client):
    """Handy Helper to fetch message from S3"""
    json_msg = loads(body)
    s3_object = sns_extended_client.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

def fetch_and_print_from_sqs(sqs, queue_url,sns_extended_client):
    sqs_msg = sqs.receive_message(
        QueueUrl=queue_url,
        AttributeNames=['All'],
        MessageAttributeNames=['All'],
        VisibilityTimeout=0,
        WaitTimeSeconds=0,
        MaxNumberOfMessages=1
    ).get("Messages")[0]

    message_body = sqs_msg.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is:
    {}\n".format(get_msg_from_s3(message_body,sns_extended_client)))

    # Delete the Processed Message
    sqs.delete_message(

```



```

        QueueUrl=queue_url,
        ReceiptHandle=sqs_msg['ReceiptHandle']
    )

sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
sns_topic_arn = create_topic_response.get("TopicArn")

# create and subscribe an sqs queue to the sns client
sqs = boto3.client("sqs", region_name="us-east-1")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
sqs_queue_arn = sqs.get_queue_attributes(
    QueueUrl=demo_queue_url, AttributeNames=["QueueArn"]
)["Attributes"].get("QueueArn")

# Adding policy to SQS queue such that SNS topic can send msg to SQS queue
policy_json = allow_sns_to_write_to_sqs(sns_topic_arn, sqs_queue_arn)
response = sqs.set_queue_attributes(
    QueueUrl = demo_queue_url,
    Attributes = {
        'Policy' : policy_json
    }
)

# Set the RawMessageDelivery subscription attribute to TRUE if you want to use
# SQSExtendedClient to help with retrieving msg from S3
sns_extended_client.subscribe(TopicArn=sns_topic_arn, Protocol="sqs",
Endpoint=sqs_queue_arn
, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# Change default s3_client attribute of sns_extended_client to use 'us-east-1' region
sns_extended_client.s3_client = boto3.client("s3", region_name="us-east-1")

# Below is the example that all the messages will be sent to the S3 bucket
sns_extended_client.always_through_s3 = True
sns_extended_client.publish(
    TopicArn=sns_topic_arn, Message="This message should be published to S3"
)
print("\n\nPublished using SNS extended client:")

```

```
fetch_and_print_from_sqs(sqs, demo_queue_url, sns_extended_client) # Prints message
    stored in s3

# Below is the example that all the messages larger than 32 bytes will be sent to the
    S3 bucket
print("\nUsing decreased message size threshold:")

sns_extended_client.always_through_s3 = False
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=sns_topic_arn,
    Message="This message should be published to S3 as it exceeds the limit of the 32
    bytes",
)

fetch_and_print_from_sqs(sqs, demo_queue_url, sns_extended_client) # Prints message
    stored in s3

# Below is the example to publish message using the SNS.Topic resource
sns_extended_client_resource = SNSExtendedClientSession().resource(
    "sns", region_name="us-east-1"
)

topic = sns_extended_client_resource.Topic(sns_topic_arn)
topic.large_payload_support = s3_extended_payload_bucket

# Change default s3_client attribute of topic to use 'us-east-1' region
topic.s3_client = boto3.client("s3", region_name="us-east-1")

topic.always_through_s3 = True
# Can Set custom S3 Keys to be used to store objects in S3
topic.publish(
    Message="This message should be published to S3 using the topic resource",
    MessageAttributes={
        "S3Key": {
            "DataType": "String",
            "StringValue": "347c11c4-a22c-42e4-a6a2-9b5af5b76587",
        }
    },
)
print("\nPublished using Topic Resource:")
fetch_and_print_from_sqs(sqs, demo_queue_url, topic)
```

```
# Below is the example to publish message using the SNS.PlatformEndpoint resource
sns_extended_client_resource = SNSExtendedClientSession().resource(
    "sns", region_name="us-east-1"
)

platform_endpoint = sns_extended_client_resource.PlatformEndpoint(sns_topic_arn)
platform_endpoint.large_payload_support = s3_extended_payload_bucket

# Change default s3_client attribute of platform_endpoint to use 'us-east-1' region
platform_endpoint.s3_client = boto3.client("s3", region_name="us-east-1")

platform_endpoint.always_through_s3 = True
# Can Set custom S3 Keys to be used to store objects in S3
platform_endpoint.publish(
    Message="This message should be published to S3 using the PlatformEndpoint
resource",
    MessageAttributes={
        "S3Key": {
            "DataType": "String",
            "StringValue": "247c11c4-a22c-42e4-a6a2-9b5af5b76587",
        }
    },
)
print("\nPublished using PlatformEndpoint Resource:")
fetch_and_print_from_sqs(sqs, demo_queue_url, platform_endpoint)
```

Output

Published using SNS extended client:

```
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
{"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"}]
```

Message Stored in S3 Bucket is: This message should be published to S3

Using decreased message size threshold:

```
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
{"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"}]
```

Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds the limit of the 32 bytes

Published using Topic Resource:

```
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
{"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxxx"}]
Message Stored in S3 Bucket is: This message should be published to S3 using the topic
resource

Published using PlatformEndpoint Resource:
Published Message: ["software.amazon.payloadoffloading.PayloadS3Pointer",
{"s3BucketName": "extended-client-bucket-store", "s3Key": "xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxxx"}]
Message Stored in S3 Bucket is: This message should be published to S3 using the
PlatformEndpoint resource
```

Attributi messaggio di Amazon SNS

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

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

Per informazioni sull'invio di messaggi con attributi utilizzando o il, consulta il tutorial. Console di gestione AWS AWS SDK per Java [Per pubblicare messaggi su argomenti di Amazon SNS utilizzando Console di gestione AWS](#)

Note

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

Puoi utilizzare gli attributi di messaggio anche per strutturare il messaggio di notifica push per gli endpoint mobili. In questo scenario, gli attributi di messaggio sono usati solo per strutturare il

messaggio di notifica push. Gli attributi non vengono distribuiti all'endpoint così come sono quando si inviano i messaggi con gli attributi di messaggio agli endpoint Amazon SQS.

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

Note

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

Elementi dell'attributo di messaggio e convalida

Ogni attributo di messaggio è costituito dai seguenti elementi:

- **Name** – Il nome dell'attributo del messaggio può contenere i seguenti caratteri: A-Z, a-z, 0-9, sottolineatura (`_`), trattino (`-`) e punto (`.`). Il nome non deve iniziare o terminare con un punto e non deve avere punti in successione. Il nome rispetta la distinzione tra maiuscole e minuscole e deve essere univoco tra tutti i nomi di attributo per il messaggio. Il nome può contenere fino a 256 caratteri. Il nome non può iniziare con `AWS.` o `Amazon.` (o qualsiasi variazione in maiuscole/minuscole) perché questi prefissi sono riservati per l'uso da parte di Amazon Web Services.
- **Type** – I tipi di dati di attributo di messaggio supportati sono `String`, `String.Array`, `Number` e `Binary`. Il tipo di dati ha le stesse restrizioni sul contenuto del corpo del messaggio. 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 tipo stringa, l'attributo value deve seguire le stesse restrizioni di contenuto del corpo del messaggio. Tuttavia, se l'attributo message viene utilizzato per il filtraggio, il valore deve essere una stringa JSON valida per garantire la compatibilità con le politiche di filtro degli abbonamenti di Amazon SNS. Per ulteriori informazioni, consulta l'operazione [Pubblicare](#) nella Documentazione di riferimento API di Amazon Simple Notification Service.

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

Tipi di dati dell'attributo di messaggio e convalida

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

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

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

Note

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

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

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

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

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

- **Binary** – Gli attributi di tipo binario possono archiviare qualsiasi tipo di dati binari, ad esempio dati compressi, dati crittografati o immagini.

Attributi di messaggio riservati per notifiche push per dispositivi mobili

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

Servizio di notifiche push	Attributo di messaggio riservato
ADM	<code>AWS.SNS.MOBILE.ADM.TTL</code>
APNs ¹	<code>AWS.SNS.MOBILE.APNS_MDM.TTL</code>
	<code>AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL</code>
	<code>AWS.SNS.MOBILE.APNS_PASSBOOK.TTL</code>
	<code>AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL</code>
	<code>AWS.SNS.MOBILE.APNS_SANDBOX.TTL</code>
	<code>AWS.SNS.MOBILE.APNS_VOIP.TTL</code>
	<code>AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL</code>
	<code>AWS.SNS.MOBILE.APNS.COLLAPSE_ID</code>
	<code>AWS.SNS.MOBILE.APNS.PRIORITY</code>
	<code>AWS.SNS.MOBILE.APNS.PUSH_TYPE</code>
	<code>AWS.SNS.MOBILE.APNS.TOPIC</code>
	<code>AWS.SNS.MOBILE.APNS.TTL</code>
Baidu	<code>AWS.SNS.MOBILE.BAIDU.DeployStatus</code>
	<code>AWS.SNS.MOBILE.BAIDU.MessageKey</code>
	<code>AWS.SNS.MOBILE.BAIDU.MessageType</code>
	<code>AWS.SNS.MOBILE.BAIDU.TTL</code>
FCM	<code>AWS.SNS.MOBILE.FCM.TTL</code>

Servizio di notifiche push	Attributo di messaggio riservato
	<code>AWS.SNS.MOBILE.GCM.TTL</code>
macOS	<code>AWS.SNS.MOBILE.MACOS_SANDBOX.TTL</code>
	<code>AWS.SNS.MOBILE.MACOS.TTL</code>
MPNS	<code>AWS.SNS.MOBILE.MPNS.NotificationClass</code>
	<code>AWS.SNS.MOBILE.MPNS.TTL</code>
	<code>AWS.SNS.MOBILE.MPNS.Type</code>
WNS	<code>AWS.SNS.MOBILE.WNS.CachePolicy</code>
	<code>AWS.SNS.MOBILE.WNS.Group</code>
	<code>AWS.SNS.MOBILE.WNS.Match</code>
	<code>AWS.SNS.MOBILE.WNS.SuppressPopup</code>
	<code>AWS.SNS.MOBILE.WNS.Tag</code>
	<code>AWS.SNS.MOBILE.WNS.TTL</code>
	<code>AWS.SNS.MOBILE.WNS.Type</code>

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

Batch di messaggi Amazon SNS

Cos'è il batch di messaggi?

Un'alternativa alla pubblicazione di messaggi su argomenti Standard o FIFO in richieste API `Publish` individuali, è l'utilizzo dell'API `PublishBatch` di Amazon SNS per pubblicare fino a 10 messaggi in una singola richiesta API. L'invio di messaggi in batch può aiutare a ridurre i costi associati alla connessione di applicazioni distribuite ([Messaggistica A2A](#)) o all'invio di notifiche alle persone ([Messaggistica A2P](#)) con Amazon SNS di un fattore fino a 10. Amazon SNS ha quote su

quanti messaggi è possibile pubblicare su un argomento al secondo in base alla regione in cui operi. Consultare [Amazon SNS endpoints and quotas](#) (Endpoint e quote di Amazon SNS) nella guida Riferimenti generali di AWS per ulteriori informazioni sulle quote delle API.

Note

La dimensione totale aggregata di tutti i messaggi inviati in una singola richiesta PublishBatch API non può superare 262.144 byte (256 KB).
L'API PublishBatch utilizza la stessa operazione API Publish per le policy IAM.

Come funziona il batch di messaggi?

Pubblicazione di messaggi con l'API PublishBatch è simile alla pubblicazione di messaggi con l'API Publish. La differenza principale è che ogni messaggio all'interno di una richiesta API PublishBatch deve essere assegnato un ID batch univoco (fino a 80 caratteri). In questo modo, Amazon SNS può restituire risposte API individuali per ogni messaggio all'interno di un batch per confermare che ogni messaggio è stato pubblicato o che si è verificato un errore. Per i messaggi pubblicati su argomenti FIFO, oltre a includere l'assegnazione di un ID batch univoco, è comunque necessario includere un MessageDeduplicationID e MessageGroupId per ogni singolo messaggio.

Esempi

Pubblicazione di un batch di 10 messaggi su un argomento Standard

```
// Imports
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishBatchRequest;
import software.amazon.awssdk.services.sns.model.PublishBatchRequestEntry;
import software.amazon.awssdk.services.sns.model.PublishBatchResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

// Code
private static final int MAX_BATCH_SIZE = 10;
```

```
public static void publishBatchToTopic(SnsClient snsClient, String topicArn, int
batchSize) {
    try {
        // Validate the batch size
        if (batchSize > MAX_BATCH_SIZE) {
            throw new IllegalArgumentException("Batch size cannot exceed " +
MAX_BATCH_SIZE);
        }

        // Create the batch entries
        List<PublishBatchRequestEntry> entries = IntStream.range(0, batchSize)
            .mapToObj(i -> PublishBatchRequestEntry.builder()
                .id("id" + i)
                .message("message" + i)
                .build())
            .collect(Collectors.toList());

        // Build the batch request
        PublishBatchRequest request = PublishBatchRequest.builder()
            .topicArn(topicArn)
            .publishBatchRequestEntries(entries)
            .build();

        // Publish the batch request
        PublishBatchResponse response = snsClient.publishBatch(request);

        // Handle successful messages
        response.successful().forEach(success -> {
            System.out.println("Successful Batch Id: " + success.id());
            System.out.println("Message Id: " + success.messageId());
        });

        // Handle failed messages
        response.failed().forEach(failure -> {
            System.err.println("Failed Batch Id: " + failure.id());
            System.err.println("Error Code: " + failure.code());
            System.err.println("Sender Fault: " + failure.senderFault());
            System.err.println("Error Message: " + failure.message());
        });

    } catch (SnsException e) {
        // Log and handle exceptions
        System.err.println("SNS Exception: " + e.awsErrorDetails().errorMessage());
    } catch (IllegalArgumentException e) {
```

```
        System.err.println("Validation Error: " + e.getMessage());
    }
}
```

Pubblicazione di un batch di 10 messaggi su un argomento FIFO

```
// Imports
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishBatchRequest;
import software.amazon.awssdk.services.sns.model.PublishBatchRequestEntry;
import software.amazon.awssdk.services.sns.model.PublishBatchResponse;
import software.amazon.awssdk.services.sns.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sns.model.SnsException;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToFifoTopic(SnsClient snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
            .mapToObj(i -> PublishBatchRequestEntry.builder()
                .id("id" + i)
                .message("message" + i)
                .messageGroupId("groupId")
                .messageDeduplicationId("deduplicationId" + i)
                .build())
            .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = PublishBatchRequest.builder()
            .topicArn(topicArn)
            .publishBatchRequestEntries(entries)
            .build();

        // Publish the batch request
        PublishBatchResponse response = snsClient.publishBatch(request);

        // Handle the successfully sent messages
```

```
response.successful().forEach(success -> {
    System.out.println("Batch Id for successful message: " + success.id());
    System.out.println("Message Id for successful message: " +
success.messageId());
    System.out.println("Sequence Number for successful message: " +
success.sequenceNumber());
});

// Handle the failed messages
response.failed().forEach(failure -> {
    System.err.println("Batch Id for failed message: " + failure.id());
    System.err.println("Error Code for failed message: " + failure.code());
    System.err.println("Sender Fault for failed message: " +
failure.senderFault());
    System.err.println("Failure Message for failed message: " +
failure.message());
});

} catch (SnsException e) {
    // Handle any exceptions from the request
    System.err.println("SNS Exception: " + e.awsErrorDetails().errorMessage());
}
}
```

Eliminazione di un argomento e di un abbonamento Amazon SNS

Quando un argomento viene eliminato, le sottoscrizioni associate vengono eliminate in modo asincrono. Sebbene i clienti possano ancora accedere a questi abbonamenti, gli abbonamenti non sono più associati all'argomento, anche se si ricrea l'argomento utilizzando lo stesso nome. Se un 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 SNS elimina automaticamente gli abbonamenti non confermati dopo 48 ore, ad eccezione degli abbonamenti e-mail, che vengono eliminati dopo 30 giorni.

Important

Gli argomenti Amazon SNS eliminati non possono essere recuperati. Ti consigliamo di eliminare gli abbonamenti prima di eliminare un argomento.

Una volta eliminato, l'abbonamento non può essere recuperato. Le notifiche pubblicate sull'argomento Amazon SNS non verranno recapitate all'abbonamento eliminato. Dovrai ricreare l'abbonamento per ricevere le notifiche.

Per eliminare un argomento o un abbonamento Amazon SNS utilizzando Console di gestione AWS

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

Per eliminare un argomento utilizzando il Console di gestione AWS

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

La console elimina l'argomento.

Per eliminare un abbonamento utilizzando il Console di gestione AWS

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

La console elimina la sottoscrizione.

Per eliminare un abbonamento e un argomento utilizzando un SDK AWS

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

Gli esempi di codice seguenti mostrano come utilizzare DeleteTopic.

.NET

SDK per .NET

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](#).

Elimina un argomento in base all'ARN dell'argomento.

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

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Eliminazione di un argomento SNS

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

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

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteTopic AWS CLI Command Reference](#).

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
  "context"  
  "encoding/json"  
  "log"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/service/sns"  
  "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// 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 i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
```

```

public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
                result.sdkHttpResponse().statusCode());

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

```

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

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

```
// }  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun deleteSNSTopic(topicArnVal: String) {  
    val request =  
        DeleteTopicRequest {  
            topicArn = topicArnVal  
        }  
  
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->  
        snsClient.deleteTopic(request)  
        println("$topicArnVal was successfully deleted.")  
    }  
}
```

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

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

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

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""
```

```
def __init__(self, sns_client: Any) -> None:
    """
    Initialize the SnsWrapper.

    :param sns_client: A Boto3 Amazon SNS client.
    """
    self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

- Per i dettagli sull'API, consulta [DeleteTopic AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per 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 i dettagli sulle API, [DeleteTopic](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS  
  
let config = try await SNSClient.SNSClientConfiguration(region: region)  
let snsClient = SNSClient(config: config)
```



```
_ = try await snsClient.deleteTopic(  
    input: DeleteTopicInput(topicArn: arn)  
)
```

- Per i dettagli sull'API, consulta la [DeleteTopic](#) guida di riferimento all'API AWS SDK for Swift.

Fasi successive

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

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

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

[Questo argomento fornisce informazioni sulle caratteristiche e le funzionalità degli argomenti FIFO di Amazon SNS \(First-In-First-Out\) e su come si integrano con le code FIFO di Amazon SQS.](#) 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 copre i casi d'uso specifici in cui gli argomenti FIFO di Amazon SNS 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 gli argomenti FIFO di Amazon SNS e la loro integrazione con le code FIFO di Amazon SQS.

Argomenti FIFO ad alto throughput in Amazon SNS

Gli argomenti FIFO ad alta velocità in Amazon SNS gestiscono in modo efficiente un'elevata velocità di trasmissione dei messaggi mantenendo al contempo un ordine rigoroso dei messaggi, garantendo affidabilità e scalabilità per le applicazioni che elaborano numerosi messaggi. Questa soluzione è ideale per scenari che richiedono sia un throughput elevato che la consegna di messaggi ordinati. Per migliorare la velocità di trasmissione dei messaggi utilizzando argomenti FIFO ad alta velocità, si consiglia di aumentare il numero di gruppi di messaggi. Per ulteriori informazioni sulle quote di messaggi ad alta velocità di trasmissione, consulta le quote dei [servizi Amazon SNS](#) nel. Riferimenti generali di Amazon Web Services

Casi d'uso per un throughput elevato per argomenti FIFO di Amazon SNS

I seguenti casi d'uso evidenziano le diverse applicazioni degli argomenti FIFO ad alta produttività, dimostrandone l'efficacia in diversi settori e scenari:

- **Elaborazione dei dati in tempo reale:** le applicazioni che gestiscono flussi di dati in tempo reale, come l'elaborazione di eventi o l'inserimento di dati di telemetria, possono trarre vantaggio dagli argomenti FIFO ad alta velocità per gestire l'afflusso continuo di messaggi preservandone l'ordine per un'analisi accurata.
- **Elaborazione degli ordini di e-commerce:** nelle piattaforme di e-commerce in cui il mantenimento dell'ordine delle transazioni con i clienti è fondamentale, gli argomenti FIFO ad alta produttività garantiscono che gli ordini vengano consegnati in sequenza e senza ritardi, anche durante l'alta stagione degli acquisti.
- **Servizi finanziari:** gli istituti finanziari che gestiscono dati di negoziazione o transazionali ad alta frequenza si affidano ad argomenti FIFO ad alta velocità per elaborare i dati di mercato e le transazioni con una latenza minima, rispettando al contempo i severi requisiti normativi per l'ordinazione dei messaggi.
- **Streaming multimediale:** le piattaforme di streaming e i servizi di distribuzione multimediale utilizzano argomenti FIFO ad alta velocità per gestire la distribuzione di file multimediali e contenuti in streaming, garantendo esperienze di riproduzione fluide per gli utenti e mantenendo il corretto ordine di distribuzione dei contenuti.

Partizioni e distribuzione dei dati per un throughput elevato per argomenti FIFO di Amazon SNS

Con argomenti a throughput elevato, Amazon SNS distribuisce i dati relativi agli argomenti FIFO tra le partizioni. Una partizione è un'allocazione di capacità per un argomento che viene replicata automaticamente su più zone di disponibilità all'interno di una Regione AWS. Non si gestiscono le partizioni. Invece, Amazon SNS gestisce automaticamente le partizioni per tuo conto, in base alla velocità di ingresso.

Per gli argomenti FIFO, Amazon SNS modifica il numero di partizioni in un argomento nelle seguenti situazioni:

- Se la velocità di pubblicazione corrente si avvicina o supera quella supportata dalle partizioni esistenti, vengono allocate partizioni aggiuntive fino a quando l'argomento non raggiunge la quota regionale. Per informazioni sulle quote, consulta le quote del [servizio Amazon SNS](#) nel [Riferimenti generali di Amazon Web Services](#).
- Se le partizioni correnti sono poco utilizzate, il numero di partizioni può essere ridotto.

La gestione delle partizioni avviene automaticamente in background ed è trasparente per le tue applicazioni. Il tuo argomento e i tuoi messaggi sono sempre disponibili.

Note

La limitazione dell'API Temporary [Publish](#) può verificarsi se aumenti improvvisamente e in modo significativo il traffico verso il tuo argomento inviando più volte il volume normale. Questa limitazione può durare fino alla durata della finestra di deduplicazione, mentre l'argomento può essere ampliato per adattarsi all'aumento del traffico.

Distribuzione dei dati per gruppo di messaggi IDs

Quando si pubblica un messaggio su un argomento FIFO, Amazon SNS utilizza il valore dell'ID del gruppo di messaggi di ogni messaggio come input per una funzione hash interna. Il valore di output della funzione hash determina quale partizione elabora il messaggio, uno o più gruppi di messaggi IDs possono essere gestiti da una determinata partizione.

Note

Amazon SNS è ottimizzato per la distribuzione uniforme degli elementi tra le partizioni di un argomento FIFO, indipendentemente dal numero di partizioni. AWS consiglia di utilizzare un gruppo di messaggi IDs che può avere un gran numero di valori distinti.

Abilita un throughput elevato sul tuo argomento Amazon SNS FIFO

[Per impostazione predefinita, gli argomenti FIFO di Amazon SNS sono configurati per la deduplicazione a livello di argomento, controllata dall'attributo `topic FifoThroughputScope` impostato su `Topic` e hanno quote di throughput più limitate, consulta le quote dei servizi Amazon SNS nel. \[Riferimenti generali di Amazon Web Services\]\(#\)](#)

Per abilitare un throughput elevato per il tuo argomento Amazon SNS FIFO, aggiorna `FifoThroughputScope` l'attributo a `MessageGroup`. Questa modifica può essere effettuata tramite la console o utilizzando l'SDK AWS CLI and e può anche essere impostata durante la creazione dell'argomento, scelta consigliata da Amazon SNS per garantire la migliore esperienza del cliente e ridurre le possibilità che l'argomento venga limitato.

 Important

Una volta abilitato l'accesso `FifoThroughputScope` a un argomento `MessageGroup`, non è possibile ripristinarne la velocità effettiva. `Topic`

Abilita la modalità High Throughput per qualsiasi coda FIFO Amazon SQS sottoscritta

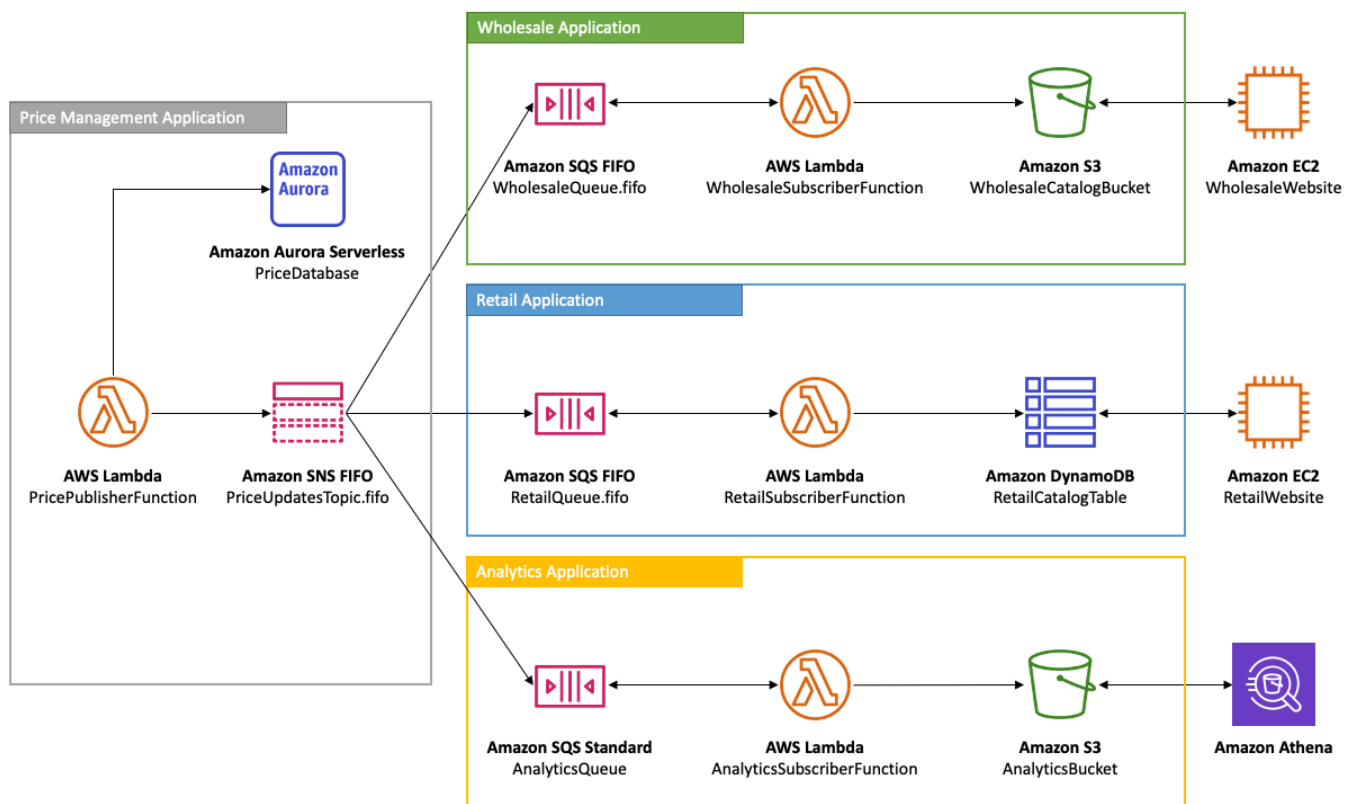
Quando pubblichi sul tuo argomento Amazon SNS FIFO con throughput elevato abilitato e sono sottoscritte una o più code FIFO di Amazon SQS, ti consigliamo di abilitare un throughput elevato sulle code FIFO di Amazon SQS per consentire una distribuzione ottimale dell'argomento Amazon SNS FIFO ad alta velocità. Per ulteriori informazioni, consulta [High throughput for FIFO queues](#) nella Guida per gli sviluppatori di Amazon Simple Queue Service.

Esempio di utilizzo dell'argomento FIFO di Amazon SNS

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

- I responsabili dell'inventario utilizzano un'applicazione di gestione dei prezzi per impostare il prezzo per ogni articolo in magazzino. In questa azienda, i prezzi dei prodotti possono cambiare in base alla fluttuazione del cambio di valuta, alla domanda del mercato e ai cambiamenti nella strategia di vendita. L'applicazione di gestione dei prezzi utilizza una funzione AWS Lambda che pubblica gli aggiornamenti dei prezzi in un argomento FIFO di Amazon SNS ogni volta che i prezzi cambiano.
- Un'applicazione all'ingrosso fornisce il backend per un sito web in cui carrozzerie auto e produttori di automobili possono acquistare parti auto della società in massa. Per ottenere le notifiche di modifica del prezzo, l'applicazione all'ingrosso sottoscrive la coda FIFO SQS all'argomento FIFO di Amazon SNS dell'applicazione di gestione dei prezzi.
- Un'applicazione di vendita al dettaglio fornisce il backend per un altro sito web in cui i proprietari di auto e gli appassionati di tuning auto possono acquistare singoli pezzi di auto per i loro veicoli. Per ottenere le notifiche di modifica del prezzo, l'applicazione di vendita al dettaglio sottoscrive anche la coda FIFO di Amazon SQS all'argomento FIFO di Amazon SNS dell'applicazione di gestione dei prezzi.

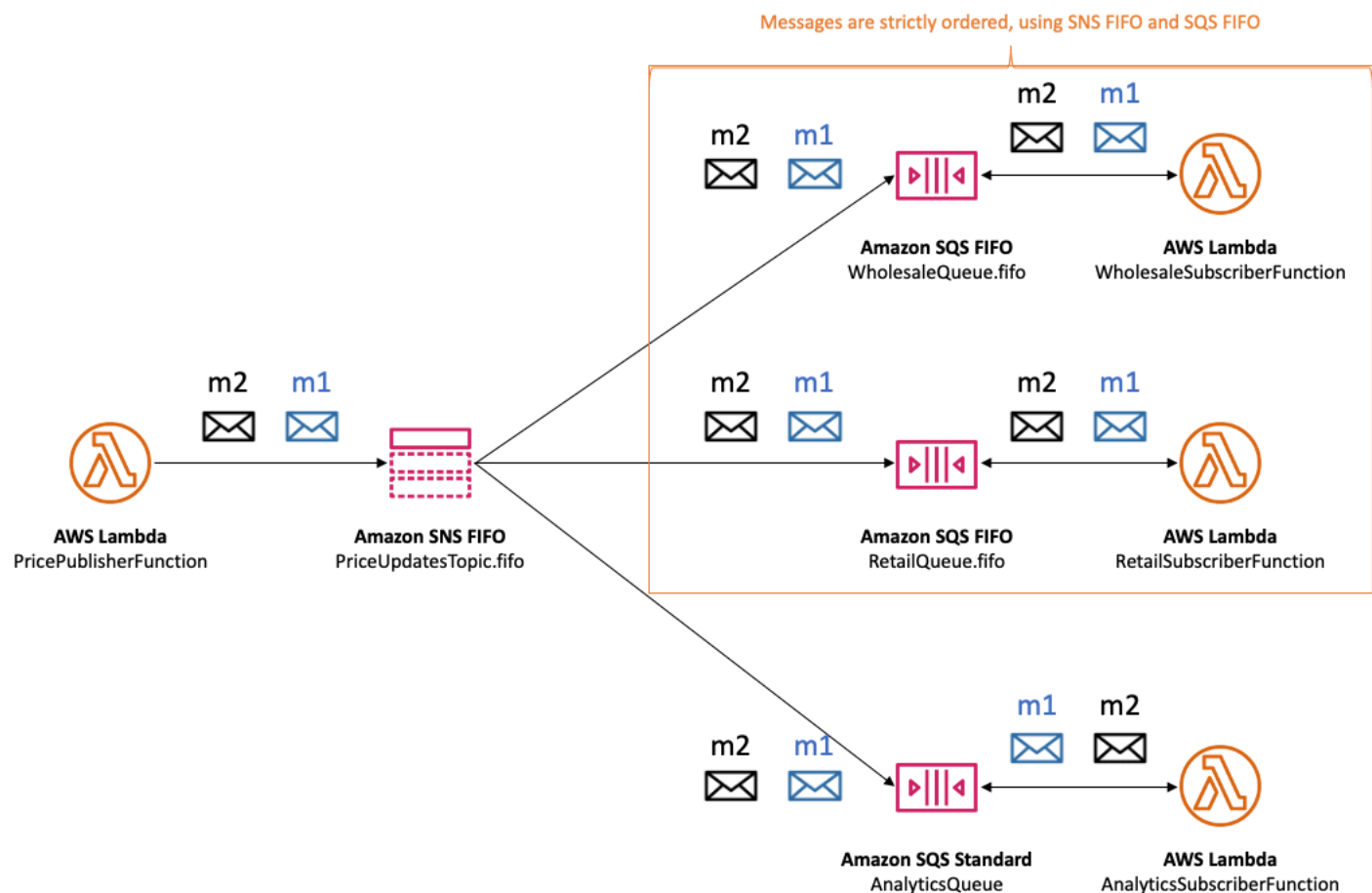
- Un'applicazione di analisi che aggrega gli aggiornamenti dei prezzi e li archivia in un bucket Amazon S3, consentendo ad Amazon Athena di eseguire query sul bucket per scopi di business intelligence (BI). Per ottenere le notifiche di modifica del prezzo, l'applicazione di analisi sottoscrive la coda standard Amazon SQS all'argomento FIFO di Amazon SNS dell'applicazione di gestione dei prezzi. A differenza delle altre applicazioni, quella di analisi non richiede che gli aggiornamenti dei prezzi siano ordinati rigorosamente.



Affinché le applicazioni all'ingrosso e al dettaglio ricevano gli aggiornamenti dei prezzi nell'ordine corretto, l'applicazione di gestione dei prezzi deve utilizzare un sistema di distribuzione dei messaggi rigorosamente ordinato. L'utilizzo di argomenti FIFO di Amazon SNS e code FIFO di Amazon SQS consente l'elaborazione dei messaggi in ordine e senza duplicazione. Per ulteriori informazioni, consulta [Dettagli di ordinazione dei messaggi Amazon SNS per argomenti FIFO](#). Per gli snippet di codice che implementano questo caso d'uso, vedere [Esempi di codice Amazon SNS per argomenti FIFO](#).

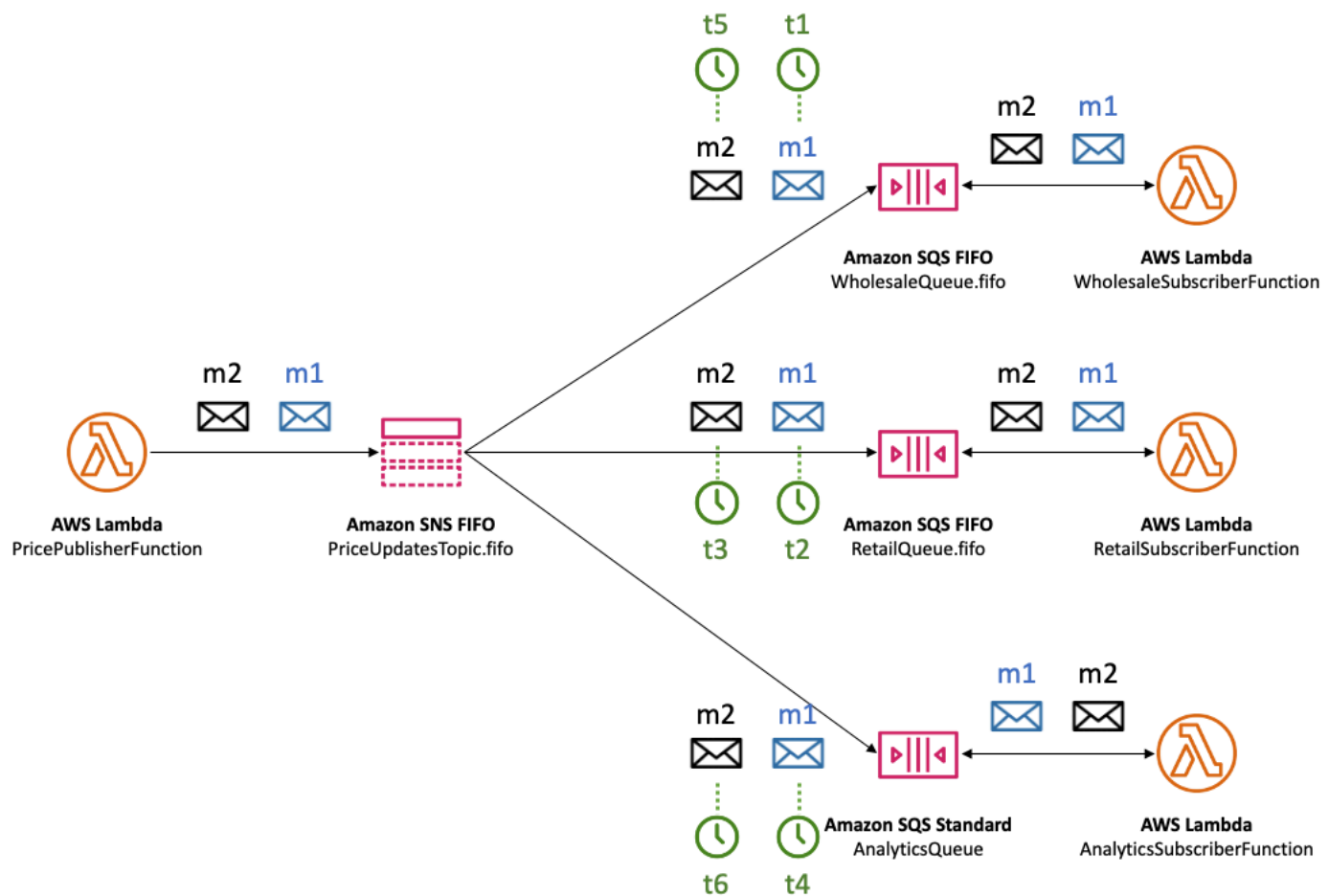
Dettagli di ordinazione dei messaggi Amazon SNS per argomenti FIFO

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

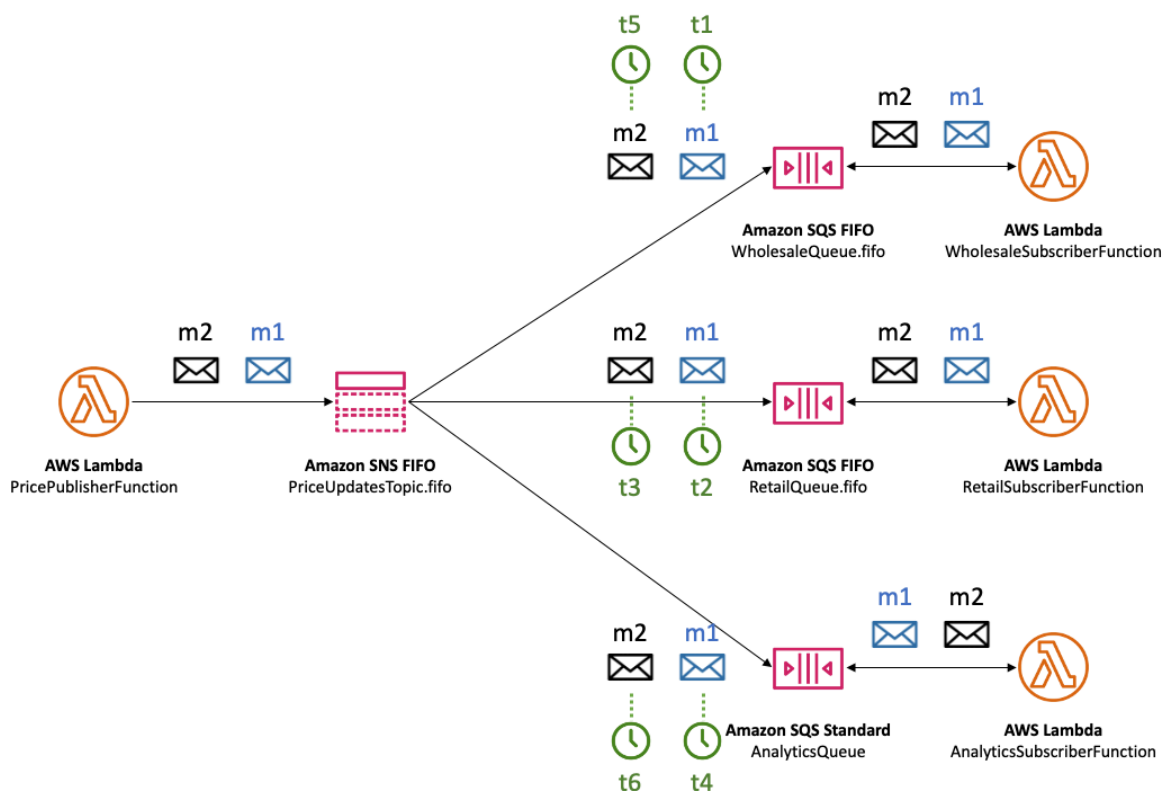


Si noti che non vi è alcun ordinamento implicito degli abbonati. L'esempio seguente ne è un'illustrazione m1 viene consegnato prima all'abbonato all'ingrosso, poi all'abbonato al dettaglio e quindi all'abbonato di analisi. Messaggio m2 viene consegnato prima all'abbonato al dettaglio, poi all'abbonato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

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

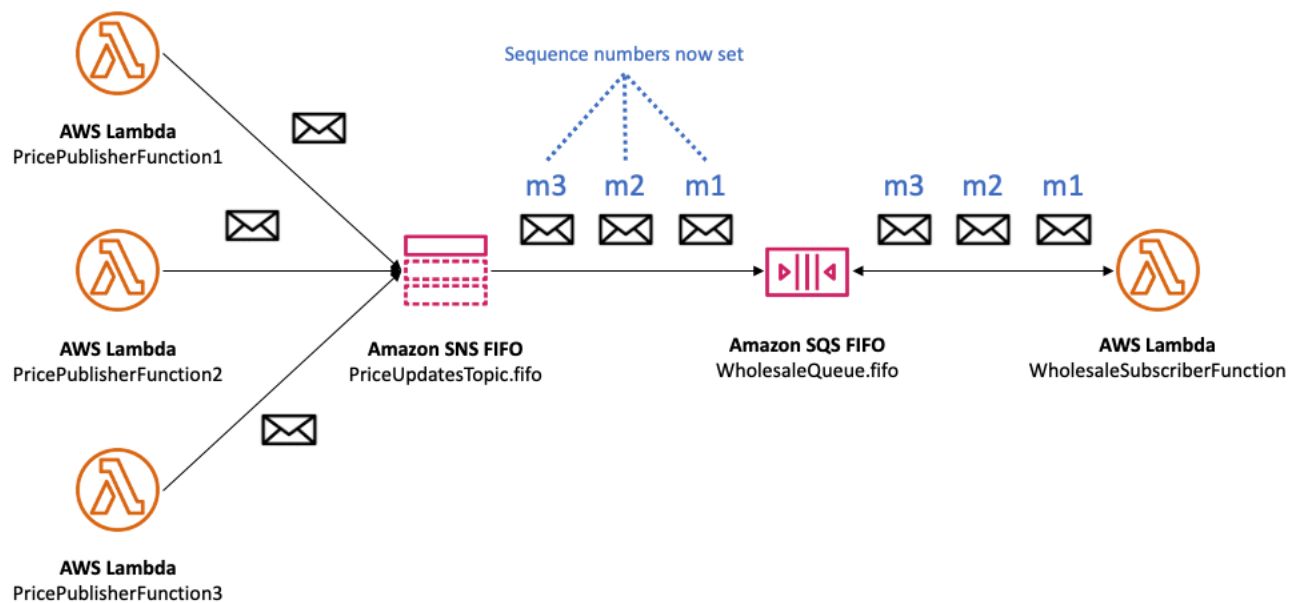


Se un sottoscrittore della coda di Amazon SQS diventa irraggiungibile, può uscire dalla sincronizzazione. Ad esempio, supponiamo che il proprietario della coda dell'applicazione all'ingrosso cambi erroneamente il [Policy Coda Amazon SQS](#) in modo da impedire all'entità servizio Amazon SNS di recapitare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



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

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



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

Raggruppamento di messaggi Amazon SNS per argomenti FIFO

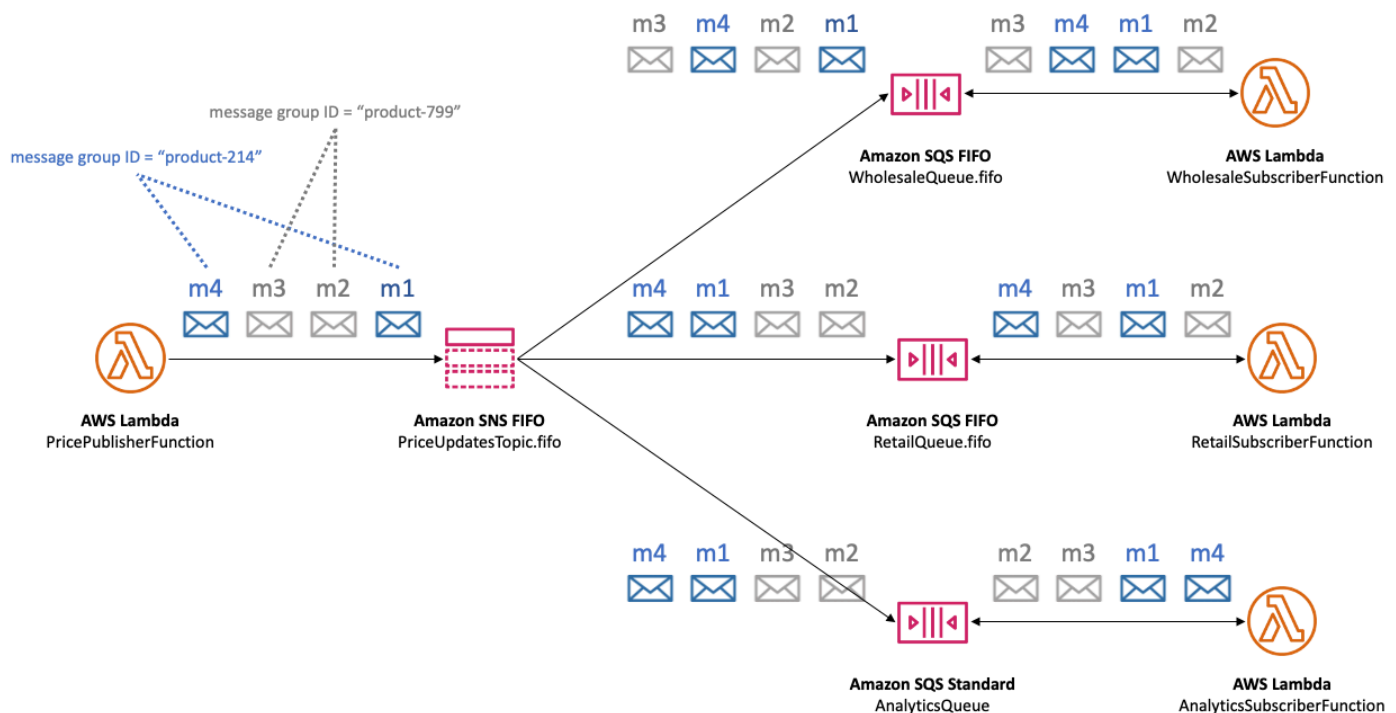
I messaggi che appartengono allo stesso gruppo vengono elaborati uno per uno, in un ordine rigoroso rispetto al gruppo.

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

Non esiste affinità tra un gruppo di messaggi e una sottoscrizione. Pertanto, i messaggi pubblicati in qualsiasi gruppo di messaggi vengono recapitati a tutte le code sottoscritte, in base alle policy di filtro associati alle sottoscrizioni. Per ulteriori informazioni, consultare [Consegna di messaggi Amazon SNS per argomenti FIFO](#) e [Filtraggio dei messaggi Amazon SNS per argomenti FIFO](#).

Nel [caso d'uso esempio di gestione dei prezzi delle parti auto](#), c'è un gruppo di messaggi dedicato per ogni prodotto venduto nella piattaforma. Lo stesso argomento FIFO di Amazon SNS viene utilizzato per l'elaborazione di tutti gli aggiornamenti dei prezzi. La sequenza di aggiornamenti dei prezzi viene mantenuta nel contesto di un singolo prodotto di ricambi auto, ma non 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 m1 viene elaborato prima di m4. Questa sequenza viene mantenuta per tutti i flussi di lavoro che utilizzano FIFO di Amazon SNS in FIFO di Amazon SQS. Allo stesso modo, per il prodotto il cui ID del gruppo di messaggi è product-799, il messaggio m2 viene elaborato prima di m3. Tuttavia, quando si utilizzano le code standard di Amazon SQS, l'ordine dei messaggi non è più garantito e i gruppi di messaggi non esistono. I product-214 e product-799 gruppi di messaggi sono indipendenti l'uno dall'altro, quindi non esiste alcuna relazione tra il modo in cui i messaggi vengono sequenziati.



Distribuzione dei dati per gruppo di messaggi per migliorare le prestazioni IDs

Per ottimizzare la velocità di trasmissione effettiva di consegna, gli argomenti FIFO di Amazon SNS recapitano i messaggi provenienti da diversi gruppi di messaggi in parallelo, mentre l'ordine dei messaggi viene mantenuto rigorosamente all'interno di ciascun gruppo di messaggi. Ogni singolo gruppo di messaggi può recapitare un massimo di 300 messaggi al secondo. Pertanto, per ottenere un throughput elevato per un singolo argomento, utilizzate un gran numero di gruppi IDs di messaggi distinti. Utilizzando un set diversificato di gruppi di messaggi, gli argomenti FIFO di Amazon SNS distribuiscono automaticamente i messaggi su un numero maggiore di partizioni parallele.

 Note

Gli argomenti FIFO di Amazon SNS sono ottimizzati per la distribuzione uniforme dei messaggi tra i gruppi di messaggi IDs, indipendentemente dal numero di gruppi. AWS consiglia di utilizzare un gran numero di gruppi di messaggi distinti IDs per prestazioni ottimizzate.

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

Consegna di messaggi Amazon SNS per argomenti FIFO

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

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

Per i carichi di lavoro che tollerano ordini e at-least-once consegne con il massimo impegno, l'iscrizione alle [code standard di Amazon SQS](#) agli argomenti FIFO di Amazon SNS offre la possibilità di ridurre i costi, oltre a condividere le code tra carichi di lavoro che non utilizzano FIFO.

 Note

Per distribuire i messaggi dagli argomenti FIFO di Amazon SNS alle AWS Lambda funzioni, sono necessari passaggi aggiuntivi. Per prima cosa, effettua la sottoscrizione delle code FIFO o standard di Amazon SQS all'argomento. Quindi configurare le code per attivare le funzioni. Per ulteriori informazioni, vedere [SQS FIFO come origine evento](#) nel AWS Blog Compute.

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

Gli argomenti FIFO SNS supportano le stesse funzionalità di filtraggio dei messaggi degli argomenti standard. Per ulteriori informazioni, consulta [Filtraggio dei messaggi Amazon SNS per argomenti FIFO](#) e il [Semplificazione dei messaggi pub/secondari con il filtro messaggi Amazon SNS](#) post sul AWS Compute blog.

Filtraggio dei messaggi Amazon SNS per argomenti FIFO

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

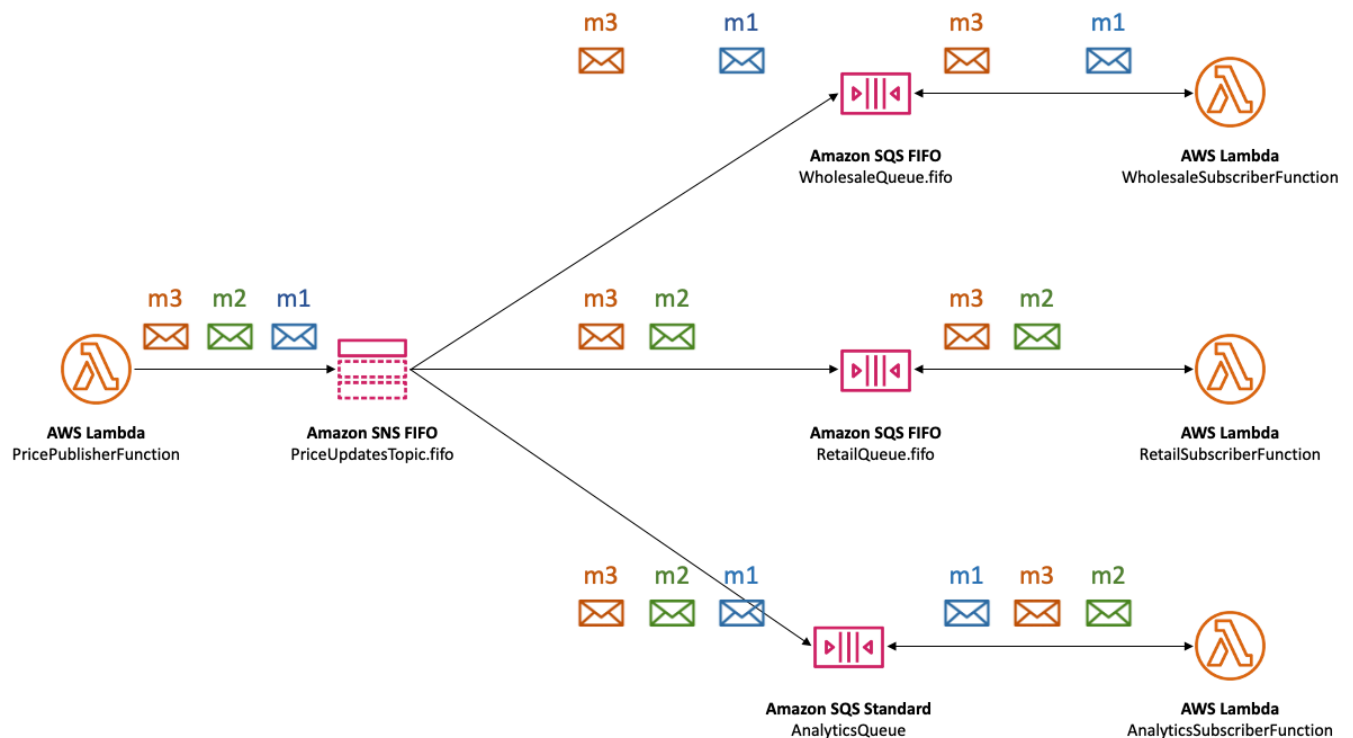
Quando effettui la sottoscrizione di una coda FIFO o standard di Amazon SQS a un argomento FIFO di SNS, puoi utilizzare il filtro dei messaggi per specificare che l'abbonato riceve un sottoinsieme di messaggi, anziché tutti. Ogni sottoscrittore può impostare la propria policy di filtro come attributi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

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

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

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

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



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

Gli argomenti FIFO SNS non recapitano messaggi duplicati agli endpoint sottoscritti. Per ulteriori informazioni, consulta [Deduplicazione dei messaggi Amazon SNS per argomenti FIFO](#).

Deduplicazione dei messaggi Amazon SNS per argomenti FIFO

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

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

Note

La deduplicazione dei messaggi si applica a un intero argomento FIFO di Amazon SNS quando l'attributo `topic` è impostato su `FifoThroughputScope`. Topic [Quando l'attributo `topic FifoThroughputScope` è impostato su `MessageGroup`, la deduplicazione dei messaggi si applica a ogni singolo gruppo di messaggi.](#)

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

Se un messaggio con un particolare ID di deduplicazione viene pubblicato con successo su un argomento Amazon SNS FIFO, qualsiasi messaggio pubblicato con lo stesso ID di deduplicazione, entro l'intervallo di deduplicazione di cinque minuti, viene accettato ma non recapitato. L'argomento Amazon SNS FIFO continua a tenere traccia dell'ID di deduplicazione del messaggio, nell'ambito di deduplicazione configurato dall'attributo `topicFifoThroughputScope`, anche dopo la consegna del messaggio agli endpoint sottoscritti.

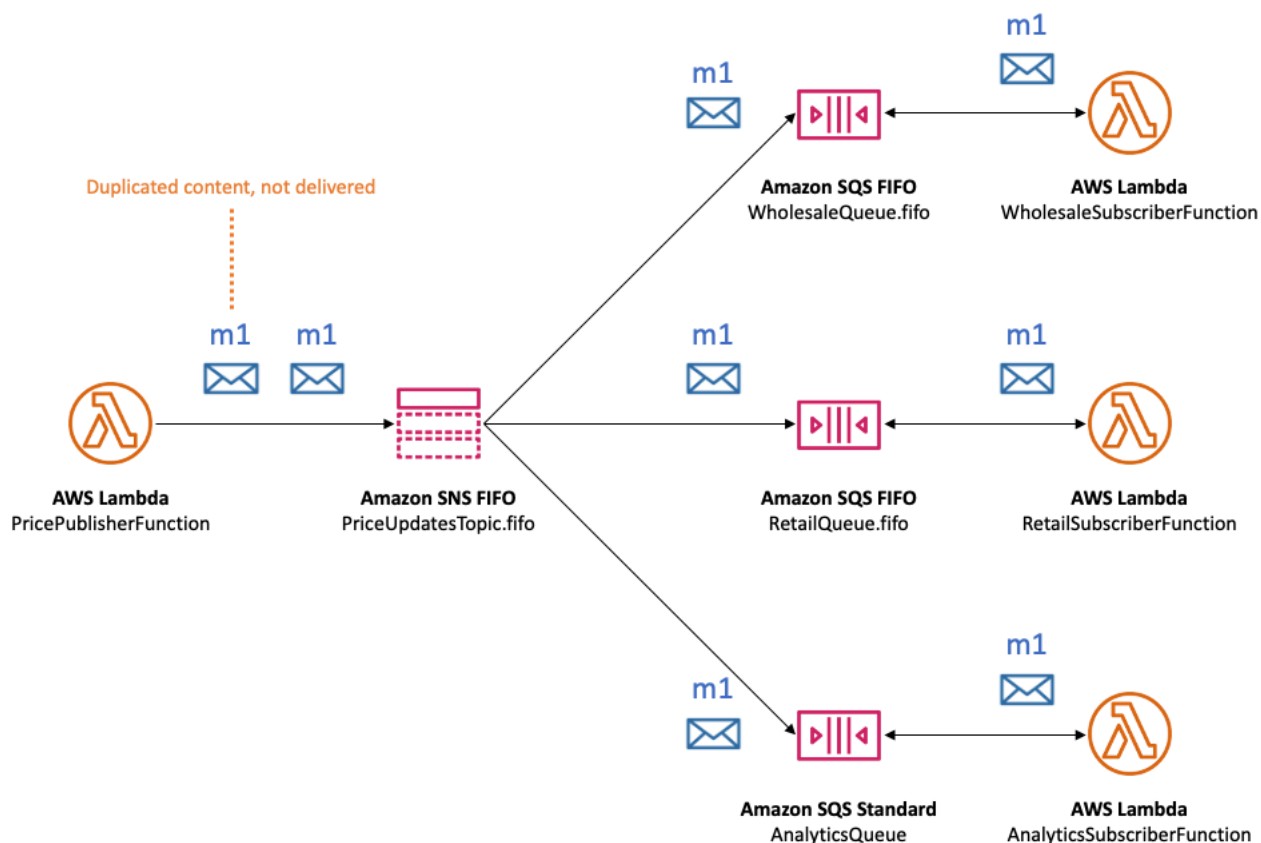
Se è garantito che il corpo del messaggio sia univoco per ogni messaggio pubblicato, puoi abilitare la deduplicazione basata sul contenuto per un argomento FIFO di Amazon SNS e per le code FIFO di Amazon SQS sottoscritte. Amazon SNS utilizza il corpo del messaggio per generare un valore hash univoco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

Note

Gli attributi del messaggio non sono inclusi nel calcolo hash.

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

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



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

Sicurezza dei messaggi Amazon SNS per argomenti FIFO

[Puoi abilitare la crittografia per gli argomenti FIFO di Amazon SNS e le code FIFO di Amazon SQS utilizzando AWS Key Management Service \(\) le chiavi master del cliente \(\)AWS KMS. CMKs](#)

- Puoi creare nuovi argomenti e code FIFO crittografati o abilitare la crittografia per quelli esistenti.
- Viene crittografato solo il corpo del messaggio. Gli attributi dei messaggi, i metadati delle risorse e le metriche delle risorse rimangono non crittografati.

Note

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

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

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

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

Durabilità dei messaggi Amazon SNS per argomenti FIFO

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

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

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

Amazon SNS non riprova a recapitare messaggi non riusciti a causa di errori sul lato client.

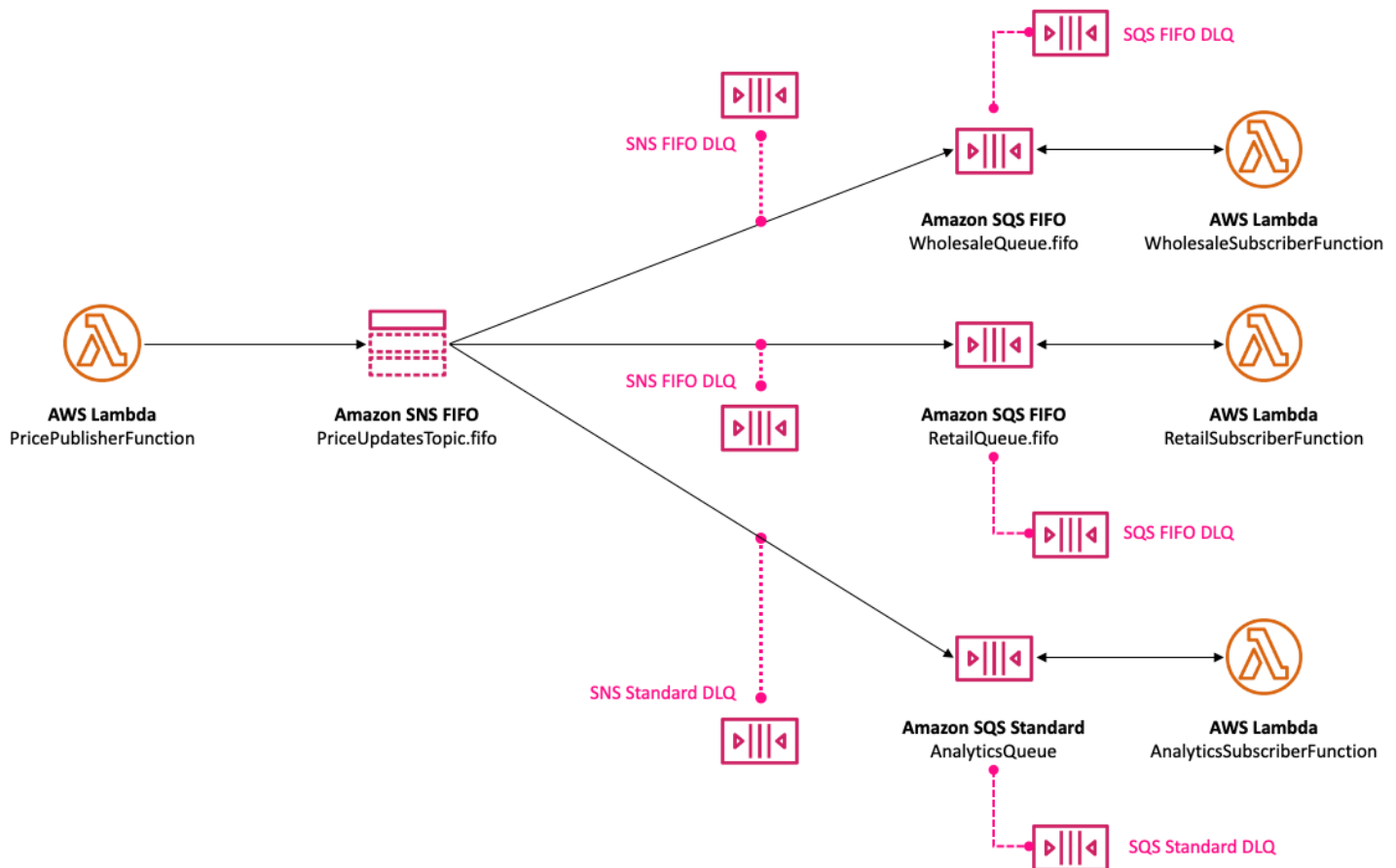
- Errori sul lato server possono verificarsi nelle seguenti situazioni:
 - Il servizio Amazon SQS non è disponibile.
 - Amazon SQS non riesce a elaborare una richiesta valida dal servizio Amazon SNS.

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

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

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

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



La coda DLQ associata a una sottoscrizione Amazon SNS deve essere una coda di Amazon SQS dello stesso tipo della coda di sottoscrizione. Ad esempio, la sottoscrizione FIFO di Amazon SNS per una coda FIFO di Amazon SQS deve avere una coda FIFO di Amazon SQS come coda DLQ. Analogamente, la sottoscrizione FIFO di Amazon SNS per una coda FIFO di Amazon SQS deve avere una coda FIFO di Amazon SQS come coda DLQ. Per ulteriori informazioni, consulta [Code di lettere non ricevute di Amazon SNS](#) il post [Progettazione di app serverless DLQs durevoli con Amazon SNS AWS Lambda, Amazon SQS](#) sul Compute Blog.AWS

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

di un'applicazione esistente. Per ulteriori informazioni, consulta [Archiviazione e riproduzione dei messaggi di Amazon SNS per argomenti FIFO](#).

Archiviazione e riproduzione dei messaggi di Amazon SNS 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 AWS API, SDK e CloudFormation Console di gestione AWS.

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 gli argomenti FIFO di Amazon SNS, tra cui l'impostazione dei periodi di conservazione, il monitoraggio dei messaggi CloudWatch archiviati, l'avvio dei replay tramite gli attributi di 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 messaggi Amazon SNS per i proprietari di argomenti FIFO](#)
- 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 [Replay dei messaggi di Amazon SNS per gli abbonati all'argomento FIFO](#).
- 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 messaggi Amazon SNS per i proprietari di argomenti FIFO

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

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

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

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

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

 Note

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

3. Nella pagina Modifica argomento espandi la sezione Policy di archiviazione.
4. Abilita la funzionalità Policy di archiviazione e inserisci il Numero di giorni per i quali desideri archiviare i messaggi nell'argomento.
5. 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 policy di archiviazione è impostata su un oggetto JSON vuoto.
 - Periodo di conservazione dei messaggi: il numero di giorni specificato per la conservazione dei messaggi.
 - Data di inizio dell'archiviazione: la data a partire dalla quale gli abbonati possono riprodurre i messaggi.
 - Anteprima JSON: l'anteprima JSON della policy di archiviazione.
- (Facoltativo) Per modificare una policy di archiviazione, vai alla pagina di riepilogo dell'argomento e scegli Modifica.
- (Facoltativo) Per disattivare una policy di archiviazione, vai alla pagina di riepilogo dell'argomento e scegli Modifica. Disattiva la policy di archiviazione e scegli Salva modifiche.
- (Facoltativo) Per eliminare un argomento con una policy di archiviazione, è necessario prima disattivare la policy di archiviazione come descritto in precedenza.

⚠ Important

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

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

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

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

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

```
{}
```

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

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta [configShared and credentials files nella AWS SDKs and Tools Reference Guide](#).

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

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

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

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

Crea una politica di archiviazione dei messaggi utilizzando CloudFormation

Per creare una politica di archiviazione, CloudFormation consulta [AWS::SNS::Topic](#) la Guida per l'AWS CloudFormation 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 necessario fornire ad Amazon SNS l'accesso Decrypt alla chiave KMS utilizzata per crittografare i messaggi nell'archivio.

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

Important

Quando aggiungi le nuove sezioni alla policy della chiave KMS, non modificare le sezioni esistenti nella policy. Se la crittografia è abilitata in un argomento e la chiave KMS è

disabilitata o eliminata oppure la policy della chiave KMS non è configurata correttamente per Amazon SNS, Amazon SNS non potrà riprodurre i messaggi ai tuoi abbonati.

Concessione delle autorizzazioni di decrittografia ad Amazon SNS

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

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

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

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, consultare [Registrazione e monitoraggio in Amazon SNS](#).

Metrica	Description
ApproximateNumberOfMessagesArchived	Fornisce al proprietario dell'argomento il numero aggregato di messaggi archiviati nell'archivio degli argomenti, con una risoluzione di 60 minuti.

Metrica	Description
ApproximateNumberOfBytesArchived	Fornisce al proprietario dell'argomento il numero aggregato di byte archiviati in tutti i 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 risoluzione di 1 minuto.
NumberOfBytesArchiveProcessing	Fornisce al proprietario dell'argomento il numero aggregato di messaggi salvati nell'archivio degli argomenti durante l'intervallo con una risoluzione di 1 minuto.

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

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

Replay dei messaggi di Amazon SNS per gli abbonati all'argomento FIFO

Amazon SNS Replay consente agli abbonati agli argomenti di recuperare e reinviare i messaggi archiviati dal Topic Data Store a un endpoint sottoscritto.

- I messaggi possono essere riprodotti immediatamente dopo la creazione dell'abbonamento.
- Un messaggio riprodotto mantiene lo stesso contenuto e `Timestamp` quello originale. `MessageId`
- Il messaggio include un `Replayed` attributo che indica che si tratta di un messaggio riprodotto.

- Per riprodurre solo messaggi specifici, applica una politica 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 il Console di gestione AWS

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

1. Accedi alla [console Amazon SNS](#).
2. Scegli una sottoscrizione all'argomento o creane una nuova. Per ulteriori informazioni sulla creazione di sottoscrizioni, consulta [Creazione di un abbonamento a un argomento di Amazon SNS](#).
3. Per avviare la riproduzione del messaggio, vai al menu a discesa Riproduci e scegli Inizia riproduzione.
4. Dalla finestra modale Intervallo di tempo di riproduzione, effettua le seguenti selezioni:
 - a. Scegli la data e l'ora di inizio della riproduzione: scegli la data (YYYY/MM/DD formato) e l'ora (formato hh:mm:ss 24 ore su 24) 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/DD formato) 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.

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

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

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

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

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

Per riprodurre tutti i messaggi inviati all'argomento a partire dal 1° ottobre 2023 e continuare a ricevere tutti i messaggi appena pubblicati sull'argomento, utilizza l'operazione API `SetSubscriptionAttributes` per impostare una `ReplayPolicy` sull'abbonamento nel modo seguente:

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

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

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

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta [configShared and credentials files nella AWS SDKs and Tools Reference Guide](#).

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

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

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

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

Comprendere il EndingPoint

Quando applichi un `ReplayPolicy` abbonamento Amazon SNS, 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

Il filtraggio dei messaggi di Amazon SNS ti consente di controllare i messaggi riprodotti che Amazon SNS riproduce sull'endpoint dell'abbonato. Quando il filtraggio e l'archiviazione dei messaggi sono entrambi abilitati, Amazon SNS recupera prima il messaggio dall'archivio dati dell'argomento, quindi applica il messaggio alla `FilterPolicy` dell'abbonamento. Il messaggio viene recapitato all'endpoint sottoscritto quando c'è una corrispondenza, altrimenti il messaggio viene filtrato. Per ulteriori informazioni, consulta [Policy di filtro degli abbonamenti Amazon SNS](#).

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, consultare [Registrazione e monitoraggio in Amazon SNS](#).

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

Esempi di codice Amazon SNS per argomenti FIFO

Utilizza i seguenti esempi di codice per integrare il [caso d'uso di esempio di gestione dei prezzi dei ricambi auto](#) con un argomento FIFO di Amazon SNS e una coda FIFO Amazon SQS o una coda standard.

Utilizzo di un SDK AWS

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

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

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

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

Java

SDK per Java 2.x

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

In questo esempio

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

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

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

    public static void main(String[] args) {

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

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

        // For convenience, the QueueData class holds metadata about a queue:
        // ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
```



```
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard));

// Create queues.
createQueues(queues);

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

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

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

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

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

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

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

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

        return topicArn;
    }
}
```

```

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

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

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

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {
    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

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

        Map<String, MessageAttributeValue> attributes = new HashMap<>();

```

```
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()
            .topicArn(topicArn)
            .subject(subject)
            .message(payload)
            .messageGroupId(groupId)
            .messageDeduplicationId(dedupId)
            .messageAttributes(attributes)
            .build();

        final PublishResponse response = snsClient.publish(pubRequest);
        System.out.println(response.messageId());
        System.out.println(response.sequenceNumber());
        System.out.println("Message was published to " + topicArn);

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

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Pubblicare](#)
 - [Subscribe](#)

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

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

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

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
    Attributes={})
```

```
queues.add(analytics_queue)
print(f"Created standard queue with URL: {analytics_queue.url}.")

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

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

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

for q in queues:
    sub = fifo_topic_wrapper.subscribe_queue_to_topic(
        topic, q.attributes["QueueArn"]
    )
    subscriptions.add(sub)

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

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

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

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

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

sns_wrapper.delete_topic(topic)

for q in queues:
    fifo_topic_wrapper.delete_queue(q)

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

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

```

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

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

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
        letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
        characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

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

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

        :param queue: The queue resource.

```

```

:param topic_arn: The ARN of the topic.
:return: None.
"""
try:
    queue.set_attributes(
        Attributes={
            "Policy": json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Sid": "test-sid",
                            "Effect": "Allow",
                            "Principal": {"AWS": "*"},
                            "Action": "SQS:SendMessage",
                            "Resource": queue.attributes["QueueArn"],
                            "Condition": {
                                "ArnLike": {"aws:SourceArn": topic_arn}
                            },
                        }
                    ],
                }
            )
        )
    logger.info("Added trust policy to the queue.")
except ClientError as error:
    logger.exception("Couldn't add trust policy to the queue!")
    raise error

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

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,

```

```

        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error

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

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

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

    :param queue: The queue to delete.

```



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

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CreateTopic](#)
 - [Pubblicare](#)
 - [Subscribe](#)

SAP ABAP

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

```
" Creates a FIFO topic. "
DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmmap_w=>tt_topicattributesmap.
DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmmap_w=>ts_topicattributesmap_maprow.
ls_tpc_attributes-key = 'FifoTopic'.
```

```

ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrmap_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_snssubscriptionlmt00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
        number of subscriptions allowed.' TYPE 'E'.
        ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
        cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
        cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.

```

```

        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
'String'

iv_stringvalue = 'High' ).
    INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

DATA(lo_result) = lo_sns->publish(
    iv_topicarn = lv_topic_arn
    iv_message = 'The price of your mobile plan has been increased from
$19 to $23'
    iv_subject = 'Changes to mobile plan'
    iv_messagegroupid = 'Update-2'
    iv_messagededuplicationid = 'Update-2.1'
    it_messageattributes = lt_msg_attributes ).
    ov_message_id = lo_result->get_messageid( ).
ov_message_id is returned for testing purposes. "
    MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
    CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.

```

- Per informazioni dettagliate sull'API, consulta gli argomenti seguenti nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.
 - [CreateTopic](#)
 - [Pubblicare](#)
 - [Subscribe](#)

Ricezione di messaggi da sottoscrizioni FIFO

Ora puoi ricevere aggiornamenti dei prezzi nelle tre applicazioni sottoscritte. Come illustrato in [the section called “Caso d'uso dell'argomento FIFO”](#), il punto di accesso per ogni applicazione consumer è la coda di Amazon SQS, che la AWS Lambda funzione corrispondente può interrogare automaticamente. Quando una coda Amazon SQS è un'origine di eventi per una funzione Lambda, Lambda ridimensiona il proprio parco istanze di strumenti per il polling in base alle esigenze per consumare i messaggi in modo efficiente.

Per ulteriori informazioni, consulta [Using AWS Lambda with Amazon SQS](#) nella AWS Lambda Developer Guide. Per informazioni sulla creazione di sondaggi di coda personalizzati, consulta la

sezione [Consigli per le code standard e FIFO di Amazon SQS nella Guida per gli sviluppatori di Amazon Simple Queue Service](#) e [ReceiveMessage](#) nel riferimento alle API di Amazon Simple Queue Service.

Usando AWS CloudFormation

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

Note

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

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "PriceUpdatesTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "TopicName": "PriceUpdatesTopic.fifo",
        "FifoTopic": true,
        "ContentBasedDeduplication": false,
        "ArchivePolicy": {
          "MessageRetentionPeriod": "30"
```

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

```

    }
  }
},
"RetailSubscription": {
  "Type": "AWS::SNS::Subscription",
  "Properties": {
    "TopicArn": {
      "Ref": "PriceUpdatesTopic"
    },
    "Endpoint": {
      "Fn::GetAtt": [
        "RetailQueue",
        "Arn"
      ]
    },
    "Protocol": "sqs",
    "RawMessageDelivery": "false",
    "FilterPolicyScope": "MessageBody",
    "FilterPolicy": {
      "business": [
        "retail"
      ]
    }
  }
},
"AnalyticsSubscription": {
  "Type": "AWS::SNS::Subscription",
  "Properties": {
    "TopicArn": {
      "Ref": "PriceUpdatesTopic"
    },
    "Endpoint": {
      "Fn::GetAtt": [
        "AnalyticsQueue",
        "Arn"
      ]
    },
    "Protocol": "sqs",
    "RawMessageDelivery": "false"
  }
},
"SalesQueuesPolicy": {
  "Type": "AWS::SQS::QueuePolicy",
  "Properties": {

```

```
"PolicyDocument": {
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": [
        "sqs:SendMessage"
      ],
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": {
            "Ref": "PriceUpdatesTopic"
          }
        }
      }
    }
  ],
},
"Queues": [
  {
    "Ref": "WholesaleQueue"
  },
  {
    "Ref": "RetailQueue"
  },
  {
    "Ref": "AnalyticsQueue"
  }
]
}
}
```

Per ulteriori informazioni sulla distribuzione AWS delle risorse utilizzando un CloudFormation modello, consulta la Guida [introduttiva](#) nella Guida per l'CloudFormation utente.

Filtraggio messaggi di Amazon SNS

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

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

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

Per ulteriori informazioni, consulta [Filtro dei messaggi pubblicati negli argomenti](#).

Ambito delle policy di filtro per le sottoscrizioni Amazon SNS

L'attributo `FilterPolicyScope` subscription consente di definire l'ambito di filtraggio impostando uno dei seguenti valori:

- `MessageAttributes`— Applica la politica di filtro agli attributi dei messaggi (impostazione predefinita).
- `MessageBody`— Applica la politica di filtro al corpo del messaggio.

Note

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

Policy di filtro degli abbonamenti Amazon SNS

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

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

Important

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

Una sottoscrizione accetta un messaggio nelle seguenti condizioni:

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

Attualmente Amazon RDS supporta i seguenti operatori di filtro:

- [Logica AND](#)
- [Logica OR](#)
- [Operatore OR](#)
- [Corrispondenza di chiave](#)
- [Corrispondenza esatta dei valori numerici](#)
- [Corrispondenza Anything-but dei valori numerici](#)
- [Corrispondenza intervallo dei valori numerici](#)

- [Corrispondenza esatta dei valori di stringa](#)
- [Corrispondenza Anything-but delle stringhe](#)
- [Corrispondenza di stringhe utilizzando un prefisso con l'operatore anything-but](#)
- [equals-ignore case dei valori di stringa](#)
- [Corrispondenza dell'indirizzo IP dei valori di stringa](#)
- [Corrispondenza del prefisso dei valori di stringa](#)
- [Corrispondenza del suffisso dei valori di stringa](#)

Esempi di politiche di filtro di Amazon SNS

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

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

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

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

```
{
  "Type": "Notification",
  "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
  "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
  "Message": "message-body-with-transaction-details",
  "Timestamp": "2019-11-03T23:28:01.631Z",
  "SignatureVersion": "4",
  "Signature": "signature",
  "UnsubscribeURL": "unsubscribe-url",
  "MessageAttributes": {
```

```

    "customer_interests": {
      "Type": "String.Array",
      "Value": "[\"soccer\", \"rugby\", \"hockey\"]"
    },
    "store": {
      "Type": "String",
      "Value": "example_corp"
    },
    "event": {
      "Type": "String",
      "Value": "order_placed"
    },
    "price_usd": {
      "Type": "Number",
      "Value": "210.75"
    }
  }
}

```

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

```

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

```

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

Policy che accetta il messaggio di esempio

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

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

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

Policy che rifiuta il messaggio di esempio

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

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

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

Filtra i vincoli delle policy in Amazon SNS

Quando configuri le politiche di filtro in Amazon SNS, ci sono alcune regole importanti da tenere a mente. Queste regole aiutano a garantire l'applicazione efficace delle politiche di filtro mantenendo al contempo le prestazioni e la compatibilità del sistema.

Vincoli comuni delle policy

Quando configuri le politiche di filtro in Amazon SNS, segui queste importanti regole per assicurarti che funzionino in modo efficace mantenendo le prestazioni e la compatibilità del sistema:

- 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: la combinazione totale di valori in una politica di filtro non deve superare 150. Per calcolare la combinazione totale, moltiplica il numero di valori in ogni array nella politica di filtro.
- Limita il numero di chiavi: una politica di filtro può avere un massimo di cinque chiavi.

Considerazioni aggiuntive

- Il JSON della policy di filtro può contenere quanto segue:
 - Stringhe tra virgolette
 - Numeri
 - Le parole chiave `true`, `false` e `null` senza virgolette
- Quando usi l'API Amazon SNS, devi passare il codice JSON della policy di filtro come stringa UTF-8 valida.
- La dimensione massima di una policy 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 impedirà la creazione di abbonamenti alla coda Amazon SQS con l'API [Subscribe](#). Tuttavia, l'operazione avrebbe esito negativo quando si collega la policy di filtro alla chiamata API [Subscribe](#) (o alla chiamata API [SetSubscriptionAttributes](#)).

Per richiedere un aumento della quota, è possibile utilizzare [AWS Service Quotas](#).

Vincoli delle policy per il filtro basato sugli attributi

Il filtro basato sugli attributi è l'opzione predefinita. [FilterPolicyScope](#) è impostato su `MessageAttributes` nella sottoscrizione.

- Amazon SNS non accetta una policy di filtri annidati per il filtro basato su attributi.
- Amazon SNS confronta le proprietà della policy solo con gli attributi di messaggio che hanno i seguenti tipi di dati:
 - `String`
 - `String.Array`

Important

Quando si utilizza il filtro basato sugli attributi in Amazon SNS, è necessario eseguire il doppio escape di determinati caratteri speciali, in particolare:

- Virgolette doppie («)
- Barre rovesciate ()

Se non si esegue il double escape di questi caratteri, la politica di filtro non corrisponderà agli attributi di un messaggio pubblicato e la notifica non verrà recapitata.

Considerazioni aggiuntive

- Il passaggio di oggetti negli array non è consigliato perché 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` è supportato per i valori degli attributi numerici.
- Amazon SNS ignora gli attributi dei messaggi con il tipo di dati binario.

Esempio di politica di complessità:

Nel seguente esempio di policy, la prima chiave ha tre operatori di corrispondenza, la seconda ha un operatore di corrispondenza e la terza ha due operatori di corrispondenza.

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

La combinazione totale viene calcolata come il prodotto del numero di operatori di corrispondenza per ogni chiave nella politica di filtro:

```
3(match operators of key_a)
x 1(match operators of key_b)
x 2(match operators of key_c)
= 6
```

Vincoli delle policy per il filtro basato su payload

Per passare dal filtro basato sugli attributi (predefinito) al filtro basato sul payload, è necessario impostare [FilterPolicyScope](#) su `MessageBody` nella sottoscrizione.

- Amazon SNS accetta una policy di filtri annidati per il filtro basato su payload.
- Per una politica annidata, solo le chiavi a foglia vengono conteggiate per il limite di cinque chiavi.

Esempio di politica per il limite delle chiavi:

Nel seguente esempio di politica:

- Ci sono due tasti a forma di foglia: `key_c` e `key_e`.
- `key_c` ha quattro operatori di corrispondenza con un livello nidificato di tre e `key_e` dispone di tre operatori di corrispondenza con un livello nidificato di due.

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

```

    }
  },
  "key_d": {
    "key_e": ["value_one", "value_two", "value_three"]
  }
}

```

La combinazione totale viene calcolata come il prodotto del numero di operatori di corrispondenza e del livello annidato per ciascuna chiave nella politica di filtro:

```

4(match operators of key_c)
x 3(nested level of key_c)
x 3(match operators of key_e)
x 2(nested level of key_e)
= 72

```

Linee guida per l'utilizzo del pattern Wildcard

Amazon SQS implementa protezioni quando registri una policy di filtro contenente wildcard per garantire che non vengano create policy di filtro troppo complesse, poiché ciò influirebbe sulle prestazioni dell'applicazione.

Struttura del pattern

I campi contengono uno o più motivi. L'esempio seguente mostra un campo che utilizza due modelli:

```

{
  "greeting": [
    {"anything-but": {"prefix": "Hello"}},
    {"wildcard": "H*"}
  ] // 2 patterns
}

```

Regole di complessità

- La complessità totale dei caratteri jolly in tutti i campi non deve superare i 100 punti
- Massimo 3 caratteri jolly per pattern

Calcolo della complessità

- Complessità del campo = (Sum of pattern points) × (Number of patterns)

- Punti del modello:

Wild card singola: 1 punto

jolly multipli: 3 punti ciascuno

Tutto tranne: 1 punto

Di seguito è riportato un esempio di calcolo della complessità:

```
{
  "filename": [
    {"wildcard": "*.txt"},      // 1 point
    {"wildcard": "log*"}      // 1 point
  ]                          // Total: (1 + 1) × 2 = 4 points
}
```

Logica AND/OR

Usa AND/OR la logica nelle politiche di filtro per abbinare gli attributi o le proprietà del corpo del messaggio in Amazon SNS. Ciò consente un filtraggio dei messaggi più preciso e flessibile.

Logica AND

Puoi applicare l'operatore logico AND utilizzando più nomi di proprietà.

Esaminiamo la seguente policy:

```
{
  "customer_interests": ["rugby"],
  "price_usd": [{"numeric": [ ">", 100 ]}]
}
```

Corrisponde a qualsiasi attributo di messaggio o corpo del messaggio che abbia il valore di `customer_interests` impostato su `rugby` e il valore di `price_usd` impostato su un numero superiore a 100.

Note

Non è possibile applicare la logica AND ai valori dello stesso attributo.

Logica OR

Puoi applicare l'operatore logico OR assegnando più valori a un nome di proprietà.

Esaminiamo la seguente policy:

```
{
  "customer_interests": ["rugby", "football", "baseball"]
}
```

Corrisponde a qualsiasi attributo di messaggio o proprietà del corpo di messaggio con il valore di `customer_interests` impostato su `rugby`, `football` o `baseball`.

Operatore OR

È possibile utilizzare l'operatore `"$or"` per definire in modo esplicito una policy di filtro per esprimere la relazione OR tra più attributi della policy.

Amazon SNS riconosce una relazione `"$or"` solo quando la policy soddisfa tutte le seguenti condizioni. Se tutte queste condizioni non sono soddisfatte, `"$or"` viene considerato come un normale nome di attributo, come qualsiasi altra stringa della policy.

- Nella regola seguita da un array è presente un attributo di campo `"$or"`, ad esempio, `"$or" : []`.
- Ci sono almeno 2 oggetti nell'array `"$or"`: `"$or": [{}, {}]`.
- Nessuno degli oggetti nell'array `"$or"` ha nomi di campo che sono parole chiave riservate.

Altrimenti `"$or"` viene considerato come un normale nome di attributo, come le altre stringhe della policy.

La seguente policy non viene analizzata come una relazione OR perché `numeric` e `prefix` sono parole chiave riservate.

```
{
  "$or": [ {"numeric" : 123}, {"prefix": "abc"} ]
}
```

Esempi di operatori OR

OR 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"}
```

or

```
"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"
}
```

or

```
{
  "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

Utilizza l'`exists` operatore in una politica di filtro per abbinare i messaggi in arrivo a seconda che una proprietà specifica sia presente o assente.

- `exists` funziona solo sui nodi foglia (attributi finali della struttura).
- Non si applica ai nodi intermedi all'interno di una struttura JSON annidata.
- 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

Filtra i messaggi abbinando i valori numerici ai valori degli attributi del messaggio o ai valori delle proprietà del corpo del messaggio. Nella policy JSON, i valori numerici non sono racchiusi tra virgolette doppie. È possibile disporre delle seguenti operazioni numeriche per il filtro.

 Note

I prefissi sono supportati solo per la corrispondenza di stringa.

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

Filtra i messaggi abbinando i valori delle stringhe ai valori degli attributi del messaggio o ai valori delle proprietà del corpo del messaggio. Nella policy JSON, i valori di stringa sono racchiusi tra virgolette doppie. È possibile utilizzare le seguenti operazioni sulle stringhe per abbinare gli attributi o le proprietà del corpo del messaggio:

Corrispondenza esatta

La corrispondenza esatta si verifica quando un valore di proprietà della policy corrisponde a uno o più valori di attributo del messaggio. Per gli attributi `String.Array` di tipo, ogni elemento dell'array viene trattato come una stringa separata per scopi di corrispondenza.

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"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"tennis\"]"}
```

Corrisponde anche ai seguenti corpi dei messaggi:

```
{  
  "customer_interests": "rugby"  
}
```

```
{  
  "customer_interests": "tennis"  
}
```

Tuttavia, non corrisponde ai seguenti attributi del messaggio:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String.Array", "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`. Per gli attributi `String.Array` di tipo, corrisponde se nessuno degli elementi dell'array è elencato nella proprietà policy.

Esaminiamo la seguente proprietà della policy:

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

Corrisponde a uno dei seguenti attributi del messaggio:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

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 ai seguenti attributi del messaggio:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\"]"}
```

Né corrisponde al seguente corpo del messaggio:

```
{
  "customer_interests": ["rugby"]
}
```

Utilizzo di un prefisso con **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"
}
```

tutt'altro che jolly

La seguente proprietà della policy nega il carattere jolly: `*ball`

```
"customer_interests" : [{ "anything-but": { "wildcard": "*ball" } }]
```

Corrisponde ai seguenti attributi:

```
{"customer_interests": ["hockey", "rugby", "soccer"] }
```

Tuttavia, non corrisponde al seguente attributo del messaggio:

```
{"customer_interests": ["baseball", "basketball"] }
```

tutto tranne il suffisso

La seguente proprietà politica nega il `-ball`

suffisso:

```
"customer_interests": [ { "anything-but": { "suffix": "ball" } } ]
```

Corrisponde ai seguenti attributi:

```
{"customer_interests": ["hockey", "rugby", "soccer"] }
```

Tuttavia, non corrisponde al seguente attributo del messaggio:

```
{"customer_interests": ["baseball", "basketball"] }
```

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 messaggi 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"
}
```

Carattere jolly

È possibile utilizzare il carattere jolly (*) per la corrispondenza con valori di stringa in modelli di eventi.

La seguente politica utilizza il carattere jolly (*):

```
"customer_interests": [ { "wildcard": "*ball" } ]
```

Corrisponde ai seguenti attributi:

```
{"customer_interests": ["baseball", "basketball"] }
```

Applicazione di una politica di filtro degli abbonamenti in Amazon SNS

Il filtraggio dei messaggi in Amazon 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 policy di filtro a una sottoscrizione Amazon SNS utilizzando la console di Amazon SNS. Oppure, per applicare le policy a livello di codice, puoi utilizzare l'API Amazon SNS, AWS Command Line Interface il AWS CLI() o AWS qualsiasi SDK che supporti Amazon SNS. Puoi anche usare. AWS CloudFormation

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, la consegna non elaborata dei messaggi non è direttamente correlata 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 sintassi JSON. 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 IAM e Amazon SNS utilizzano un modello di calcolo distribuito chiamato eventuale consistenza. Le aggiunte o le modifiche a una policy di filtro sottoscrizione richiedono fino a 15 minuti per essere pienamente effettive.

Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, scegli Sottoscrizioni.
3. Seleziona una sottoscrizione e quindi scegli Edit (Modifica).
4. Nella pagina Edit (Modifica), espandi la sezione Policy di filtro per sottoscrizione.
5. Scegli tra il filtro basato sugli attributi o sul payload.
6. Nel campo Editor JSON, specifica il corpo JSON della policy di filtro.
7. Seleziona Save changes (Salva modifiche).

Amazon SNS applica la policy di filtro alla sottoscrizione.

AWS CLI

Per applicare una politica di filtro con AWS Command Line Interface (AWS CLI), usa il [set-subscription-attributes](#) comando, come mostrato nell'esempio seguente. Per l'opzione `--attribute-name` specifica `FilterPolicy`. Per `--attribute-value`, specifica la policy JSON.

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

Per specificare codice JSON valido per la policy, racchiudi i nomi e i valori degli attributi tra virgolette doppie. Devi inoltre racchiudere l'intero argomento della policy tra virgolette. Per evitare l'escape delle virgolette, puoi utilizzare virgolette singole per racchiudere la policy e virgolette doppie per racchiudere i nomi e i valori JSON, come mostrato nell'esempio qui sopra.

Se desideri passare dal filtraggio dei messaggi basato sugli attributi (impostazione predefinita) a quello basato sul payload, puoi utilizzare anche il comando. [set-subscription-attributes](#) Per l'opzione `--attribute-name` specifica `FilterPolicyScope`. Per `--attribute-value`, specificare `MessageBody`.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-  
name FilterPolicyScope --attribute-value MessageBody
```

Per verificare l'applicazione della policy di filtro, usa il comando `get-subscription-attributes`. Gli attributi nell'output su terminale devono mostrare la policy di filtro per la chiave `FilterPolicy`, come mostrato nell'esempio seguente:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...  
{  
  "Attributes": {  
    "Endpoint": "endpoint . . .",  
    "Protocol": "https",  
    "RawMessageDelivery": "false",  
    "EffectiveDeliveryPolicy": "delivery policy . . .",  
    "ConfirmationWasAuthenticated": "true",  
    "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed  
\"]}",  
    "FilterPolicyScope": "MessageAttributes",  
    "Owner": "111122223333",  
    "SubscriptionArn": "arn:aws:sns: . . .",
```

```

    "TopicArn": "arn:aws:sns: . . ."
  }
}

```

AWS SDKs

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

Important

Se si utilizza l'esempio SDK for Java 2.x, la classe `SNSMessageFilterPolicy` non è pronta all'uso. Per istruzioni su come installare questa classe, consulta l'[esempio](#) tratto dal GitHub sito Web.

CLI

AWS CLI

Impostazione degli attributi della sottoscrizione

Nell'esempio `set-subscription-attributes` seguente viene impostato l'attributo `RawMessageDelivery` su una sottoscrizione SQS.

```

aws sns set-subscription-attributes \
  --subscription-arn arn:aws:sns:us-east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
  --attribute-name RawMessageDelivery \
  --attribute-value true

```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene impostato un attributo `FilterPolicy` su una sottoscrizione SQS.

```

aws sns set-subscription-attributes \
  --subscription-arn arn:aws:sns:us-east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
  --attribute-name FilterPolicy \
  --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"

```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene rimosso l'attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \  
  --subscription-arn arn:aws:sns:us-  
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \  
  --attribute-name FilterPolicy \  
  --attribute-value "{}"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consultate [SetSubscriptionAttributes AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.SnsException;  
import java.util.ArrayList;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class UseMessageFilterPolicy {  
    public static void main(String[] args) {  
        final String usage = ""
```

```
Usage:    <subscriptionArn>

Where:
    subscriptionArn - The ARN of a subscription.

""";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String subscriptionArn = args[0];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

usePolicy(snsClient, subscriptionArn);
snsClient.close();
}

public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);
    }
}
```

```

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

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

```

- Per i dettagli sull'API, consulta la [SetSubscriptionAttributes](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

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

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

    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """

```



```

    Adds a filter policy to a subscription. A filter policy is a key and a
    list of values that are allowed. When a message is published, it must
    have an
        attribute that passes the filter or it will not be sent to the
    subscription.

    :param subscription: The subscription the filter policy is attached to.
    :param attributes: A dictionary of key-value pairs that define the
    filter.
    """
    try:
        att_policy = {key: [value] for key, value in attributes.items()}
        subscription.set_attributes(
            AttributeName="FilterPolicy",
            AttributeValue=json.dumps(att_policy)
        )
        logger.info("Added filter to subscription %s.", subscription.arn)
    except ClientError:
        logger.exception(
            "Couldn't add filter to subscription %s.", subscription.arn
        )
        raise

```

- Per i dettagli sull'API, consulta [SetSubscriptionAttributes AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per 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->setsubscriptionattributes(
        iv_subscriptionarn = iv_subscription_arn

```

```
        iv_attributename  = 'FilterPolicy'
        iv_attributevalue = iv_filter_policy ).
    MESSAGE 'Added filter policy to subscription.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Subscription does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, [SetSubscriptionAttributes](#) consulta AWS SDK for SAP ABAP API reference.

API Amazon SNS

Per applicare una policy di filtro con l'API di Amazon SNS, effettua una richiesta all'operazione [SetSubscriptionAttributes](#). Imposta il parametro `AttributeName` su `FilterPolicy` e il parametro `AttributeValue` sul JSON della policy di filtro.

Se desideri passare dal filtro dei messaggi basato sugli attributi (opzione predefinita) a quello basato sul payload, puoi utilizzare anche l'azione [SetSubscriptionAttributes](#). Imposta il parametro `AttributeName` su `FilterPolicyScope` e il parametro `AttributeValue` su `MessageBody`.

AWS CloudFormation

Per applicare una politica di filtro utilizzando CloudFormation, utilizza un modello JSON o YAML per creare uno stack. CloudFormation [Per ulteriori informazioni, consultate la FilterPolicy proprietà della AWS::SNS::Subscription risorsa nella Guida per l'AWS CloudFormation utente e il modello di esempio. CloudFormation](#)

1. Accedi alla [console CloudFormation](#).
2. Scegli Create Stack (Crea stack).
3. Nella pagina Select Template (Scegli modello), scegli Upload a template to Amazon S3 (Carica un modello in Amazon S3), scegli il file, quindi scegli Next (Avanti).
4. Nella pagina Specify Details (Specifica dettagli), procedi come segue:
 - a. Per Nome stack, digita `MyFilterPolicyStack`.
 - b. Per `myHttpEndpoint`, digita l'endpoint HTTP a cui iscriverti al tuo argomento.

 Tip

Se non disponi di un endpoint HTTP, creane uno.

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

Rimozione di una politica di filtro degli abbonamenti in Amazon SNS

Per interrompere il filtraggio dei messaggi inviati a una sottoscrizione, rimuovi la policy di filtro delle sottoscrizioni sovrascrivendola con un corpo JSON vuoto. Dopo aver rimosso la policy, la sottoscrizione accetta ogni messaggio pubblicato.

Utilizzando il Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, scegli Sottoscrizioni.
3. Seleziona una sottoscrizione e quindi scegli Edit (Modifica).
4. Nella **EXAMPLE1-23bc-4567-d890-ef12g3hij456** pagina Modifica, espandi la sezione Politica di filtro degli abbonamenti.
5. Nel campo JSON editor (Editor JSON) specificare un corpo JSON vuoto per la policy di filtro. {}
6. Seleziona Save changes (Salva modifiche).

Amazon SNS applica la policy di filtro alla sottoscrizione.

Utilizzando il AWS CLI

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

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

Utilizzo dell'API Amazon SNS

Per rimuovere una policy di filtro con l'API di Amazon SNS, effettua una richiesta all'operazione [SetSubscriptionAttributes](#). Imposta il parametro `AttributeName` su `FilterPolicy` e fornisci un corpo JSON vuoto per il parametro `AttributeValue`.

Protezione dei dati dei messaggi in Amazon SNS

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Cos'è la protezione dei dati dei messaggi?

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

La protezione dei dati dei messaggi esegue la scansione dei dati in movimento alla ricerca di informazioni di identificazione personale (PII) e di dati sanitari protetti (PHI) tramite identificatori di dati. Puoi scegliere di utilizzare identificatori di dati [predefiniti](#) (o gestiti da Amazon SNS), ad esempio nomi, indirizzi, numeri di carte di credito e codici di farmaci soggetti a prescrizione medica, oppure puoi creare identificatori di dati [personalizzati](#), specifici per il tuo caso d'uso aziendale. Utilizzando le informazioni scansionate, la protezione dei dati dei messaggi fornisce registri di controllo dettagliati e consente di eseguire operazioni specifiche volte a proteggere tali dati.

La protezione dei dati dei messaggi supporta le seguenti operazioni per proteggere le informazioni sensibili dei clienti:

- [Audit](#) (Verifica): controlla fino al 99% dei dati pubblicati in un argomento Amazon SNS. Puoi quindi scegliere di inviare i risultati ad [Amazon CloudWatch](#), [Amazon S3](#) o <https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html>
- [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 standard di Amazon SNS.

Perché utilizzare la protezione dei dati dei messaggi?

L'introduzione della protezione dei dati dei messaggi nei programmi di governance, gestione del rischio e conformità ti consente di implementare policy di protezione che ti aiutano a individuare e prevenire la fuga di dati. Ciò fornisce ai team strumenti che possono aiutare a ridurre i rischi finanziari, legali e normativi rispettando le normative sulla privacy quali, ad esempio, HIPAA, GDPR, PCI e FedRAMP. Inoltre, libera gli sviluppatori dal sovraccarico operativo associato alla creazione e alla gestione di strumenti specifici per la protezione dei dati sensibili.

Ad esempio, puoi utilizzare la protezione dei dati dei messaggi per creare una policy di verifica per determinare se uno dei tuoi sistemi invia o riceve inavvertitamente dati sensibili. Se i risultati della verifica mostrano che i sistemi inviano i dati relativi alle carte di credito a sistemi che non li richiedono, puoi utilizzare una policy di blocco per impedire che ciò accada.

 Note

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

Modifica della disponibilità della protezione dei dati dei messaggi Amazon SNS

 Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Dopo un'attenta valutazione, la funzionalità di protezione dei dati dei messaggi di Amazon SNS non sarà più disponibile per i nuovi clienti a partire dal 30 aprile 2026. Se sei un cliente esistente con le politiche di protezione dei dati dei messaggi SNS configurate, puoi continuare a utilizzare la funzionalità all'interno di tali account. Sebbene non introdurremo miglioramenti alla funzionalità, restiamo impegnati a fornire aggiornamenti di sicurezza.

Architettura alternativa

Un'architettura AWS Lambda basata su Amazon Bedrock Guardrails è l'approccio consigliato per i clienti che cercano una soluzione alternativa. Questa soluzione consente il rilevamento e la protezione dei dati sensibili in tempo reale con la flessibilità necessaria per personalizzare la protezione dei dati per soddisfare requisiti specifici.

Un esempio che dimostra questa architettura consigliata è disponibile nel repository AWS Samples su GitHub: [Proteggi i dati sensibili nei messaggi SNS usando Amazon Bedrock Guardrails](#). L'esempio mostra come sfruttare Amazon Bedrock Guardrails e il pattern matching personalizzato per il rilevamento di dati sensibili.

Panoramica dell'architettura

L'architettura consigliata basata su Lambda funziona come segue:

1. Gli editori inviano messaggi a un argomento in entrata di Amazon SNS.
2. Una funzione Lambda sottoscritta all'argomento in entrata controlla il contenuto del messaggio.
3. La funzione Lambda sfrutta Amazon Bedrock Guardrails per rilevare i dati sensibili nel messaggio e applicare le tue policy:
 - LOG: registra i rilevamenti sensibili e pubblica il messaggio originale.
 - BLOCCA: elimina completamente il messaggio.
 - REDACT: oscura i dati sensibili e pubblica il messaggio redatto.
4. I messaggi elaborati vengono pubblicati sull'argomento Amazon SNS di destinazione per essere recapitati agli abbonati all'argomento.

Per ulteriori indicazioni e codice di esempio, consulta [Proteggi i dati sensibili nei messaggi SNS utilizzando Amazon Bedrock Guardrails](#).

Visualizzazione delle politiche di protezione dei dati dei messaggi esistenti

Se attualmente utilizzi la protezione dei dati dei messaggi Amazon SNS, puoi rivedere le policy configurate tramite o. Console di gestione AWS AWS CLI

Usando il Console di gestione AWS

1. Accedi alla console [Amazon SNS](#).
2. Seleziona Argomenti dal pannello di navigazione.
3. Scegli un argomento per visualizzarne i dettagli.
4. Controlla se una politica di protezione dei dati è configurata nella scheda Politica di protezione dei dati.

Usando il AWS CLI

Per verificare se per un argomento specifico è abilitata la protezione dei dati dei messaggi, esegui il comando seguente. Sostituisci *topic-arn* con il tuo argomento di Amazon SNS ARN.

```
aws sns get-data-protection-policy --resource-arn topic-arn
```

Disattivazione della protezione dei dati dei messaggi Amazon SNS

Puoi rimuovere le politiche di protezione dei dati dai tuoi argomenti di Amazon SNS in qualsiasi momento, indipendentemente dal fatto che tu stia migrando a un'alternativa basata su Lambda o che non richieda più la protezione dei dati. Il processo di rimozione delle policy può essere completato tramite gli strumenti IaC (Infrastructure as Code) o i Console di gestione AWS tuoi AWS CLI strumenti Infrastructure as Code (IaC).

Utilizzando il Console di gestione AWS

1. Accedi alla console [Amazon SNS](#).
2. Seleziona Argomenti dal pannello di navigazione.
3. Scegli l'argomento che desideri modificare.
4. Seleziona Edit (Modifica).
5. Vai alla sezione Politica di protezione dei dati.
6. Rimuovi la configurazione della politica di protezione dei dati associata all'argomento.

Usando il AWS CLI

Per disabilitare la protezione dei dati dei messaggi, elimina la politica di protezione dei dati dal tuo argomento. Sostituisci *topic-arn* con il tuo argomento di Amazon SNS ARN.

```
aws sns put-data-protection-policy --resource-arn topic-arn --data-protection-policy ""
```

Se hai altre domande, contatta l'[AWS assistenza](#).

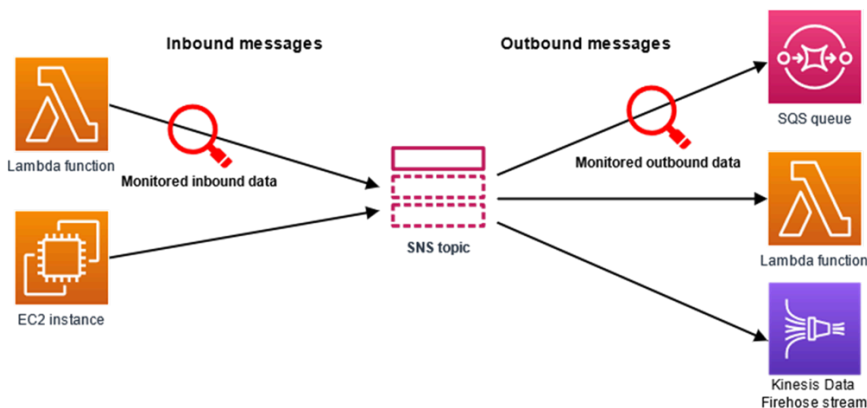
Comprendere le politiche di protezione dei dati di Amazon SNS

⚠ Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Cosa sono le policy di protezione dei dati?

Amazon SNS utilizza le policy di protezione dei dati per selezionare i dati sensibili da sottoporre a scansione e le operazioni che desideri intraprendere per proteggere tali dati dall'interscambio a livello di argomenti Amazon SNS. Per selezionare i dati sensibili di interesse, utilizza gli [identificatori di dati](#). La protezione dei dati dei messaggi di Amazon SNS rileva quindi la presenza di dati sensibili utilizzando il machine learning e i criteri di ricerca. Per agire sugli identificatori di dati trovati, è possibile definire un'operazione di verifica, deidentificazione o rifiuto. Queste operazioni consentono di registrare i dati sensibili trovati (o non trovati), di mascherare o oscurare i dati sensibili o di rifiutare il recapito dei messaggi.

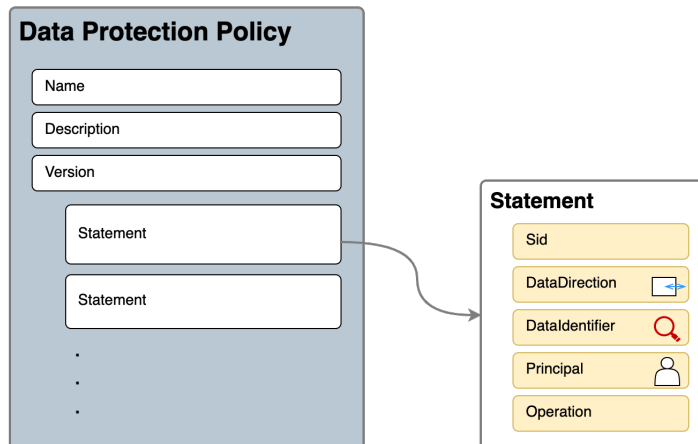


Come è strutturata una policy di protezione dei dati?

Come illustrato nella figura riportata di seguito, un documento relativo alla policy di protezione dei dati include questi elementi:

- Informazioni opzionali sulla policy nella parte superiore del documento
- Una o più istruzioni singole

Ogni istruzione include informazioni su una singola autorizzazione.



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

Proprietà JSON per la policy di protezione dei dati

Una policy di protezione dei dati richiede le seguenti informazioni di base ai fini dell'identificazione:

- Name (Nome): nome della policy.
- Description (Descrizione): (facoltativo) la descrizione della policy.
- Version (Versione): la versione del linguaggio della policy. La versione corrente è 2021-06-01.
- Statement (Dichiarazione): l'elenco di dichiarazioni che specificano le operazioni della policy di protezione dei dati.

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
```

```
"Statement": [  
    ...  
]  
}
```

Proprietà JSON per una dichiarazione di policy

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

- **Sid:** (facoltativo) l'identificatore della dichiarazione.
- **DataDirection**— In entrata (per le richieste Publish API) o in uscita (per le consegne di notifiche) per quanto riguarda l'argomento Amazon SNS.
- **DataIdentifier**— I dati sensibili che l'argomento Amazon SNS deve analizzare. Ad esempio, nome, indirizzo o numero di telefono.
- **Principale:** il principale IAM che ha pubblicato nell'argomento o il principale IAM che ha effettuato la sottoscrizione all'argomento.
- **Operation (Operazione):** l'operazione successiva, ovvero Audit (Verifica), De-identify (Deidentificazione) (mascheramento o oscuramento) o Deny (Rifiuto) (blocco), eseguita dall'argomento Amazon SNS se è stata rilevata la presenza di dati sensibili.

```
{  
  "Sid": "basicPII-inbound-protection",  
  "DataDirection": "Inbound",  
  "Principal": ["*"],  
  "DataIdentifier": [  
    "arn:aws:dataprotection::aws:data-identifier/Name",  
    "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"  
  ],  
  "Operation": {  
    ...  
  }  
}
```

Proprietà JSON per un'operazione della dichiarazione di policy

Una dichiarazione di policy definisce una delle seguenti operazioni di protezione dei dati.

- [Audit](#) (Verifica): genera parametri e registri di risultati della ricerca senza interrompere la pubblicazione o il recapito dei messaggi.

- [De-identify](#) (Deidentificazione): maschera o oscura i dati sensibili senza interrompere la pubblicazione dei messaggi.
- [Deny](#) (Rifiuto): blocca la richiesta di pubblicazione di Amazon SNS o non finalizza il recapito dei messaggi.

Come faccio a determinare i principali IAM per la policy di protezione dei dati?

La protezione dei dati dei messaggi utilizza due principali IAM che interagiscono con Amazon SNS.

1. Principale dell'API Publish (in entrata): il principale IAM autenticato che chiama l'API Amazon SNS Publish.
2. Subscription Principal (Principale di sottoscrizione) (in uscita): il principale IAM autenticato che ha chiamato l'API Subscribe durante la creazione della sottoscrizione.

SubscriptionPrincipal è una proprietà di sottoscrizione Amazon SNS disponibile pubblicamente e che può essere recuperata dall'API GetSubscriptionAttributes.

```
{
  "Attributes": {
    "SubscriptionPrincipal": "arn:aws:iam::123456789012:user/NoNameAccess",
    "Owner": "123412341234",
    "RawMessageDelivery": "true",
    "TopicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
    "Endpoint": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
    "Protocol": "sqs",
    "PendingConfirmation": "false",
    "ConfirmationWasAuthenticated": "true",
    "SubscriptionArn": "arn:aws:sns:us-east-1:123412341234:PII-data-
topic:5d8634ef-67ef-49eb-a824-4042b28d6f55"
  }
}
```

Operazioni relative alla politica di protezione dei dati in Amazon SNS

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Di seguito sono riportati alcuni esempi di policy di protezione dei dati che puoi utilizzare per controllare e rifiutare i dati sensibili. Per un tutorial completo che include un'applicazione di esempio, consulta il post di blog relativo alla [presentazione della protezione dei dati dei messaggi per Amazon SNS](#).

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'argomento Amazon SNS trova dati sensibili nel payload.
2. NoFindingsDestination— La destinazione di registrazione quando l'argomento Amazon SNS non trova dati sensibili nel payload.

Puoi utilizzare 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/.
- (Facoltativo): DeliveryStream deve rientrare nell'area tematica e avere Direct PUT come fonte del flusso di distribuzione. Per ulteriori dettagli, consulta [Source, Destination and Name](#) nella Amazon Data Firehose Developer Guide.
- Amazon S3 (facoltativo): il nome di un bucket Amazon S3. [L'utilizzo del bucket Amazon S3 con la crittografia SSE-KMS abilitata richiede azioni aggiuntive](#).

```
{  
  "Operation": {
```

```

"Audit": {
  "SampleRate": "99",
  "FindingsDestination": {
    "CloudWatchLogs": {
      "LogGroup": "/aws/vendedlogs/log-group-name"
    },
    "Firehose": {
      "DeliveryStream": "delivery-stream-name"
    },
    "S3": {
      "Bucket": "bucket-name"
    }
  },
  "NoFindingsDestination": {
    "CloudWatchLogs": {
      "LogGroup": "/aws/vendedlogs/log-group-name"
    },
    "Firehose": {
      "DeliveryStream": "delivery-stream-name"
    },
    "S3": {
      "Bucket": "bucket-name"
    }
  }
}
}
}
}

```

Autorizzazioni richieste per specificare le destinazioni della registrazione

Quando specifichi le destinazioni di registrazione nella policy di protezione dei dati, devi aggiungere le seguenti autorizzazioni alla policy di identità IAM del principale IAM che chiama l'API `PutDataProtectionPolicy` di Amazon SNS o l'API `CreateTopic` con il parametro `--data-protection-policy`.

Destinazione della verifica	Autorizzazione IAM
Predefinita	logs:CreateLogDelivery
	logs:GetLogDelivery
	logs:UpdateLogDelivery

Destinazione della verifica	Autorizzazione IAM
	logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy L'utilizzo del bucket Amazon S3 con la crittografia SSE-KMS abilitata richiede azioni aggiuntive.

JSON

```
{
  "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:us-west-1:123456789012:SampleLogGroupName:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}

```

Policy di chiave richiesta per l'uso con SSE-KMS

Se utilizzi un bucket Amazon S3 come 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 (SSE-KMS). AWS KMS keys Per ulteriori informazioni, consulta [Protezione dei dati con la crittografia lato server](#) nella Guida per l'utente di Amazon S3.

Se si sceglie SSE-S3, non è richiesta alcuna configurazione aggiuntiva. Amazon S3 gestisce la chiave di crittografia.

Se si sceglie SSE-KMS, è necessario utilizzare una chiave gestita dal cliente. Devi aggiornare la policy delle chiavi per la chiave gestita dal cliente in modo che l'account di consegna del log possa scrivere nel bucket S3. Per ulteriori informazioni sulla politica delle chiavi richiesta per l'uso con SSE-KMS, consulta la [crittografia lato server con bucket Amazon S3 nella Amazon Logs User Guide](#).

CloudWatch

Esempi di registro della destinazione della verifica

Nell'esempio seguente, `callerPrincipal` viene utilizzato per identificare l'origine del contenuto sensibile e `messageID` viene utilizzato come un riferimento per eseguire il controllo in base alla risposta dell'API Publish.

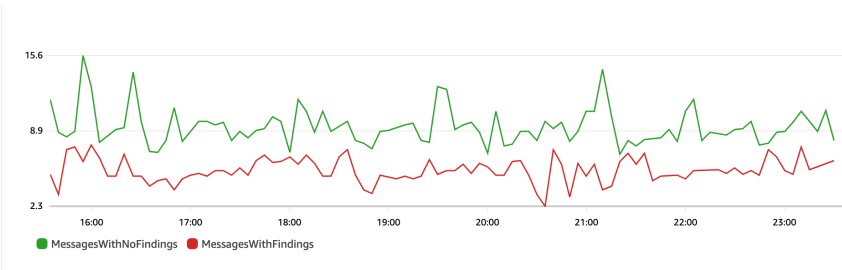
```
{
  "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
  "auditTimestamp": "2022-05-12T2:10:44Z",
  "callerPrincipal": "arn:aws:iam::123412341234:role/Publisher",
  "resourceArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
  "dataIdentifiers": [
    {
      "name": "Name",
      "count": 1,
      "detections": [
        {
          "start": 1,
          "end": 2
        }
      ]
    },
    {
      "name": "PhoneNumber",
      "count": 2,
      "detections": [
        {
          "start": 3,
          "end": 4
        },
        {
          "start": 5,
          "end": 6
        }
      ]
    }
  ]
}
```

```
}

```

Parametri dell'operazione di verifica

Quando un'operazione di controllo ha specificato la proprietà FindingsDestination o la NoFindingsDestination proprietà, i proprietari dell'argomento ricevono anche 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

Carattere di mascheramento supportato	Nome
!	Punto esclamativo
\$	Simbolo del dollaro
%	Segno percentuale
&	E commerciale
()	Parentesi
+	Segno più
,	Virgola
-	Trattino
.	Periodo
^	Barra, barra rovesciata
#	Segno numerico
:	Due punti
;	Punto e virgola
=, <>	Uguale a, minore di o maggiore di
@	Chiocciola
[]	Parentesi
^	Simbolo dell'accento circonflesso
—	Carattere di sottolineatura
`	Accento grave
	Barra verticale

Carattere di mascheramento supportato	Nome
~	Simbolo della tilde

- **RedactConfig**— Redigi rimuovendo completamente i dati. Ad esempio, ssn: 123-45-6789 diventa ssn: .

```
{
  "Operation": {
    "Deidentify": {
      "RedactConfig": {}
    }
  }
}
```

In un messaggio in entrata, i dati sensibili vengono deidentificati dopo l'operazione di verifica e il chiamante dell'API `SNS:Publish` riceve il seguente errore di parametro non valido quando l'intero messaggio è sensibile.

Error code: `AuthorizationError` ...

Operazione di rifiuto

L'operazione `Deny` (Rifiuto) interrompe l'esecuzione della richiesta API `Publish` o la consegna del messaggio se il messaggio contiene dati sensibili. L'oggetto dell'operazione `Deny` (Rifiuto) è vuoto in quanto non richiede una configurazione aggiuntiva.

```
"Operation": {
  "Deny": {}
}
```

In un messaggio in entrata, il chiamante dell'API `SNS:Publish` riceve un errore di autorizzazione.

Error code: `AuthorizationError` ...

In un messaggio in uscita, l'argomento Amazon SNS non consegna il messaggio alla sottoscrizione. Per tenere traccia di operazioni di consegna non autorizzate, abilita l'opzione [Delivery status logging](#) (Registrazione dello stato della consegna) dell'argomento. Nell'esempio seguente viene mostrato un esempio di registro dello stato della consegna:

```
{
```

```
"notification": {
  "messageMD5Sum": "29638742ffb68b32cf56f42a79bcf16b",
  "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
  "topicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
  "timestamp": "2022-05-12T2:12:44Z"
},
"delivery": {
  "deliveryId": "98236591c-56aa-51ee-a5ed-0c7d43493170",
  "destination": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
  "providerResponse": "The topic's data protection policy prohibits this message
from being delivered to <subscription-arn>",
  "dwellTimeMs":20,
  "attempts":1,
  "statusCode": 403
},
"status": "FAILURE"
}
```

Esempi di policy di protezione dei dati di Amazon SNS

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Gli esempi riportati di seguito sono policy di protezione dei dati che puoi utilizzare per controllare e rifiutare i dati sensibili. Per un tutorial completo che include un'applicazione di esempio, consulta il post di blog relativo alla [presentazione della protezione dei dati dei messaggi per Amazon SNS](#).

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](#) e [Amazon S3](#). <https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html>

Ad esempio, puoi creare una policy di verifica per valutare se uno dei tuoi sistemi invia o riceve inavvertitamente dati sensibili. Se i risultati della verifica mostrano che i sistemi inviano i dati relativi alle carte di credito a sistemi che non li richiedono, puoi implementare una policy di blocco per impedire che ciò accada.

L'esempio seguente verifica il 99% dei messaggi che attraversano l'argomento cercando i numeri delle carte di credito e inviando i risultati a CloudWatch Logs, Firehose e Amazon S3.

Policy di protezione dei dati:

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": ["*"],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Audit": {
          "SampleRate": "99",
          "FindingsDestination": {
            "CloudWatchLogs": {
              "LogGroup": "<example log name>"
            },
            "Firehose": {
              "DeliveryStream": "<example stream name>"
            },
            "S3": {
              "Bucket": "<example bucket name>"
            }
          }
        }
      }
    }
  ]
}
```

Esempio di formato dei risultati della verifica:

```
{
  "messageId": "...",
  "callerPrincipal": "arn:aws:sts::123456789012:assumed-role/ExampleRole",
  "resourceArn": "arn:aws:sns:us-east-1:123456789012:ExampleArn",
  "dataIdentifiers": [
```

```

    {
      "name": "CreditCardNumber",
      "count": 1,
      "detections": [
        { "start": 1, "end": 2 }
      ]
    }
  ],
  "timestamp": "2021-04-20T00:33:40.241Z"
}

```

Policy di esempio con dichiarazione di deidentificazione tramite mascheramento in entrata

L'esempio seguente impedisce a un utente di pubblicare un messaggio in un argomento con `CreditCardNumber` mascherando i dati sensibili contenuti nel messaggio.

```

{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}

```

Esempio di risultati di deidentificazione tramite mascheramento in entrata:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is #####
```

Policy di esempio con dichiarazione di deidentificazione tramite oscuramento in entrata

L'esempio seguente impedisce a un utente di pubblicare un messaggio in un argomento con `CreditCardNumber` oscurando i dati sensibili dal contenuto del messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

Esempio di risultati di deidentificazione in entrata:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```


Policy di esempio con dichiarazione di anonimizzazione tramite mascheramento in uscita

L'esempio seguente impedisce a un utente di ricevere un messaggio con `CreditCardNumber` mascherando i dati sensibili contenuti nel messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "-"
          }
        }
      }
    }
  ]
}
```

Esempio di risultati di anonimizzazione tramite mascheramento in uscita:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is -----
```

Policy di esempio con dichiarazione di anonimizzazione tramite oscuramento in uscita

L'esempio seguente impedisce a un utente di ricevere un messaggio con `CreditCardNumber` oscurando i dati sensibili contenuti nel messaggio.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

Esempio di risultati di anonimizzazione in uscita:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Esempio di policy con una dichiarazione di rifiuto in entrata

L'esempio seguente impedisce a un utente di pubblicare in un argomento un messaggio contenente `CreditCardNumber`. I payload negati nella risposta dell'API hanno il codice di stato "403 `AuthorizationError`".

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
```

```

    "DataDirection": "Inbound",
    "Principal": [
      "arn:aws:iam::123456789012:user/ExampleUser"
    ],
    "DataIdentifier": [
      "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
    ],
    "Operation": {
      "Deny": {}
    }
  }
]
}

```

Esempio di policy con una dichiarazione di rifiuto in uscita

L'esempio seguente 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": "2e8f58ff2eed723b56b15493fbfb5a5",
  "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
  "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
  "timestamp": "2022-09-08 15:40:57.144"
},
"delivery": {
  "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
  "destination": "arn:aws:sqs:us-east-2:664555388960:test",
  "providerResponse": "The topic's data protection policy prohibits this message from being delivered to <subscription arn>",
  "dwellTimeMs": 22,
  "attempts": 1,
  "statusCode": 403
},
"status": "FAILURE"
}
```

Creazione di policy di protezione dei dati in Amazon SNS

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Le [policy di protezione dei dati](#) semplificano la protezione dei dati pubblicati negli argomenti Amazon SNS tramite la verifica, la deidentificazione (mascheramento o oscuramento) e il rifiuto (blocco) delle informazioni sensibili in transito tra applicazioni o Servizi AWS. Puoi utilizzare l' AWS API, AWS CLI CloudFormation, o Console di gestione AWS per creare politiche di protezione dei dati in Amazon SNS. È possibile definire solo una policy per argomento Amazon SNS. Ciascuna policy di protezione dei dati può avere una o più dichiarazioni di deidentificazione e rifiuto, ma solo una dichiarazione di verifica.

Argomenti

- [Creazione di politiche di protezione dei dati in Amazon SNS utilizzando l'API](#)
- [Creazione di politiche di protezione dei dati in Amazon SNS utilizzando la 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 policy di protezione dei dati di Amazon SNS per proteggere i dati dei messaggi utilizzando l'SDK](#)

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

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

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

Creazione di una politica di protezione dei dati tramite API

Crea una policy di protezione dei dati di Amazon SNS utilizzando l' AWS API.

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

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

- [CreateTopic](#)

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

Chiamare una delle seguenti operazioni:

- [GetDataProtectionPolicy](#)
- [PutDataProtectionPolicy](#)

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

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

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

Creazione di politiche di protezione dei dati utilizzando AWS CLI

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

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (AWS CLI)

Usa questa opzione per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

- [create-topic](#)

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

Chiamare una delle seguenti operazioni:

- [get-data-protection-policy](#)
- [put-data-protection-policy](#)

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

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

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

Creazione di policy di protezione dei dati (CloudFormation)

Crea una policy di protezione dei dati di Amazon SNS utilizzando CloudFormation

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (CloudFormation)

Usa questa opzione per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

- [AWS::SNS::Topic](#)

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

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

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

Per creare una policy di protezione dei dati in combinazione con un argomento Amazon SNS (console)

Usa questa opzione per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS.

1. Accedi alla [console Amazon SNS](#).
2. Scegli un argomento o creane uno nuovo. Per ulteriori dettagli sulla creazione di argomenti, consulta [Creare un argomento Amazon SNS](#).
3. Nella pagina Create topic (Crea argomento), nella sezione Details (Dettagli) scegli Standard.
 - a. Inserisci un Nome per l'argomento.
 - b. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.
4. Espandi Data protection policy (Policy di protezione dei dati).
5. Scegli una modalità in Configuration mode (Modalità di configurazione):
 - Basic (Base): consente di definire una policy di protezione dei dati utilizzando un semplice menu.
 - Advanced (Avanzato): consente di definire una policy di protezione dei dati personalizzata utilizzando JSON.
6. (Facoltativo) Per creare un identificatore di dati personalizzato, espandi la sezione Configurazione dell'identificatore di dati personalizzato, procedi come segue:
 - a. Immetti un nome univoco per l'identificatore di dati personalizzato. I nomi di identificatori di dati personalizzati supportano i caratteri alfanumerici, il carattere di sottolineatura (_) e il trattino (-). Sono supportati fino a 128 caratteri. Questo nome non può condividere lo stesso nome di un [identificatore di dati gestito](#). Per un elenco completo delle limitazioni relative agli identificatori di dati personalizzati, consulta [Vincoli degli identificatori di dati personalizzati](#).
 - b. Inserisci un'espressione regolare (Regex) per l'identificatore di dati personalizzato. Regex supporta caratteri alfanumerici, caratteri Regex riservati e simboli. Regex ha una lunghezza massima di 200 caratteri. Se Regex è troppo complicato, Amazon SNS fallirà la chiamata API. Per un elenco completo delle Regex limitazioni, consulta [Vincoli 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](#).
 - — Firehose soddisfa le richieste di streaming in tempo reale su Splunk e OpenSearch Amazon Redshift per ulteriori analisi dei log. [Per informazioni su, consulta la Guida per l'utente.](#)
 - 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 de-identificazione per, seleziona AWS gli account o i principali IAM a cui si applica questa dichiarazione di de-identificazione. Puoi applicarlo a tutti gli AWS account o a AWS account o entità IAM specifici (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 i seguenti principali [IAM](#):

- IAM account principals (Principali degli account IAM): ad esempio, `arn:aws:iam::AWS-account-ID:root`.
 - IAM role principals (Principali dei ruoli IAM): ad esempio, `arn:aws:iam::AWS-account-ID:role/role-name`.
 - IAM user principals (Principali degli utenti IAM): ad esempio, `arn:aws:iam::AWS-account-ID:user/user-name`.
- iii. Per De-identify Option (Opzione di deidentificazione), seleziona il modo in cui desideri deidentificare i dati sensibili. Sono supportate le seguenti opzioni:
 - Redact (Oscuramento): rimuove completamente i dati. Ad esempio, l'e-mail: `classified@amazon.com` diventa l'e-mail: `.`
 - Mask (Mascheramento): sostituisce i dati con caratteri singoli. Ad esempio, l'e-mail: `classified@amazon.com` diventa l'e-mail: `*****`.
 - iv. (Facoltativo) Continua ad aggiungere le dichiarazioni di deidentificazione, se necessario.
- c. Add deny statement (Aggiungi dichiarazione di rifiuto): configura per quali dati sensibili impedire lo spostamento nell'argomento e per quali principali impedire la trasmissione di tali dati.
 - i. Per Data direction (Direzione dei dati), scegli la direzione dei messaggi per la dichiarazione di rifiuto:

- Inbound messages (Messaggi in entrata): applica questa dichiarazione di rifiuto ai messaggi inviati all'argomento.
 - Outbound messages (Messaggi in uscita): applica questa dichiarazione di rifiuto ai messaggi che l'argomento invia agli endpoint di sottoscrizione.
- ii. Scegli Data identifiers (Identificatori di dati) per definire i dati sensibili che desideri negare.
 - iii. In IAM principals (Principali IAM) seleziona i principali IAM a cui applicare questa dichiarazione di rifiuto. Puoi applicarlo a tutti gli AWS account, a entità specifiche Account AWS o IAM (ad esempio, account root, ruoli o utenti) che utilizzano l'account IDs o l'entità IAM ARNs. Separare più ARNs elementi IDs o utilizzare una virgola (,). Sono supportati i seguenti principali [IAM](#):
 - IAM account principals (Principali degli account IAM): ad esempio, `arn:aws:iam::AWS-account-ID:root`.
 - IAM role principals (Principali dei ruoli IAM): ad esempio, `arn:aws:iam::AWS-account-ID:role/role-name`.
 - IAM user principals (Principali degli utenti IAM): ad esempio, `arn:aws:iam::AWS-account-ID:user/user-name`.
 - iv. (Facoltativo) Continua ad aggiungere le dichiarazioni di rifiuto, se necessario.

Creazione di policy di protezione dei dati di Amazon SNS per proteggere i dati dei messaggi utilizzando l'SDK

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

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

Creazione di politiche di protezione dei dati utilizzando l'SDK AWS

Crea una policy di protezione dei dati di Amazon SNS utilizzando l' AWS SDK.

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

Usa le seguenti opzioni per creare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

Java

```
/**
 * For information regarding CreateTopic see this documentation topic:
 *
 * https://docs.aws.amazon.com/code-samples/latest/catalog/javav2-sns-src-main-java-com-example-sns-CreateTopic.java.html
 */

public static String createSNSTopicWithDataProtectionPolicy(SnsClient snsClient,
String topicName, String dataProtectionPolicy) {

    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .dataProtectionPolicy(dataProtectionPolicy)
            .build();

        CreateTopicResponse result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {CreateTopicCommand } from "@aws-sdk/client-sns";
import {snsClient } from "../libs/snsClient.js";

// Set the parameters
const params = { Name: "TOPIC_NAME", DataProtectionPolicy:
    "DATA_PROTECTION_POLICY" };

const run = async () => {
```

```
try {
    const data = await snsClient.send(new CreateTopicCommand(params));
    console.log("Success.", data);
    return data; // For unit tests.
} catch (err) {
    console.log("Error", err.stack);
}
};
run();
```

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

Usa le seguenti opzioni per creare o recuperare una nuova policy di protezione dei dati in combinazione con un argomento standard di Amazon SNS:

Java

```
public static void putDataProtectionPolicy(SnsClient snsClient, String topicName,
String dataProtectionPolicy) {

    try {
        PutDataProtectionPolicyRequest request =
PutDataProtectionPolicyRequest.builder()
            .resourceArn(topicName)
            .dataProtectionPolicy(dataProtectionPolicy)
            .build();

        PutDataProtectionPolicyResponse result =
snsClient.putDataProtectionPolicy(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
            + "\n\nTopic " + request.resourceArn()
            + " DataProtectionPolicy " + request.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```

public static void getDataProtectionPolicy(SnsClient snsClient, String topicName) {

    try {
        GetDataProtectionPolicyRequest request =
        GetDataProtectionPolicyRequest.builder()
            .resourceArn(topicName)
            .build();

        GetDataProtectionPolicyResponse result =
        snsClient.getDataProtectionPolicy(request);

        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
            + "\n\nDataProtectionPolicy: \n\n" + result.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

JavaScript

```

// Import required AWS SDK clients and commands for Node.js
import {PutDataProtectionPolicyCommand, GetDataProtectionPolicyCommand } from "@aws-
sdk/client-sns";
import {snsClient } from "../libs/snsClient.js";

// Set the parameters
const putParams = { ResourceArn: "TOPIC_ARN", DataProtectionPolicy:
    "DATA_PROTECTION_POLICY" };

const runPut = async () => {
    try {
        const data = await snsClient.send(new
        PutDataProtectionPolicyCommand(putParams));
        console.log("Success.", data);
        return data; // For unit tests.
    } catch (err) {
        console.log("Error", err.stack);
    }
};
runPut();

// Set the parameters

```

```
const getParams = { ResourceArn: "TOPIC_ARN" };

const runGet = async () => {
  try {
    const data = await snsClient.send(new
    GetDataProtectionPolicyCommand(getParams));
    console.log("Success.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};

runGet();
```

Eliminazione di policy di protezione dei dati in Amazon SNS

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

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

Per informazioni generali sulle policy di protezione dei dati di Amazon SNS, consulta [Comprendere le politiche di protezione dei dati di Amazon SNS](#).

Il numero e la dimensione delle risorse delle policy di protezione dei dati di Amazon SNS in un account AWS sono limitati. Per ulteriori informazioni, consultare [Amazon SNS API throttling](#) (Limitazione della larghezza di banda della rete nell'API SNS) nei Riferimenti generali di AWS.

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 [console Amazon SNS](#).
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 policy di protezione dei dati utilizzando una stringa JSON vuota

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

Eliminazione di una 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 di dati Amazon SNS

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

Amazon SNS utilizza una combinazione di criteri e tecniche, tra cui machine learning e criteri di ricerca, per rilevare i dati sensibili. Questi criteri e tecniche, denominati collettivamente identificatori di dati, possono rilevare un elenco ampio e crescente di tipi di dati sensibili per molti Paesi e regioni. Gli identificatori di dati gestiti di Amazon SNS offrono tipi di dati pre-configurati per proteggere i dati finanziari, le informazioni sanitarie personali (PHI) e le informazioni di identificazione personale (PII). Puoi anche utilizzare identificatori di dati personalizzati per creare identificatori di dati personalizzati in base al tuo caso d'uso specifico.

Using managed data identifiers in Amazon SNS

Cosa sono gli identificatori di dati gestiti?

Gli identificatori di dati gestiti di Amazon SNS 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 policy di protezione dei dati, puoi configurare Amazon SNS in modo che utilizzi questi identificatori per analizzare i messaggi che attraversano l'argomento ed eseguire operazioni specifiche quando tali dati vengono rilevati.

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

- Credenziali, come chiavi private o AWS chiavi di accesso segrete
- Identificatori di dispositivo, ad esempio indirizzo IP o indirizzo MAC
- Informazioni finanziarie, ad esempio i numeri di carte di credito
- Dati sanitari personali (PHI), ad esempio l'assicurazione sanitaria o i numeri di assistenza sanitaria
- Informazioni di identificazione personale (PII), ad esempio patenti di guida o codici fiscali

All'interno di ogni categoria, Amazon SNS è in grado di rilevare più tipi di dati sensibili. Negli argomenti di questa sezione sono elencati e descritti i vari tipi e tutti i requisiti pertinenti per la rilevazione. Per ogni tipo, vengono indicati anche gli identificatori univoci (ID) degli identificatori di dati gestiti progettati per rilevare i dati. Quando si crea una policy di protezione dei dati, è possibile utilizzare questo ID per includere l'identificatore di dati gestiti da rilevare per la protezione dei dati dei messaggi.

Requisiti delle parole chiave

Per rilevare determinati tipi di dati sensibili, Amazon SNS cerca parole chiave in prossimità dei dati. Se questo è il caso di un particolare tipo di dati, in un argomento più avanti in questa sezione sono indicati i requisiti specifici relativi alle parole chiave per tali dati.

Le parole chiave non distinguono tra maiuscole e minuscole. Inoltre, se una parola chiave contiene uno spazio, Amazon SNS abbina automaticamente le varianti di parole chiave che non contengono lo spazio o che contengono un trattino basso (_) o un trattino (-) anziché lo spazio. In alcuni casi, Amazon SNS espande o accorcia una parola chiave per rispondere alle varianti più comuni della parola chiave.

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

Nella tabella seguente sono elencati e descritti i tipi di dati relativi a credenziali, dispositivi, informazioni finanziarie e dati medici sanitari protetti (PHI) che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti. Questi dati si aggiungono a determinati tipi di dati che potrebbero anche essere considerati informazioni personali di identificazione (PII).

Gli identificatori di dati dipendenti dalla regione prevedono il nome dell'identificatore con un trattino e i codici a due lettere (ISO 3166-1 alpha-2). Ad esempio, DriversLicense -US.

Identificatore	Categoria	Paesi/lingue
BankAccountNumber	Servizi finanziari	DE, ES, FR, GB, IT
CepCode	Personale	BR
Cnpj	Personale	BR
CpfCode	Personale	BR
DriversLicense	Personale	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Integrità	US
ElectoralRollNumber	Personale	GB
HealthInsuranceCardNumber	Integrità	UE
HealthInsuranceClaimNumber	Integrità	US
HealthInsuranceNumber	Integrità	FR
HealthcareProcedureCode	Integrità	US
IndividualTaxIdentificationNumber	Personale	US

Identificatore	Categoria	Paesi/lingue
InseeCode	Personale	FR
MedicareBeneficiaryNumber	Integrità	US
NationalDrugCode	Integrità	US
NationalIdentificationNumber	Personale	DE, ES, IT
NationalInsuranceNumber	Personale	GB
NationalProviderId	Integrità	US
NhsNumber	Integrità	GB
NieNumber	Personale	ES
NifNumber	Personale	ES
PassportNumber	Personale	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	Personale	CA
PersonalHealthNumber	Integrità	CA
PhoneNumber	Personale	BR, DE, ES, FR, GB, IT, US
PostalCode	Personale	CA
RgNumber	Personale	BR
SocialInsuranceNumber	Personale	CA
Ssn	Personale	ES, US
TaxId	Personale	DE, ES, FR, GB
ZipCode	Personale	US

Identificatori supportati indipendenti dalla lingua/regione

Identificatore	Categoria
Indirizzo	Personale
AwsSecretKey	Credenziali
CreditCardExpiration	Servizi finanziari
CreditCardNumber	Servizi finanziari
CreditCardSecurityCode	Servizi finanziari
EmailAddress	Personale
IpAddress	Personale
LatLong	Personale
Nome	Personale
OpenSshPrivateKey	Credenziali
PgpPrivateKey	Credenziali
PkcsPrivateKey	Credenziali
PuttyPrivateKey	Credenziali
VehicleIdentificationNumber	Personale

Tipi di dati sensibili di Amazon SNS: credenziali

Nella tabella seguente sono elencati e descritti i tipi di credenziali che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e Regioni
AWS chiave di accesso segreta	AwsSecretKey	aws_secret_access_key, credentials,	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e Regioni
		secret access key, secret key, set-aws-credential	
Chiave privata OpenSSH	OpenSshPrivateKey	No	Qualsiasi
Chiave privata PGP	PgpPrivateKey	No	Qualsiasi
Chiave privata PKCS (Public-Key Cryptography Standard)	PkcsPrivateKey	No	Qualsiasi
Chiave privata PuTTY	PuttyPrivateKey	No	Qualsiasi

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 sensibili di Amazon SNS: dispositivi

Nella tabella seguente sono elencati e descritti i tipi di identificatori di dispositivi che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Indirizzo IP	IpAddress	No	Qualsiasi

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.

ARN di identificatore dei dati del dispositivo

```
arn:aws:dataprotection: :aws:data-identifier/ IpAddress
```

Tipi di dati sensibili di Amazon SNS: Finanziari

Nella tabella seguente sono elencati e descritti i tipi di informazioni finanziarie che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero del conto bancario	BankAccountNumber BankAccountNumber-US	Sì, consulta Parole chiave per i numeri di conto bancario.	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 delle carte di credito hanno riservato a test pubblici:

122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 42222222222222, 4444333322221111, 4462030000000000, 4484070000000000, 49118300000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 52042300800000017, 5204740009900014, 5420923878724339, 5454545454545454, 54553307600000018, 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 bancario internazionale (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 sensibili di Amazon SNS: Informazioni sanitarie protette (PHI)

Nella tabella seguente sono elencati e descritti i tipi di dati sanitari protetti (PHI) che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Numero di registrazione DEA (Drug Enforcement Agency)	DrugEnforcementAgencyNumber	dea number, dea registration	US
Numero EHIC (Health Insurance Card)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d'assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d'assurance maladie, numéro de carte d'assurance,	UE

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
		numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkringsnummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer	
Health Insurance Claim Number (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	US
Numero di identificazione medica e assistenza sanitaria	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR
Codice HCPCS (Healthcare Common Procedure Coding System)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	US

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Paesi e regioni
Numero MBN (Medicare Beneficiary Number)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	US
National Drug Code (NDC)	NationalDrugCode	national drug code, ndc	US
National Provider Identifier (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	US
Numero NHS (National Health Service)	NhsNumber	national health service, NHS	GB
Personal Health Number (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Parole chiave per numeri di identificazione medica e assistenza sanitaria

Per rilevare vari tipi di assicurazioni sanitarie e numeri di identificazione medica, Amazon SNS richiede la prossimità di parole chiave e numeri. Sono inclusi i numeri della tessera sanitaria europea (UE, Finlandia), i numeri di assicurazione sanitaria (Francia), gli identificatori dei beneficiari Medicare (Stati Uniti), i numeri di assicurazione sanitaria (Regno Unito), i numeri NHS (Regno Unito) e i numeri Personal Health Number (PHN) (Canada).

Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Canada	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
UE	assicurazione sanitaria numero, carta assicurazione numero, carte d'assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d'assurance maladie, numéro de carte d'assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvaikutuskortti, 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, sairausvaikutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomen sairausvaikutuskortti, suomi ehic-numero, terveyskortti

Paese o regione	Parole chiave
Francia	carte d'assuré social, carte vitale, insurance card
UK	servizio sanitario nazionale, NHS
US	mbi, medicare beneficiary

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

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

Identificatore di dati PHI 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 sensibili di Amazon SNS: informazioni di identificazione personale (PII)

Nella tabella seguente sono elencati e descritti i tipi di informazioni personali di identificazione (PII) che Amazon SNS è in grado di rilevare utilizzando identificatori di dati gestiti.

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Data di nascita	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	Il supporto include la maggior parte dei formati di data, ad esempio tutte le cifre e le combinazioni di cifre e nomi dei mesi. I componenti della data possono essere separati da spazi, barre (/) o trattini (-).	Qualsiasi
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, código de endereçamento postal	–	Brasile
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	Brasile

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas físicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa física, cpf	–	Brasile

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero identificativo della patente di guida	DriversLicense	Sì, consulta Parole chiave per i numeri identificativi delle patenti di guida.	–	Australia, Austria, Belgio, Bulgaria, Canada, Cipro, Croazia, Danimarca, Estonia, Finlandia, Francia, Germania, Grecia, Irlanda, Italia, Lettonia, Lituania, Lussemburgo, Malta, Paesi Bassi, Polonia, Portogallo, Regno Unito, Repubblica Ceca, Romania, Slovacchi a, 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	Individual TaxIdentification Number	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	US
Institut national de la statistique et des études économiques (INSEE)	InseeCode	Sì, consulta Parole chiave per i numeri di carta d'identità.	–	Francia

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numeri di carta d'identità	NationalIdentificationNumber	Sì, consulta Parole chiave per i numeri di carta d'identità.	Sono inclusi gli identificatori DNI (Documento Nacional de Identidad, Spagna), il codice fiscale (Italia) e i numeri di carta d'identità nazionale (Germania).	Germania, Italia, Spagna
Numero NINO (National Insurance Number)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, national insurance#, , national insurancenum, nin, nino	–	UK
Número de identidad de extranjero (NIE)	NieNumber	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	Spagna

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Número de Identificación Fiscal (NIF)	NifNumber	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	–	Spagna
Numero di passaporto	PassportNumber	Sì, consulta Parole chiave per i numeri di passaporto.	–	Canada, Francia, Germania, Italia, Regno Unito, Spagna, Stati Uniti

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero di residenza permanente (Green Card)	Permanent Residence Number	carte résident permanent , numéro carte résident permanent, numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	–	Canada

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Numero di telefono	PhoneNumber	Brasile: le parole chiave includono anche: cel, celular, fone, móvel, número residencial, número residencial, telefone Altri paesi: cellulare, contatto, fax, numero di fax, smartphone, telefono, mobile, numero di telefono	Sono inclusi i numeri verdi negli Stati Uniti e i numeri di fax. Se una parola chiave si trova in prossimità dei dati, il numero non deve includere il prefisso internazionale. Se una parola chiave non è in prossimità dei dati, il numero deve includere un prefisso internazionale.	Brasile, Canada, Francia, Germania, Italia, Regno Unito, Spagna, Stati Uniti
Codice postale	PostalCode	No	–	Canada
Registro Geral (RG)	RgNumber	Sì, consulta Parole chiave per i numeri di carta d'identità.	–	Brasile
Social Insurance Number (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Canada

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Social Security number (SSN)	Ssn	Spagna: número de la seguridad social, n. di previdenza sociale, numero di previdenza sociale, nprevidenzasociale#, ssn, ssn# Stati Uniti: previdenza sociale, ss#, ssn	–	Spagna, Stati Uniti
Numero identificativo del contribuente o codice fiscale	TaxId	Sì, consulta Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento.	Sono inclusi TIN (Francia); Steueridentifikationsnummer (Germania); CIF (Spagna) e TRN, UTR (Regno Unito).	Francia, Germania, Regno Unito, Spagna
Codice postale (Stati Uniti)	ZipCode	zip code, zip+4	–	US

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Indirizzo postale	Address	No	Sebbene non sia richiesta una parola chiave, il rilevamento richiede che l'indirizzo includa il nome di una città o di un luogo e un CAP o un codice postale.	Australia, Canada, Francia, Germania, Italia, Regno Unito, Spagna, Stati Uniti
Indirizzo e-mail	EmailAddress	email, indirizzo email, e mail, indirizzo e mail	–	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Coordinate del sistema di posizionamento globale (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS è in grado di rilevare le coordinate GPS se le coordinate di latitudine e longitudine sono memorizzate in coppia e sono nel formato di gradi decimali (DD), ad esempio 41,948614, -87.655311. Il supporto non include le coordinate e nel formato DDM (Gradi Decimali Minuti), ad esempio 41°56.9168'N 87°39.3187'W, o nel formato Gradi, Minuti, Secondi (DMS), ad esempio 41°56'55.0104"N 87°39'19.1196"W.	Qualsiasi

Tipo di rilevamento	ID identificatore dei dati gestiti	Parola chiave obbligatoria	Informazioni aggiuntive	Paesi e regioni
Nome completo	Name	No	Amazon SNS è in grado di rilevare solo i nomi completi. Il supporto è limitato ai set di caratteri latini.	Qualsiasi
Numeri di matricola del veicolo (VIN)	VehicleIdentificationNumber	Fahrgeste llnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóvil es, numéro d'identification du véhicule, vehicle identification number, vin, VIN numerus	Amazon SNS è in grado di rilevare VINs che consistono in una sequenza di 17 caratteri e rispettare gli standard ISO 3779 e 3780. Questi standard sono stati progettati per l'uso a livello mondiale.	Qualsiasi

Parole chiave per i numeri identificativi delle patenti di guida

Per rilevare vari tipi di numeri identificativi delle patenti di guida, Amazon SNS richiede la prossimità di parole chiave e numeri. Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Australia	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
Austria	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Belgio	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgaria	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Canada	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit, permis de conduire
Croazia	vozačka dozvola
Cipro	άδεια οδήγησης

Paese o regione	Parole chiave
Repubblica Ceca	číslo licence, číslo licence řidiče, číslo řidičského o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský průkaz, 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, fuehrerscheinnummer, fuhrerschein, fuhrerschein, fuhrerschein- nr, fuhrerschein- nr, fuhrersch einnummer, fuhrerscheinnnummer
Grecia	δεία οδηγησης, adeia odigisis
Ungheria	illesztőprogramok lic, jogosítvány, jogsi, licensszám, 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, fährerschäin
Malta	licenzja 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 conducir, el carnet de conducir, licencia conducir, licencia de manejo, número carnet conducir, número de carnet de conducir, número de permiso conducir, número de permiso de conducir, número licencia conducir, número permiso conducir, permiso conducción, permiso conducir, permiso de conducción
Svezia	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

Paese o regione	Parole chiave
UK	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
US	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

Parole chiave per i numeri di carta d'identità

Per rilevare vari tipi di numeri di carta d'identità, Amazon SNS richiede la prossimità di parole chiave e numeri. Sono inclusi gli identificatori DNI (Documento Nacional de Identidad, Spagna), i codici INSEE (Institut national de la statistique et des études économiques), i numeri delle carte d'identità tedesche e i numeri RG (Registro Geral, Brasile).

Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Brasile	registro geral, rg
Francia	assurance sociale, carte nationale d'identité, cni, code sécurité sociale, French social security number, fssn#, insee, insurance

Paese o regione	Parole chiave
	number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Germania	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italia	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Spagna	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationali dno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

Parole chiave per i numeri di passaporto

Per rilevare vari tipi di numeri di passaporto, Amazon SNS richiede la prossimità di parole chiave e numeri. Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Canada	passepport, passeport#, passport, passport#, passportno, passportno#
Francia	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

Paese o regione	Parole chiave
Germania	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reiseepass, reiseepass-nr, reiseepassnummer
Italia	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
Spagna	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
UK	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
US	passport, travel document

Parole chiave per i numeri di identificazione dei contribuenti e i numeri di riferimento

Per rilevare vari tipi di numeri di identificazione dei contribuenti e di numeri di riferimento, Amazon SNS richiede la prossimità di parole chiave e numeri. Nella tabella seguente sono elencate le parole chiave riconosciute da Amazon SNS per paesi e regioni specifici.

Paese o regione	Parole chiave
Brasile	cadastro de pessoa física, cadastro de pessoa física, cadastro de pessoas físicas, cadastro de pessoas físicas, 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, steueridentifikationsnummer, 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 per informazioni di identificazione personale ARNs (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.

Identificatore di dati PII 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

Identificatore di dati PII 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

Identificatore di dati PII 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

Identificatore di dati PII 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

Identificatore di dati PII 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 gli identificatori di dati personalizzati, puoi indirizzare i casi d'uso delle informazioni di identificazione personale (PII) specifici dell'azienda che gli [identificatori di dati gestiti](#) non sono in grado di fornire. Ad esempio, è possibile 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.

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 gli identificatori di dati personalizzati, puoi indirizzare i casi d'uso delle informazioni di identificazione personale (PII) specifici dell'azienda che gli [identificatori di dati gestiti](#) non sono in grado di fornire. Ad esempio, è possibile 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'argomento Amazon SNS di rilevare i payload che trasportano IDs dipendenti specifici dell'azienda, quindi di IDs mascherarli utilizzando il simbolo cancelletto (#).

1. Creazione di un blocco `Configuration` all'interno della policy di protezione dei dati.
2. Inserisci un `Name` per l'identificatore di dati personalizzato. Ad esempio, **EmployeeId**.
3. Inserisci un `Regex` per l'identificatore di dati personalizzato. Ad esempio, **EID-\d{9}-US**.
4. Riferimento al seguente identificatore di dati personalizzato in una dichiarazione di policy.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Configuration": {
    "CustomDataIdentifier": [
      {"Name": "EmployeeId", "Regex": "EID-\d{9}-US"}
    ]
  },
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": ["*"],
      "DataIdentifier": [
        "EmployeeId"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}
```

5. (Facoltativo) Continua ad aggiungere altri identificatori di dati personalizzati al blocco `Configuration`, se necessario. Le policy di protezione dei dati attualmente supportano un massimo di 10 identificatori di dati personalizzati.

Vincoli degli identificatori di dati personalizzati

Gli identificatori di dati personalizzati di Amazon SNS presentano le seguenti limitazioni:

- Ciascuna policy di protezione dei dati attualmente supporta un massimo di 10 identificatori di dati personalizzati.
- I nomi degli identificatori di dati personalizzati hanno una lunghezza massima di 128 caratteri. Sono supportati i seguenti caratteri:
 - Alfanumerici: (a-zA-Z0-9)
 - Simboli: (' _ ' | ' - ')
- RegEx ha una lunghezza massima di 200 caratteri. Sono supportati i seguenti caratteri:
 - Alfanumerici: (a-zA-Z0-9)
 - Simboli: (' _ ' | '#' | '=' | '@' | '/' | ';' | ',' | '-' | '')
 - Caratteri riservati RegEx: ('^' | '\$' | '?' | '[' | ']' | '{' | '}' | '|' | '\w' | '*' | '+' | '.')
- Gli identificatori di dati personalizzati non possono condividere lo stesso nome di un identificatore di dati gestito.
- Gli identificatori di dati personalizzati devono essere specificati in ogni policy di protezione dei dati per ogni argomento di Amazon SNS.

Consegna dei messaggi Amazon SNS

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 consegna i messaggi all'endpoint nel loro formato originale e non modificato. Scoprirai anche come inviare messaggi da un argomento di Amazon SNS a una coda Amazon SQS in un Account AWS altro modo, fornendo informazioni sulla messaggistica tra account.

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

Inoltre, imparerai come monitorare e interpretare lo stato di consegna dei messaggi, il che fornisce informazioni fondamentali sulla corretta consegna dei messaggi o sui problemi riscontrati. Nei casi in cui il recapito dei messaggi non va a buon fine, conoscerai la procedura di nuovo tentativo di recapito dei messaggi, incluso il modo in cui Amazon SNS tenta automaticamente di recapitare i messaggi per garantire che raggiungano le destinazioni previste. Questo argomento illustra anche l'uso di 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.

Consegna di messaggi non elaborati Amazon SNS

Per evitare che [???](#) gli endpoint [Amazon SQS e HTTP/S](#) elaborino la formattazione JSON dei messaggi, Amazon SNS consente la consegna di messaggi non elaborati:

- Quando abiliti la consegna di messaggi non elaborati per gli endpoint Amazon SQS, tutti i metadati di Amazon SNS vengono rimossi dal messaggio pubblicato e il messaggio viene inviato così com'è.
- Quando abiliti la consegna di messaggi non elaborati per gli HTTP/S endpoint, l'intestazione HTTP `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 HTTP/S endpoint, 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 la consegna di messaggi non elaborati utilizzando un AWS SDK, è necessario utilizzare l'azione `SetSubscriptionAttribute` API e impostare il valore dell'`RawMessageDelivery` attributo su `true`.

Abilitare il recapito di messaggi non elaborati utilizzando il Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Argomenti, scegli un argomento sottoscritto a un Firehose, Amazon SQS o endpoint. HTTP/S
4. Nella **MyTopic** pagina, nella sezione Abbonamento, scegli un abbonamento e scegli Modifica.
5. Nella **EXAMPLE1-23bc-4567-d890-ef12g3hij456** pagina Modifica, 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 coda Amazon SQS. L'unica differenza è che il recapito dei messaggi non elaborati è disabilitato per il primo messaggio e abilitato per il secondo.

- Consegna di messaggi non elaborati disabilitato

```
{
  "Type": "Notification",
  "MessageId": "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
  "TopicArn": "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
  "Subject": "TestSubject",
  "Message": "This is a test message.",
  "Timestamp": "2021-02-16T21:41:19.978Z",
  "SignatureVersion": "1",
  "Signature":
    "FMG5t1ZhJNHLHUXvZgtZz1k24FzVa7oX0T4P03neeXw8ZEXZx6z35j2F0TuNYShn2h0bKNC/
    zLTnMyIx Ezmi2X1sh0BWsJHkrW2xkr58ABZF+4uWHEE73yDVR4SyYAIkP9jstZzDRm
    +bcVs8+T0yaLiEGLrIIIL4esi11lhIkGErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
    rd5071X1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNk1VyPfy"
```

```
+7Ta1MD01zmJu0rExtnSIbZew3foxgx8GT+1bZkLd0ZdtdRj1IyPRP44eyq78sU0Eo/  
LsDr0Iak4ZDpg8dXg==",  
  "SigningCertURL": "https://sns.us-east-2.amazonaws.com/  
SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",  
  "UnsubscribeURL": "https://sns.us-east-2.amazonaws.com/?  
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-  
east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"  
}
```

- Consegna di messaggi non elaborati abilitato

This is a test message.

Attributi dei messaggi e consegna di messaggi non elaborati per gli abbonamenti Amazon SQS

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

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

Questo documento descrive come pubblicare una notifica in un argomento Amazon SNS con una o più sottoscrizioni a code Amazon SQS in un altro account. La procedura di configurazione di argomento e code è identica a quella utilizzata quando questi si trovano nello stesso account (vedi [Esegui l'eliminazione delle notifiche Amazon SNS alle code di Amazon SQS 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 coda Amazon SQS ha un volume di messaggi elevato, raccomandiamo che il proprietario della coda crei la sottoscrizione.

Creazione della sottoscrizione da parte del proprietario della coda

L'account che ha creato la coda Amazon SQS è il proprietario della coda. Quando il proprietario della coda crea la sottoscrizione, la conferma della sottoscrizione non è necessaria. La coda inizia a ricevere le notifiche dall'argomento al termine dell'operazione `Subscribe`. Per consentire al proprietario della coda di effettuare la sottoscrizione all'argomento, il proprietario dell'argomento deve autorizzare l'account del proprietario della coda a chiamare l'operazione `Subscribe` sull'argomento.

Passaggio 1: Per impostare la politica dell'argomento utilizzando il Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, scegliere Argomenti.
3. Selezionare un argomento, quindi scegliere Edit (Modifica).
4. Nella *MyTopic* pagina Modifica, 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 account di delegare l'autorizzazione al proprio utente/ruolo IAM. Solo l'account root o gli utenti amministratori saranno autorizzati a richiamare `sns:Subscribe`. L'IAM user/role deve inoltre consentire `sns:subscribe` alla propria coda di iscriversi.

6. Scegli Save changes (Salva modifiche).

Un utente con le credenziali dell'account 111122223333 può iscriversi a MyTopic

Fase 2: Per aggiungere un abbonamento alla coda Amazon SQS a un argomento in un altro, utilizzare il Account AWS Console di gestione AWS

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

1. Accedere alla [console Amazon SQS](#).
2. Nel pannello di navigazione, scegliere Code.
3. Dall'elenco di code, scegliere la coda per iscriversi all'argomento Amazon SNS.
4. Scegli Subscribe to Amazon SNS topic (Iscriviti all'argomento Amazon SNS).
5. Da Specify an Amazon SNS topic available for this queue menu (Specifica un argomento Amazon SNS disponibile per questo menu di coda), scegliere Amazon SNS topic (Argomento Amazon SNS) per la coda.
6. Scegliere Enter Amazon SNS topic ARN (Inserisci ARN dell'argomento Amazon SNS) e quindi inserire l'Amazon Resource Name (ARN) dell'argomento.
7. Scegli Save (Salva).

 Note

- Per poter comunicare con il servizio, la coda deve disporre delle autorizzazioni per Amazon SNS.
- Poiché sei il proprietario della coda, non devi confermare la sottoscrizione.

Creazione di una sottoscrizione da parte di un utente non proprietario della coda

Qualsiasi utente che crea una sottoscrizione ma non è il proprietario della coda deve confermare la sottoscrizione.

Quando si utilizza l'azione `Subscribe`, Amazon SNS invia una conferma di sottoscrizione alla coda. La sottoscrizione viene visualizzata nella console Amazon SNS, con l'ID di sottoscrizione impostato su Conferma in sospeso.

Per confermare la sottoscrizione, un utente autorizzato a leggere i messaggi dalla coda deve recuperare l'URL di conferma della sottoscrizione e il proprietario della sottoscrizione deve confermare la sottoscrizione utilizzando l'URL di conferma della sottoscrizione. Fino alla conferma della sottoscrizione, nessuna delle notifiche pubblicate nell'argomento viene inviata alla coda. Per confermare la sottoscrizione, è possibile utilizzare la console Amazon SNS o l'azione [ReceiveMessage](#).

Note

Prima di sottoscrivere un endpoint all'argomento, assicurarsi che la coda possa ricevere messaggi dall'argomento impostando l'autorizzazione `sqs:SendMessage` per la coda. Per ulteriori informazioni, consulta [Fase 2: Concedere all'argomento Amazon SNS l'autorizzazione a inviare messaggi alla coda Amazon SNS](#).

Fase 1: Per aggiungere un abbonamento alla coda Amazon SNS a un argomento in un altro, utilizzare il Account AWS Console di gestione AWS

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 [console Amazon SNS](#).
2. Nel pannello di navigazione, scegliere Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegliere Create subscription (Crea sottoscrizione).
4. Nella pagina Create subscription (Crea sottoscrizione), nella sezione Dettagli, eseguire queste operazioni:
 - a. Per Argomento ARN, immettere l'ARN dell'argomento.

- b. Per Protocollo, scegliere Amazon SQS.
- c. Per Endpoint, immettere l'ARN della coda.
- d. Scegli Create Subscription (Crea sottoscrizione).

 Note

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

Il seguente è un esempio di istruzione di policy che consente all'argomento Amazon SNS di inviare un messaggio alla coda Amazon SQS.

```
{
  "Sid": "Stmt1234",
  "Effect": "Allow",
  "Principal": "*",
  "Action": "sqs:SendMessage",
  "Resource": "arn:aws:sqs:us-west-2:111111111111:QueueName",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:sns:us-west-2:555555555555:TopicName"
    }
  }
}
```

Passaggio 2: Per confermare un abbonamento utilizzando il Console di gestione AWS

1. Accedere alla [console Amazon SQS](#).
2. Selezionare la coda con una sottoscrizione in sospeso all'argomento.
3. Scegli Send and receive messages (Invia e ricevi messaggi), quindi seleziona Poll for messages (Polling per i messaggi).

Nella coda viene ricevuto un messaggio con la conferma della sottoscrizione.

4. Nella colonna Corpo eseguire le operazioni seguenti:
 - a. Scegliere Maggiori dettagli.

- b. Nella finestra di dialogo Message Details (Dettagli messaggio), trovare e annotare il valore SubscribeURL. Questo è il collegamento di sottoscrizione (esempio qui di seguito). Per ulteriori dettagli sulla convalida dei token API, consulta [ConfirmSubscription](#) nella Documentazione di riferimento delle API di Amazon SNS.

```
https://sns.us-west-2.amazonaws.com/?  
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-  
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

- c. Prendere nota del collegamento di conferma della sottoscrizione. L'URL deve essere passato dal proprietario della coda al proprietario della sottoscrizione. Il proprietario della sottoscrizione deve inserire l'URL nella [console Amazon SNS](#).
5. Accedere come proprietario della sottoscrizione alla [console Amazon SNS](#). Il proprietario della sottoscrizione effettua la conferma.
6. Scelta dell'argomento pertinente.
7. Scegliere la sottoscrizione pertinente nella tabella degli elenchi di sottoscrizione dell'argomento. È etichettato come "In attesa di conferma".
8. Scegliere Confirm subscription (Conferma sottoscrizione).
9. Viene visualizzato un modale che richiede il collegamento di conferma della sottoscrizione. Incollare il collegamento di conferma della sottoscrizione.
10. Selezionare Confirm subscription (Conferma sottoscrizione) nel modale.

Viene visualizzata una risposta XML, ad esempio:

```
<ConfirmSubscriptionResponse>  
  <ConfirmSubscriptionResult>  
    <SubscriptionArn>arn:aws:sns:us-east-2:123456789012:MyTopic:1234a567-  
bc89-012d-3e45-6fg7h890123i</SubscriptionArn>  
  </ConfirmSubscriptionResult>  
  <ResponseMetadata>  
    <RequestId>abcd1efg-23hi-jkl4-m5no-p67q8rstuvw9</RequestId>  
  </ResponseMetadata>  
</ConfirmSubscriptionResponse>
```

La coda per la quale hai confermato la sottoscrizione è pronta a ricevere messaggi dall'argomento.

11. (Facoltativo) Se visualizzi la sottoscrizione dell'argomento nella console Amazon SNS, puoi verificare che il messaggio Pending Confirmation (Conferma in sospeso) è stato sostituito dall'ARN della sottoscrizione nella colonna Subscription ID (ID sottoscrizione).

Come faccio a forzare una sottoscrizione a richiedere l'autenticazione per le richieste di annullamento della sottoscrizione?

Il proprietario della sottoscrizione deve impostare il flag `AuthenticateOnUnsubscribe` su `true` alla conferma della sottoscrizione.

- `AuthenticateOnUnsubscribe` viene automaticamente impostato su `true` quando il proprietario della coda crea la sottoscrizione.
- `AuthenticateOnUnsubscribe` non può essere impostato su `true` quando si passa al collegamento di conferma della sottoscrizione senza autenticazione.

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

Amazon SNS supporta le spedizioni tra aree geografiche, sia per le regioni abilitate per impostazione predefinita che per [Regioni opt-in](#). Per l'elenco corrente di regioni AWS supportate da Amazon SNS, incluse le regioni abilitate, consultare [Amazon Simple Notification Service endpoints and quotas](#) (Endpoint e quote Amazon Simple Notification Service) nei Riferimenti generali di Amazon Web Services.

Amazon SNS supporta la consegna in più aree geografiche delle notifiche alle code Amazon SQS e a AWS Lambda funzioni. Quando una delle regioni è una regione opt-in, devi specificare un'entità Amazon SNS diverso nella policy della risorsa sottoscritta.

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

Regioni 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 Europa (Milano)	eu-south-1
Regione Europa (Spagna)	eu-south-2
Regione Europa (Zurigo)	eu-central-2
Regione di Israele (Tel Aviv)	il-central-1
Regione Medio Oriente (Bahrein)	me-south-1
Regione Medio Oriente (EAU)	me-central-1

Per informazioni sull'attivazione di una regione opt-in, vedere [Gestione delle AWS regioni](#) in Riferimenti generali di Amazon Web Services

Quando si utilizza Amazon SNS per recapitare messaggi dalle regioni di accesso alle regioni abilitate per impostazione predefinita, è necessario modificare le policy delle risorse create per la coda. Sostituire l'entità `sns.amazonaws.com` con `sns.<opt-in-region>.amazonaws.com`. Ad esempio:

- Ad esempio, se si desidera sottoscrivere una coda Amazon SQS negli Stati Uniti orientali (Virginia settentrionale) a un argomento Amazon SNS in Asia Pacifico (Hong Kong), modificare il principale nella policy della coda in `sns.ap-east-1.amazonaws.com`. Le regioni di attivazione includono tutte le regioni lanciate dopo il 20 marzo 2019 e includono Asia Pacifico (Hong Kong), Asia Pacifico (Giacarta) Medio Oriente (Bahrein), Europa (Milano) e Africa (Città del Capo). Le regioni lanciate prima del 20 marzo 2019 sono abilitate per impostazione predefinita.

Supporto per la consegna tra regioni ad Amazon SQS

Tipo di consegna tra regioni	Supportato/Non supportato	
Regione abilitata per impostazione predefinita per la regione di attivazione	Supportato utilizzando <code>sns.queue-region.amazonaws.com</code> nel principale di servizio per la coda	
Regione di attivazione per la regione abilitata per impostazione predefinita	Supportato utilizzando <code>sns.topic-region.amazonaws.com</code> nel principale di servizio per la coda	
Regione di attivazione a regione di attivazione	Supportato utilizzando <code>sns.queue-region.amazonaws.com</code> nel principale di servizio per la coda	

Di seguito è riportato un esempio di dichiarazione sulla politica di accesso che consente a un argomento Amazon SNS in una regione opzionale (af-south-1) di effettuare la consegna a una coda Amazon SQS in una regione (us-east-1). `enabled-by-default` Contiene la configurazione del principale di servizio regionalizzato necessario nel percorso `Statement/Principal/Service`.

JSON

```
{
  "Version": "2012-10-17",
  "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-
west-1:111111111111:destination_queue_name",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:sns:af-
south-1:111111111111:source_topic_name"
        }
    }
}
]
}

```

- Per sottoscrivere una AWS Lambda funzione negli Stati Uniti orientali (Virginia settentrionale) a un argomento di Amazon SNS in Asia Pacifico (Hong Kong), modifica il principio nella politica AWS Lambda della funzione in `sns.ap-east-1.amazonaws.com`. Le regioni di attivazione includono tutte le regioni lanciate dopo il 20 marzo 2019 e includono Asia Pacifico (Hong Kong), Asia Pacific (Giacarta) Medio Oriente (Bahrein), Europa (Milano) e Africa (Città del Capo). Le regioni lanciate prima del 20 marzo 2019 sono abilitate per impostazione predefinita.

Supporto per la consegna in più regioni a AWS Lambda

Tipo di consegna tra regioni	Supportato/Non supportato	
Regione abilitata per impostazione predefinita per la regione di attivazione	Supportato utilizzando <code>sns.lambda-function-region</code> .amazonaws.com nel principale di servizio per la funzione Lambda	
Regione di attivazione per la regione abilitata per impostazione predefinita	Supportato utilizzando <code>sns.topic-region</code> .amazonaws.com nel principale di servizio per la funzione Lambda	
Regione di attivazione a regione di attivazione	Supportato utilizzando <code>sns.lambda-function-region</code> .amazonaws.com nel principale di servizio per la funzione Lambda	

Stato di consegna dei messaggi Amazon SNS

Amazon SNS fornisce supporto per la registrazione dello stato di consegna dei messaggi di notifica inviati ad argomenti con i seguenti endpoint Amazon SNS:

- Amazon Data Firehose
- Amazon Simple Queue Service
- AWS Lambda
- HTTPS
- Endpoint applicazione piattaforma

I log dello stato della spedizione vengono inviati ad Amazon CloudWatch Logs, fornendo informazioni dettagliate sulle operazioni di consegna dei messaggi. Questi log ti aiutano a:

- Determinare se un messaggio è stato recapitato correttamente a un endpoint.
- Identifica la risposta dall'endpoint ad Amazon SNS.
- Misura il tempo di permanenza dei messaggi (tempo tra il timestamp di pubblicazione e il trasferimento all'endpoint).

È possibile configurare la registrazione dello stato di consegna utilizzando l'API Console di gestione AWS,, AWS SDKs Query o. AWS CloudFormation

Prerequisiti per la registrazione dello stato della consegna

Questo argomento descrive le autorizzazioni IAM necessarie per consentire ad Amazon SNS di scrivere i log di consegna e spiega la convenzione di denominazione dei gruppi di log CloudWatch predefinita. Ciò garantisce la configurazione e l'accesso corretti per monitorare e analizzare i log di consegna dei messaggi nei log. CloudWatch

Autorizzazioni IAM richieste

Il ruolo IAM associato alla registrazione dello stato della consegna deve includere le seguenti autorizzazioni per consentire ad Amazon SNS di scrivere nei registri. CloudWatch Puoi utilizzare un ruolo esistente con queste autorizzazioni o creare un nuovo ruolo durante la configurazione.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

Convenzione di denominazione dei gruppi di log

Per impostazione predefinita, Amazon SNS crea gruppi di log per i CloudWatch log dello stato di consegna utilizzando la seguente convenzione di denominazione. I flussi di log all'interno di questo gruppo corrispondono ai protocolli degli endpoint (ad esempio, Lambda, Amazon SQS). Assicurati di disporre delle autorizzazioni per visualizzare questi log nella console Logs. CloudWatch

```
sns/<region>/<account-id>/<topic-name>
```

Configurazione della registrazione dello stato di consegna utilizzando il Console di gestione AWS

Questo argomento spiega come abilitare la registrazione dello stato di consegna dei messaggi per gli argomenti di Amazon SNS, tra cui la configurazione delle impostazioni di registrazione, l'assegnazione di ruoli IAM e la verifica CloudWatch che Logs acquisisca i log di consegna per il monitoraggio e la risoluzione dei problemi.

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, scegliere Argomenti.
3. Seleziona l'argomento desiderato, quindi scegli Modifica.

4. Espandi la sezione Registrazione dello stato della consegna.
5. Scegli il protocollo per il quale desideri abilitare la registrazione (ad esempio, HTTP, Lambda, Amazon SQS).
6. Inserisci la frequenza di campionamento di successo, che è la percentuale di messaggi riusciti per i quali desideri ricevere i log. CloudWatch
7. Nella sezione Ruoli IAM, devi configurare i ruoli sia per la registrazione degli eventi riusciti che per quelli non riusciti:
 - Usa un ruolo di servizio esistente: seleziona un ruolo IAM esistente con le autorizzazioni richieste per Amazon SNS su cui scrivere i log. CloudWatch
 - Crea un nuovo ruolo di servizio: scegli Crea nuovi ruoli per definire i ruoli IAM per consegne riuscite e non riuscite nella console IAM. Per i dettagli sulle autorizzazioni, consulta [Prerequisiti per la registrazione dello stato della consegna](#).
8. Scegli Save changes (Salva modifiche).

Dopo aver abilitato la registrazione, è possibile visualizzare e analizzare CloudWatch i registri contenenti lo stato di consegna dei messaggi. [Per ulteriori informazioni sull'utilizzo CloudWatch, consulta la documentazione. CloudWatch](#)

Verifica della configurazione del registro

1. Accedi alla console CloudWatch Logs.
2. Individua il gruppo di log denominato `sns/<region>/<account-id>/<topic-name>`.
3. Assicurati che esistano flussi di log per il protocollo endpoint configurato.
4. Invia un messaggio di prova al tuo argomento e conferma che vengano visualizzate le voci di registro, che indicano che le consegne sono riuscite o non riuscite.

Configurazione della registrazione dello stato della consegna utilizzando il AWS SDKs

AWS SDKs Forniscono APIs in diverse lingue per impostare gli attributi degli argomenti per la registrazione dello stato di consegna dei messaggi. Ad esempio, utilizza l'[SetTopicAttributes](#) API per configurare:

- **LambdaSuccessFeedbackRoleArn**— Ruolo IAM per una corretta consegna dei messaggi agli endpoint Lambda.
- **LambdaSuccessFeedbackSampleRate**— Frequenza di campionamento dei messaggi inviati con successo agli endpoint Lambda.
- **LambdaFailureFeedbackRoleArn**— Ruolo IAM per la mancata consegna dei messaggi agli endpoint Lambda.

Comando di esempio AWS CLI

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name LambdaSuccessFeedbackRoleArn \  
  --attribute-value arn:aws:iam::123456789012:role/MyFeedbackRole
```

Attributi di argomento

Utilizza i seguenti valori dei nomi degli attributi dell'argomento per lo stato di recapito dei messaggi:

HTTP

- **HTTPSuccessFeedbackRoleArn**— Stato di recapito dei messaggi riuscito per un argomento Amazon SNS sottoscritto a un endpoint HTTP.
- **HTTPSuccessFeedbackSampleRate**— Percentuale di messaggi riusciti da campionare per un argomento Amazon SNS sottoscritto a un endpoint HTTP.
- **HTTPFailureFeedbackRoleArn**— Stato di recapito dei messaggi non riuscito per un argomento Amazon SNS sottoscritto a un endpoint HTTP.

Amazon Data Firehose

- **FirehoseSuccessFeedbackRoleArn**— Stato di recapito dei messaggi riuscito per un argomento di Amazon SNS sottoscritto a un endpoint Amazon Data Firehose.
- **FirehoseSuccessFeedbackSampleRate**— Percentuale di messaggi riusciti da campionare per un argomento Amazon SNS sottoscritto a un endpoint Amazon Data Firehose.
- **FirehoseFailureFeedbackRoleArn**— Stato di recapito dei messaggi non riuscito per un argomento di Amazon SNS sottoscritto a un endpoint Amazon Data Firehose.

AWS Lambda

- **LambdaSuccessFeedbackRoleArn**— Stato di recapito dei messaggi riuscito per un argomento di Amazon SNS sottoscritto a un endpoint Lambda.
- **LambdaSuccessFeedbackSampleRate**— Percentuale di messaggi riusciti da campionare per un argomento di Amazon SNS sottoscritto a un endpoint Lambda.
- **LambdaFailureFeedbackRoleArn**— Stato di recapito dei messaggi non riuscito per un argomento di Amazon SNS sottoscritto a un endpoint Lambda.

Endpoint applicativi della piattaforma

- **ApplicationSuccessFeedbackRoleArn**— Stato di recapito dei messaggi riuscito per un argomento di Amazon SNS sottoscritto a un AWS endpoint applicativo.
- **ApplicationSuccessFeedbackSampleRate**— Percentuale di messaggi riusciti da campionare per un argomento di Amazon SNS sottoscritto a un AWS endpoint applicativo.
- **ApplicationFailureFeedbackRoleArn**— Stato di recapito dei messaggi non riuscito per un argomento di Amazon SNS sottoscritto a un AWS endpoint dell'applicazione.

Note

Inoltre, puoi configurare gli attributi dell'applicazione per registrare lo stato di consegna direttamente nei servizi di notifica push. Per ulteriori informazioni, consulta la pagina sull'[utilizzo degli attributi di applicazione di Amazon SNS per lo stato di consegna dei messaggi](#).

Amazon SQS

- **SQSSuccessFeedbackRoleArn**— Stato di recapito dei messaggi riuscito per un argomento di Amazon SNS sottoscritto a un endpoint Amazon SQS.
- **SQSSuccessFeedbackSampleRate**— Percentuale di messaggi riusciti da campionare per un argomento di Amazon SNS sottoscritto a un endpoint Amazon SQS.
- **SQSFailureFeedbackRoleArn**— Stato di recapito dei messaggi non riuscito per un argomento di Amazon SNS sottoscritto a un endpoint Amazon SQS.

I log per gli endpoint delle applicazioni della piattaforma vengono scritti nello stesso CloudWatch gruppo Logs degli altri endpoint.

Note

<ENDPOINT>FailureFeedbackRoleArn Gli attributi <ENDPOINT>SuccessFeedbackRoleArn and vengono utilizzati per consentire ad Amazon SNS l'accesso in scrittura per utilizzare CloudWatch i log per tuo conto. L'attributo <ENDPOINT>SuccessFeedbackSampleRate consente di specificare la percentuale della frequenza di campionamento (0-100) dei messaggi consegnati. Dopo aver configurato l'<ENDPOINT>FailureFeedbackRoleArn attributo, tutte le consegne di messaggi non riuscite generano log. CloudWatch

AWS Esempi SDK per configurare gli attributi degli argomenti

Gli esempi di codice seguenti mostrano come utilizzare SetTopicAttributes.

CLI

AWS CLI

Impostazione di un attributo per un argomento

Nell'esempio set-topic-attributes seguente vengono impostati gli attributi DisplayName per l'argomento specificato.

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name DisplayName \  
  --attribute-value MyTopicDisplayName
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [SetTopicAttributes AWS CLI](#) Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.
            """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

    }

    String attribute = args[0];
    String topicArn = args[1];
    String value = args[2];

    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    setTopAttr(snsClient, attribute, topicArn, value);
    snsClient.close();
}

public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
    try {
        SetTopicAttributesRequest request =
SetTopicAttributesRequest.builder()
            .attributeName(attribute)
            .attributeValue(value)
            .topicArn(topicArn)
            .build();

        SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
        System.out.println(
            "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
"\n\nTopic " + request.topicArn()
            + " updated " + request.attributeName() + " to " +
request.attributeValue());

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

```

- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    })
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
```



```
//    extendedRequestId: undefined,  
//    cfId: undefined,  
//    attempts: 1,  
//    totalRetryDelay: 0  
//  }  
// }  
return response;  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun setTopAttr(  
    attribute: String?,  
    topicArnVal: String?,  
    value: String?,  
) {  
    val request =  
        SetTopicAttributesRequest {  
            attributeName = attribute  
            attributeValue = value  
            topicArn = topicArnVal  
        }  
  
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->  
        snsClient.setTopicAttributes(request)  
        println("Topic ${request.topicArn} was updated.")  
    }  
}
```

- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSClient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
}
```

```

} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}

```

- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK per PHP API Reference.

Ruby

SDK per Ruby

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
      attribute_name: policy_name,
      attribute_value: policy
    })
  end
end

```

```

    @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 per Ruby](#).
- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK per Ruby API Reference.

SAP ABAP

SDK per 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 i dettagli sulle API, [SetTopicAttributes](#) consulta AWS SDK for SAP ABAP API reference.

Configurazione della registrazione dello stato di consegna tramite CloudFormation

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

Assicurati che i ruoli IAM a cui si fa riferimento `SuccessFeedbackRoleArn` e `FailureFeedbackRoleArn` dispongano delle autorizzazioni Logs richieste. CloudWatch

JSON

```
"Resources": {
  "MySNSTopic" : {
    "Type" : "AWS::SNS::Topic",
    "Properties" : {
      "TopicName" : "TestTopic",
      "DisplayName" : "TEST",
      "SignatureVersion" : "2",
      "DeliveryStatusLogging" : [{
        "Protocol": "sqs",
        "SuccessFeedbackSampleRate": "45",
        "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
        "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
      }]
    }
  }
}
```

YAML

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName: TestTopic
      DisplayName: TEST
      SignatureVersion: 2
      DeliveryStatusLogging:
        - Protocol: sqs
          SuccessFeedbackSampleRate: 45
          SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
          FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Tentativi di consegna dei messaggi di Amazon SNS

Amazon SNS definisce una policy di consegna per ogni protocollo di consegna. La policy di consegna definisce il modo in cui Amazon SNS esegue nuovi tentativi di consegna dei messaggi quando si verificano errori sul lato server (quando il sistema che ospita l'endpoint sottoscritto diventa non disponibile). Quando la politica di spedizione è esaurita, Amazon SNS smette di tentare nuovamente la consegna e scarnerà il messaggio, a meno che all'abbonamento non sia allegata una coda di lettere non recapitate. Per ulteriori informazioni, consulta [Code di lettere non ricevute di Amazon SNS](#).

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 tentativi di consegna. Per ulteriori informazioni, vedere il post [Backoff esponenziale e Jitter](#) in Blog architettura AWS .
- Il tempo totale di riprova delle policy per un HTTP/S endpoint non può essere superiore a 3.600 secondi. Questo è un limite fissato che non può essere modificato.

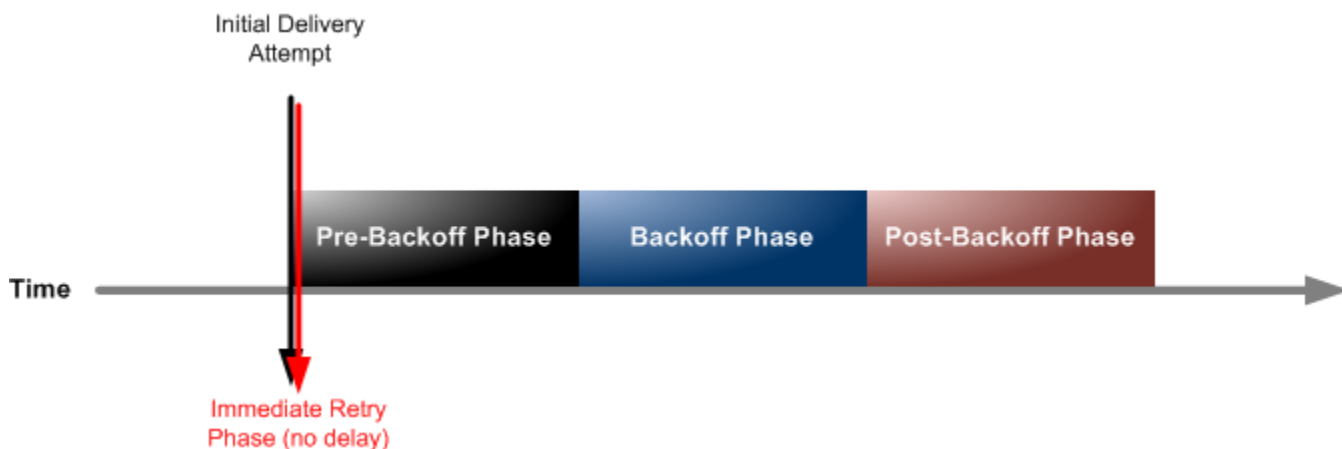
Tipo di endpoint	Protocolli di consegna	Fase dei nuovi tentativi immediati (nessun ritardo)	Fase di prebackoff	Fase di backoff	Fase di postbackoff	Totale tentativi
AWS endpoint gestiti	¹ AWS Lambda	3 volte, senza ritardo	2 volte, 1 secondo di distanza	10 volte, con backoff esponenziale, da 1 secondo 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	0 volte, senza ritardo	2 volte, 10 secondi di distanza	10 volte, con backoff esponenziale, da 10 secondi a 600 secondi (10 minuti)	38 volte, 600 secondi (10 minuti) di distanza	50 tentativi, oltre 6 ore
	SMS					
	Push per dispositivi mobili					

¹ Per gli errori di limitazione con il protocollo Firehose, Amazon SNS utilizza la stessa politica di distribuzione degli endpoint gestiti dai clienti.

Fasi della policy di consegna

Il diagramma seguente mostra le fasi di una policy di consegna.



Ogni policy di consegna si compone di quattro fasi.

1. Fase di nuovo tentativo immediato (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 pre-backoff: questa fase segue la fase di tentativo immediato. Amazon SNS utilizza questa fase per effettuare una serie di tentativi prima di applicare una funzione di backoff. Questa fase specifica il numero di tentativi e la quantità di ritardo tra di loro.
3. Fase di backoff: questa fase controlla il ritardo tra i 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 post-backoff: 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 definire in che modo Amazon SNS ritenta la consegna dei messaggi agli HTTP/S endpoint utilizzando una politica di consegna con quattro fasi: senza ritardo, pre-backoff, backoff e post-backoff. Questa policy ti consente di ignorare le impostazioni predefinite per i nuovi tentativi e di personalizzarle in base alla capacità del tuo server HTTP.

Puoi definire la tua politica HTTP/S di consegna come oggetto JSON a livello di argomento o di abbonamento:

- Politica a livello di argomento: si applica a tutti gli HTTP/S abbonamenti collegati all'argomento. Utilizza l'azione [CreateTopic](#) [SetTopicAttributes](#) API per impostare questa politica.
- Politica a livello di abbonamento: si applica solo a un abbonamento specifico. Utilizza l'azione [Subscribe](#) [SetSubscriptionAttributes](#) API per impostare questa politica.

In alternativa, puoi anche utilizzare la [AWS::SNS::Subscription](#) risorsa nei tuoi CloudFormation modelli.

È necessario personalizzare la politica di distribuzione in base alla capacità del HTTP/S server:

- Server singolo per tutti gli abbonamenti: se tutti gli HTTP/S abbonamenti di un argomento utilizzano lo stesso server, imposta la politica di distribuzione come attributo dell'argomento per garantire la coerenza tra tutti gli abbonamenti.

- Server diversi per gli abbonamenti: se gli abbonamenti sono destinati a server diversi, crea una politica di distribuzione unica per ogni abbonamento, personalizzata in base alla capacità del server specifico.

Puoi anche impostare l'`Content-Type` intestazione nella politica di richiesta per specificare il tipo di supporto della notifica. Per impostazione predefinita, Amazon SNS invia tutte le notifiche agli HTTP/S endpoint con il tipo di contenuto impostato su `text/plain; charset=UTF-8`. Tuttavia, puoi ignorare questa impostazione predefinita utilizzando il [headerContent-Type](#) campo nella politica di richiesta.

Il seguente oggetto JSON definisce una politica di consegna con nuovi tentativi strutturati in quattro fasi:

1. Fase senza ritardo: riprova immediatamente 3 volte.
2. Fase pre-backoff: riprova 2 volte con un intervallo di 1 secondo.
3. Fase di backoff: riprova 10 volte con ritardi esponenziali compresi tra 1 e 60 secondi.
4. Fase post-backoff: riprova 35 volte con un intervallo fisso di 60 secondi.

Amazon SNS effettua un totale di 50 tentativi di recapito di un messaggio prima di eliminarlo. Per conservare i messaggi che non possono essere recapitati dopo tutti i tentativi, configura l'abbonamento per spostare i messaggi non recapitabili in una coda di lettere non recapitate (DLQ). Per ulteriori informazioni, consulta [Code di lettere non ricevute di Amazon SNS](#).

Amazon SNS considera risolvibili tutti gli errori 5XX e 429 (troppe richieste inviate) errori. Questi errori sono soggetti alla politica di consegna. Tutti gli altri errori sono considerati errori permanenti e non verranno tentati nuovi tentativi.

Note

Questa politica di consegna utilizza la `maxReceivesPerSecond` proprietà per limitare il traffico di consegna a una media di 10 messaggi al secondo per abbonamento. Sebbene questo meccanismo aiuti a evitare che l'HTTP/S endpoint venga sopraffatto da un traffico elevato, è progettato per mantenere un tasso di consegna medio e non impone un limite rigoroso. Occasionalmente possono verificarsi picchi di traffico relativi alle consegne superiori al limite specificato, soprattutto se la frequenza di pubblicazione è notevolmente superiore al limite di limitazione.

Quando il traffico di pubblicazione (in entrata) supera la velocità di recapito (in uscita), può verificarsi un arretrato dei messaggi e una maggiore latenza di consegna. Per evitare tali problemi, assicurati che il `maxReceivesPerSecond` valore sia in linea con i requisiti di capacità e carico di lavoro HTTP/S del server.

Il seguente esempio di politica di distribuzione sostituisce il tipo di contenuto predefinito per la notifica HTTP/S. `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 politica di consegna è composta da una politica di riprova, una politica di limitazione e una politica di richiesta. In totale, ci sono 9 attributi in una politica di consegna.

Policy	Description	Vincolo
<code>minDelayTarget</code>	Il ritardo minimo per un nuovo tentativo. Unità: secondi	1 al massimo ritardo Di default: 20
<code>maxDelayTarget</code>	Il ritardo massimo per un nuovo tentativo.	Ritardo minimo su 3.600 Di default: 20

Policy	Description	Vincolo
	Unità: secondi	
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 default: 0
numMinDelayRetries	Il numero di tentativi nella fase di pre-backoff, con il ritardo minimo specificato tra essi.	Uguale o maggiore di 0 Di default: 0
numMaxDelayRetries	Il numero di tentativi nella fase post-backoff, con il massimo ritardo tra essi.	Uguale o maggiore di 0 Di default: 0
backoffFunction	Il modello per il backoff tra i nuovi tentativi.	Una delle quattro opzioni: • Aritmetica • Esponenziale • Geometrica • Lineare Di default: lineare
maxReceivesPerSecond	Il numero medio massimo di recapiti di messaggi al secondo, per abbonamento.	Uguale o maggiore di 1 Impostazione predefinita: nessuna limitazione (nessun limite alla velocità di consegna)

Policy	Description	Vincolo
headerContentType	Il tipo di contenuto della notifica inviato agli endpoint HTTP/S.	Se la policy di richiesta non è definita, il tipo di contenuto è preimpostato su <code>text/plain; charset=UTF-8</code>. 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'istanza supportati sono <code>application/json</code> e <code>text/plain</code>. Quando la distribuzione di messaggi non elaborati è abilitata per una sottoscrizione, sono supportati i seguenti tipi di contenuto: • <code>text/css</code> • <code>text/csv</code> • <code>text/html</code> • <code>text/plain</code> • <code>text/xml</code> • <code>application/atom+xml</code> • <code>application/json</code> • <code>application/octet-stream</code> • <code>application/soap+xml</code> • <code>applicazione/ x-www-form-urlencoded</code>

Policy	Description	Vincolo
		• application/xhtml+xml • application/xml

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

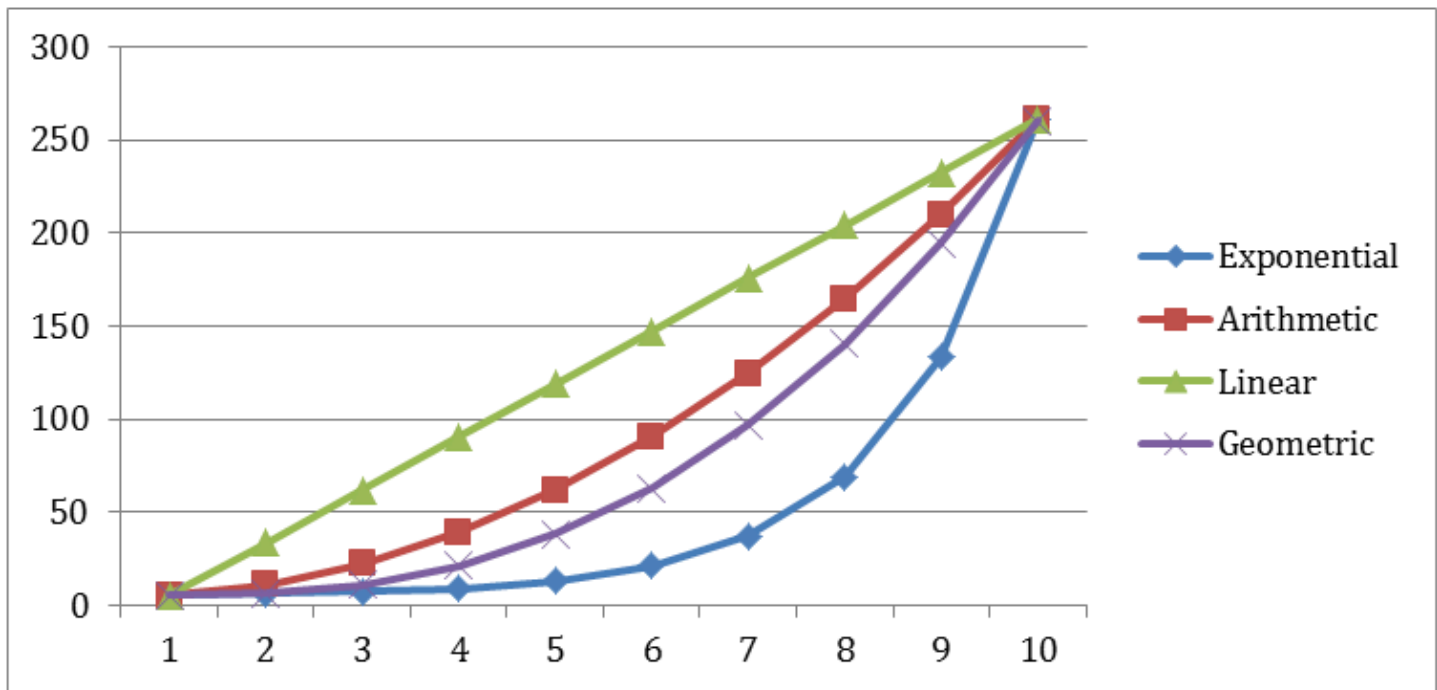
```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

È possibile controllare la frequenza dei nuovi tentativi durante la fase di backoff utilizzando tre parametri:

- **minDelayTarget**— Imposta il ritardo per il primo tentativo di nuovo tentativo nella fase di backoff.
- **maxDelayTarget**— Imposta il ritardo per l'ultimo tentativo di nuovo tentativo nella fase di backoff.
- **backoffFunction**— Determina l'algoritmo utilizzato da Amazon SNS per calcolare i ritardi di tutti i tentativi tra il primo e l'ultimo tentativo. Puoi scegliere tra quattro funzioni di retry-backoff disponibili.

Il diagramma seguente illustra come le diverse funzioni di retry backoff influiscano sui ritardi tra i tentativi durante la fase di backoff. La politica di consegna utilizzata per questo esempio include le seguenti impostazioni: 10 tentativi totali, un ritardo minimo di 5 secondi e un ritardo massimo di 260 secondi.

- L'asse verticale mostra il ritardo (in secondi) per ogni nuovo tentativo.
- L'asse orizzontale rappresenta la sequenza di tentativi, che va dal primo al decimo tentativo.



Code di lettere non ricevute di Amazon SNS

Una coda DLQ è una coda Amazon SQS a cui un abbonamento Amazon SNS può mirare per i messaggi che non possono essere recapitati correttamente agli abbonati. I messaggi che non possono essere recapitati a causa di errori client o errori server vengono mantenuti nella coda DLQ per ulteriori analisi o elaborazione. Per ulteriori informazioni, consultare [Configurazione di una coda Amazon SNS dead-letter per una sottoscrizione](#) e [Tentativi di consegna dei messaggi di Amazon SNS](#).

Note

- L'abbonamento Amazon SNS e la coda Amazon SQS devono appartenere allo stesso account e alla AWS stessa regione.
- Per un [Argomento FIFO](#), puoi utilizzare una coda di Amazon SQS come una coda DLQ per la sottoscrizione ad Amazon SNS. Le sottoscrizioni agli argomenti FIFO utilizzano le code FIFO e le sottoscrizioni agli argomenti standard utilizzano le code standard.
- Per utilizzare una coda Amazon SQS crittografata come coda di lettere non scritte, devi utilizzare un KMS personalizzato con una policy chiave che garantisca al servizio Amazon SNS l'accesso principale alle azioni API. AWS KMS Per ulteriori informazioni, consulta [Protezione dei dati di Amazon SNS con crittografia lato server](#) in questa guida e [Protezione](#)

[dei dati Amazon SQS con la crittografia lato server \(SSE\) e AWS KMS](#) nella Guida per sviluppatori Amazon Simple Queue Service.

Perché le consegne dei messaggi non riescono?

In generale, la consegna dei messaggi non riesce quando Amazon SNS non può accedere a un endpoint sottoscritto a causa di un errore lato client o lato server. Quando Amazon SNS riceve un errore sul lato client o continua a ricevere un errore sul lato server per un messaggio che supera il numero di tentativi specificato dalla policy di ripetizione dei tentativi corrispondente, Amazon SNS ignora il messaggio, a meno che non sia allegata una coda DLQ all'abbonamento. Le consegne non riuscite non modificano lo stato delle sottoscrizioni. Per ulteriori informazioni, consulta [Tentativi di consegna dei messaggi di Amazon SNS](#).

Errori lato client

Gli errori lato client possono verificarsi quando Amazon SNS dispone di metadati di sottoscrizione obsoleti. Questi errori si verificano in genere quando un proprietario elimina l'endpoint (ad esempio, una funzione Lambda sottoscritta a un argomento Amazon SNS) o quando un proprietario modifica la policy associata all'endpoint sottoscritto in modo da impedire ad Amazon SNS la policy di consegna dei messaggi all'endpoint. Amazon SNS non riprova il recapito dei messaggi che non riesce a causa di un errore sul lato client.

Errori lato server

Gli errori lato server possono verificarsi quando il sistema responsabile dell'endpoint sottoscritto diventa non disponibile o restituisce un'eccezione che indica che non è in grado di elaborare una richiesta valida da Amazon SNS. Quando si verificano errori lato server, Amazon SNS ripropone le consegne non riuscite utilizzando una funzione di backoff lineare o esponenziale. Per gli errori lato server causati da endpoint AWS gestiti supportati da Amazon SQS o Amazon AWS Lambda SNS, ritenta la consegna fino a 100.015 volte, in 23 giorni.

Gli endpoint gestiti dal cliente (ad esempio HTTP, SMTP, SMS o push mobile) possono anche causare errori lato server. Amazon SNS tenta di eseguire nuovamente la consegna anche a questi tipi di endpoint. Mentre gli endpoint HTTP supportano le policy di nuovi tentativi definite dal cliente, Amazon SNS imposta una policy di nuovi tentativi di consegna interna su 50 volte nell'arco di 6 ore per gli endpoint push SMTP, SMS e dispositivi mobili.

Come funzionano le code DLQ?

Una coda dead-letter è allegata a una sottoscrizione Amazon SNS (anziché a un argomento) perché la consegna dei messaggi avviene a livello di sottoscrizione. In questo modo è possibile identificare più facilmente l'endpoint di destinazione originale per ogni messaggio.

Una coda dead-letter associata a una sottoscrizione Amazon SNS è una coda Amazon SQS ordinaria. Per ulteriori informazioni sul periodo di conservazione dei messaggi, consulta [Quote correlate ai messaggi](#) nella Guida per sviluppatori Amazon Simple Queue Service. È possibile modificare il periodo di conservazione dei messaggi utilizzando l'operazione API Amazon SQS [SetQueueAttributes](#). Per rendere le applicazioni più resilienti, è consigliabile impostare il periodo massimo di conservazione per le code dead-letter a 14 giorni.

Come vengono spostati i messaggi in una coda DLQ?

I messaggi vengono spostati in una coda dead-letter utilizzando una policy di redrive. Una policy di redrive è un oggetto JSON che fa riferimento all'ARN della coda dead-letter. L'attributo `deadLetterTargetArn` specifica l'ARN. L'ARN deve puntare a una coda Amazon SQS nella Account AWS stessa regione dell'abbonamento Amazon SNS. Per ulteriori informazioni, consulta [Configurazione di una coda Amazon SNS dead-letter per una sottoscrizione](#).

L'oggetto JSON seguente è una policy di redrive di esempio, allegato a una sottoscrizione SNS.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Come posso spostare i messaggi fuori da una coda DLQ?

È possibile spostare i messaggi fuori da una coda dead-letter in due modi:

- Evitare di scrivere logica consumer Amazon SQS - Impostare la coda dead-letter come origine evento sulla funzione Lambda per drenare la coda dead-letter.
- Scrivi la logica consumer di Amazon SQS: utilizza l'API Amazon SQS AWS, l'SDK AWS CLI o per scrivere una logica consumer personalizzata per il polling, l'elaborazione e l'eliminazione dei messaggi nella coda delle lettere non scritte.

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 code Amazon SQS emettono CloudWatch parametri a intervalli di un minuto. Per ulteriori informazioni, consulta i [CloudWatch parametri disponibili per Amazon SQS nella Amazon Simple Queue Service Developer Guide](#). Tutti gli abbonamenti Amazon SNS con code di lettera morta emettono anche parametri. CloudWatch Per ulteriori informazioni, consulta [Monitoraggio degli argomenti di Amazon SNS 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 un DLQ a seguito di tentativi di elaborazione non riusciti. Utilizza invece la `ApproximateNumberOfMessagesVisible` metrica, che acquisisce tutti i messaggi attualmente disponibili nel DLQ, 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 tue aspettative e al traffico DLQ).
3. Specificare un argomento di Amazon SNS per ricevere una notifica quando scatta l'allarme. In questo argomento Amazon SNS è possibile inviare la notifica di allarme a qualsiasi tipo di endpoint (ad esempio un indirizzo e-mail, un numero di telefono o un'app per il cercapersone mobile).

Puoi utilizzare CloudWatch Logs per esaminare le eccezioni che causano il fallimento delle consegne di Amazon SNS 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 dell'app mobile Amazon SNS](#).

Configurazione di una coda Amazon SNS dead-letter per una sottoscrizione

Una coda DLQ è una coda Amazon SQS a cui un abbonamento Amazon SNS può mirare per i messaggi che non possono essere recapitati correttamente agli abbonati. I messaggi che non possono essere recapitati a causa di errori client o errori server vengono mantenuti nella coda DLQ per ulteriori analisi o elaborazione. Per ulteriori informazioni, consultare [Code di lettere non ricevute di Amazon SNS](#) e [Tentativi di consegna dei messaggi di Amazon SNS](#).

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

Note

Per un [Argomento FIFO](#), puoi utilizzare una coda di Amazon SQS come una coda DLQ per la sottoscrizione ad Amazon SNS. Le sottoscrizioni agli argomenti FIFO utilizzano le code FIFO e le sottoscrizioni agli argomenti standard utilizzano le code standard.

Prerequisiti

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

1. [Creare un argomento Amazon SNS](#), denominato MyTopic.
2. [Crear una coda Amazon SQS](#) denominata MyEndpoint, da utilizzare come endpoint per l'abbonamento Amazon SNS.
3. (Salta per CloudFormation) [Iscriviti alla coda per accedere all'argomento](#).
4. [Creare un'altra coda Amazon SQS](#) denominata MyDeadLetterQueue, da utilizzare come coda dead-letter per la sottoscrizione Amazon SNS.
5. Per concedere l'accesso Amazon SNS principale all'azione Amazon SQS API, impostare la policy della coda seguente per MyDeadLetterQueue.

JSON

```
{
  "Version": "2012-10-17",
  "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"
        }
      }
    }
  ]
}
```

```
}  
}  
}  
]  
}
```

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

Prima di iniziare questo tutorial, completare i [prerequisiti](#) descritti di seguito.

1. Accedere alla [console Amazon SQS](#).
2. [Creare una coda Amazon SQS](#) o utilizzare una coda esistente e prendere nota dell'ARN della coda nella scheda Details (Dettagli) della coda, ad esempio:

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

3. Accedi alla [console Amazon SNS](#).
4. Nel riquadro di navigazione, scegli Sottoscrizioni.
5. Sulla pagina Subscriptions (Abbonamenti), selezionare una sottoscrizione esistente, quindi scegliere Edit (Modifica).
6. Nella **1234a567-bc89-012d-3e45-6fg7h890123i** pagina Modifica, espandi la sezione Redrive policy (dead-letter queue), quindi procedi come segue:
 - a. Scegli Enabled (Abilitato).
 - b. Specificare l'ARN di una coda Amazon SQS.
7. Seleziona Save changes (Salva modifiche).

La sottoscrizione è configurata per l'utilizzo di una coda dead-letter.

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

Prima di eseguire questo esempio, completare i [prerequisiti](#).

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

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

Java

SDK per Java 1.x

 Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
// attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

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

Prima di iniziare questo tutorial, completare i [prerequisiti](#) descritti di seguito.

1. Installa 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 SNS utilizzando CloudFormation

Prima di iniziare questo tutorial, completare i [prerequisiti](#).

1. Copia il seguente codice JSON in un file denominato `MyDeadLetterQueue.json`.

```
{  
  "Resources": {  
    "mySubscription": {  
      "Type" : "AWS::SNS::Subscription",  
      "Properties" : {  
        "Protocol": "sqs",  
        "Endpoint": "arn:aws:sqs:us-east-2:123456789012:MyEndpoint",  
        "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",  
        "RedrivePolicy": {  
          "deadLetterTargetArn":  
            "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"  
        }  
      }  
    }  
  }  
}
```

2. Accedi alla [console 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).

CloudFormation inizia a creare lo **MyDeadLetterQueue** stack e visualizza lo stato `CREATE_IN_PROGRESS`. Quando il processo è completo, visualizza lo stato `CREATE_COMPLETE`. CloudFormation

Archiviazione, riproduzione e analisi dei messaggi di Amazon SNS

Gli argomenti standard di Amazon SNS supportano l'archiviazione dei messaggi tramite. È possibile inviare notifiche ai flussi di distribuzione di Firehose, che consentono di inviare notifiche alle destinazioni di archiviazione e analisi supportate da Firehose, tra cui Amazon Simple Storage Service (Amazon S3), Amazon Redshift e altre.

Gli argomenti FIFO di Amazon SNS supportano un archivio di messaggi locale e senza codice che consente ai proprietari degli argomenti di archiviare i messaggi pubblicati su un argomento per un massimo di 365 giorni. Per gli argomenti con una `ArchivePolicy` attiva, gli abbonati possono quindi creare una `ReplayPolicy` per recuperare (o riprodurre) i messaggi archiviati su un endpoint sottoscritto. Per ulteriori informazioni su questa funzionalità, consulta [Archiviazione e riproduzione dei messaggi di Amazon SNS per argomenti FIFO](#).

Funzionalità	Argomenti standard	Argomenti FIFO
Archiviazione di messaggi	Flussi di distribuzione da Fanout a Firehose	Archiviazione dei messaggi Amazon SNS per i proprietari di argomenti FIFO
Riproduzione di messaggi	La riproduzione per argomenti standard non è una funzionalità integrata. Molti clienti ne creano una propria in base al proprio archivio di messaggi.	Replay dei messaggi di Amazon SNS per gli abbonati all'argomento FIFO

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

- [Assegnazione di tag all'argomento Amazon SNS](#)

Assegnazione di tag all'argomento Amazon SNS

Amazon SNS supporta l'assegnazione di tag agli argomenti Amazon SNS. 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 cercare o filtrare facilmente migliaia di argomenti. Il tagging ti consente di gestire gli argomenti di Amazon SNS utilizzando Resource AWS Groups. Per ulteriori informazioni sui Resource Groups, consulta la [Guida per l'utente dei Resource Groups AWS](#).

Assegnazione di tag per l'allocazione dei costi

Per organizzare e identificare gli argomenti Amazon SNS per l'allocazione dei costi, puoi aggiungere tag che identificano lo scopo di un argomento. Questo è particolarmente utile in presenza di molti argomenti. È possibile utilizzare i tag di allocazione dei costi per organizzare la AWS fattura in modo che rifletta la propria 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 degli argomenti Amazon SNS, come indicato di seguito:

Risorsa	Chiave	Valore
Argomento 1	Centro costi	43289
	Applicazione	Elaborazione di ordini
Argomento 2	Centro costi	43289

Risorsa	Chiave	Valore
Argomento 3	Applicazione	Elaborazione dei pagamenti
	Centro costi	76585
	Applicazione	Archiviazione

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

Assegnazione di tag per il controllo degli accessi

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

È possibile limitare l'accesso per un'identità IAM. Ad esempio, è possibile limitare l'accesso Publish e PublishBatch a tutti gli argomenti Amazon SNS che includono un tag con la chiave `environment` e il valore `production`, consentendo al contempo l'accesso a tutti gli altri argomenti Amazon SNS. Nell'esempio riportato di seguito, la policy limita la possibilità di pubblicare messaggi su argomenti con taggati con `production`, consentendo al contempo la pubblicazione di messaggi su argomenti taggati con `development`. Per ulteriori informazioni, consulta [Controllo dell'accesso tramite tag](#) nella Guida per l'utente di IAM.

Note

Impostare l'autorizzazione IAM per Publish imposta l'autorizzazione sia per Publish che per PublishBatch.

JSON

```
{  
  "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 argomenti di Amazon SNS (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. Aniché dover cercare manualmente questi argomenti, è possibile interrogare tutti gli argomenti con un determinato tag:

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

```
import com.amazonaws.services.resourcegroups.model.SearchResourcesResult;

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

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

        // Query all resources with certain tags from ResourceGroups
        SearchResourcesResult result = awsResourceGroups.searchResources(
            new SearchResourcesRequest().withResourceQuery(
                new ResourceQuery()
                    .withType(QueryType.TAG_FILTERS_1_0)
                    .withQuery(QUERY)
            ));

        System.out.println("SNS Topics with certain tags are " +
            result.getResourceIdentifiers());
    }
}
```

Configurare i tag dell'argomento Amazon SNS

Questo argomento spiega come configurare i tag per un [argomento Amazon SNS](#) utilizzando Console di gestione AWS, un AWS SDK o il AWS CLI

 Important

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

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

1. Accedi alla [console Amazon SNS](#).

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

Vengono elencati i tag aggiunti all'argomento.

5. Modificare i tag dell'argomento:
 - Per aggiungere un tag, scegliere Add tag (Aggiungi tag) e specificare Key (Chiave) e Value (Valore) (opzionale),
 - Per rimuovere un tag, scegliere Remove tag (Rimuovi tag) accanto a una coppia chiave-valore.
6. Scegli Save changes (Salva modifiche).

Aggiungere tag a un argomento utilizzando un SDK AWS

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

Gli esempi di codice seguenti mostrano come utilizzare TagResource.

CLI

AWS CLI

Aggiungere un tag a un argomento

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

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

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta AWS CLI Command [TagResource](#)Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

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

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String topicArn = args[0];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();

addTopicTags(snsClient, topicArn);
snsClient.close();
}

public static void addTopicTags(SnsClient snsClient, String topicArn) {
    try {
        Tag tag = Tag.builder()
            .key("Team")
            .value("Development")
            .build();

        Tag tag2 = Tag.builder()
            .key("Environment")
            .value("Gamma")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag);
        tagList.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .resourceArn(topicArn)
            .tags(tagList)
            .build();

        snsClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to " + topicArn);

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

- Per i dettagli sull'API, [TagResource](#) consulta AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

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

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

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

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

- Per i dettagli sull'API, [TagResource](#) consulta AWS SDK for Kotlin API reference.

Gestione dei tag con le operazioni dell'API Amazon SNS

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

- [ListTagsForResource](#)
- [TagResource](#)
- [UntagResource](#)

Azioni API che supportano ABAC

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

- [AddPermission](#)
- [ConfirmSubscription](#)
- [DeleteTopic](#)
- [GetDataProtectionPolicy](#)
- [GetSubscriptionAttributes](#)
- [GetTopicAttributes](#)
- [ListSubscriptionsByTopic](#)
- [ListTagsForResource](#)
- [Publish](#)
- [PublishBatch](#)
- [PutDataProtectionPolicy](#)
- [RemovePermission](#)
- [SetSubscriptionAttributes](#)
- [SetTopicAttributes](#)
- [Subscribe](#)
- [TagResource](#)
- [Unsubscribe](#)
- [UntagResource](#)

Sorgenti e destinazioni degli eventi Amazon SNS

Amazon SNS si connette Servizi AWS a 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 scalabilità di Amazon EC2 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 Data Firehose per lo streaming di dati verso servizi di storage e analisi. Per la messaggistica A2P, Amazon SNS può inviare notifiche tramite SMS, e-mail e notifiche push ai dispositivi mobili, garantendo che gli utenti o i team ricevano avvisi tempestivi.

Fungendo da 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.

Origini eventi Amazon SNS

Amazon SNS si integra con un'ampia gamma di diverse Servizi AWS categorie, consentendo a questi servizi di pubblicare eventi su argomenti di Amazon SNS. 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 [Argomenti FIFO](#) nel mese di ottobre 2020. Attualmente, la maggior parte AWS dei servizi supporta l'invio di eventi solo ad argomenti standard.

Servizi di analisi

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS analisi come Athena e AWS Data Pipeline Amazon Redshift per fornire notifiche in tempo reale per eventi chiave, 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 operazioni relative ai dati.

Servizio AWS	Vantaggio dell'utilizzo di con Amazon SNS
Amazon Athena - consente di analizzare i dati in Amazon S3 utilizzando SQL standard.	Ricevi notifiche quando i limiti di controllo vengono superati. Per ulteriori informazioni, consulta Impostazione dei limiti per il controllo dell'utilizzo nella Guida per l'utente di Amazon Athena.
AWS Data Pipeline – Automatizza il movimento e la trasformazione dei dati.	Ricevere notifiche sullo stato dei component i della pipeline. Per ulteriori informazioni, consulta la sezione SnsAlarm nella Guida per gli sviluppatori di AWS Data Pipeline .
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 e le notifiche dei dati 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	Vantaggio dell'utilizzo con Amazon SNS
Amazon EventBridge : fornisce un flusso di dati in tempo reale dalle tue applicazioni, applicazioni software-as-a-service (SaaS) e AWS servizi e indirizza tali dati verso destinazioni, tra cui	Ricevi notifiche di eventi. EventBridge Per ulteriori informazioni, consulta EventBridge gli obiettivi di Amazon nella Amazon EventBridge User Guide.

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
Amazon SNS. EventBridge in precedenza si chiamava Events. CloudWatch	
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 Richiama Amazon SNS con Step Functions nella Guida per gli sviluppatori di AWS Step Functions .

Servizi di gestione di costi e fatturazione

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

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

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
Gestione dei costi e fatturazione AWS - fornisce funzioni che consentono di monitorare i costi e pagare la fattura.	Ricevi notifiche di budget, notifiche di modifica del prezzo e avvisi di anomalia. Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente di AWS Billing : • Creazione di un argomento Amazon SNS per le notifiche dei budget • 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 Chime SDK per migliorare i tuoi strumenti di collaborazione all'interno e all'esterno dell'organizzazione.

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

Servizi di elaborazione

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 EC2 Image 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 le tue risorse rimanendo informato sugli aggiornamenti e sulle azioni critiche in corso. Servizi AWS

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
Amazon EC2 Auto Scaling - Consente di avere il numero corretto di istanze Amazon Elastic Compute Cloud (Amazon EC2) disponibili per gestire il carico dell'applicazione.	Ricevi notifiche quando Auto Scaling avvia o interrompe le istanze Amazon EC2 nel gruppo Auto Scaling. Per ulteriori informazioni, consulta l'argomento relativo alla ricezione di notifiche Amazon SNS quando il gruppo Auto Scaling viene ridimensionato nella Guida per l'utente di Amazon EC2 Auto Scaling.

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
EC2 Image Builder : aiuta ad automatizzare la creazione, la gestione e l'implementazione di immagini server personalizzate up-to-date, sicure e preinstallate con software e impostazioni per soddisfare specifici standard IT.	Ricevi notifiche quando le build sono completate. Per ulteriori informazioni, consulta Monitoraggio delle immagini server più recenti nelle pipeline di EC2 Image Builder sul Compute blog AWS .
AWS Elastic Beanstalk – Gestisce automaticamente i dettagli di provisioning della capacità, bilanciamento del carico, dimensionamento e monitoraggio dello stato dell'applicazione.	Ricevi notifiche di eventi importanti che interessano la tua applicazione. Per ulteriori informazioni, consulta Notifiche dell'ambiente Elastic Beanstalk con Amazon SNS nella Guida per gli sviluppatori AWS Elastic Beanstalk .
AWS Lambda - Consente di eseguire il codice senza effettuare il provisioning dei server o senza gestirli.	Ricevere i dati di output della funzione impostando un argomento SNS come una coda di lettere non recapitate Lambda o una destinazione Lambda. Per ulteriori informazioni, consulta Chiamata asincrona nella Guida per gli sviluppatori AWS Lambda .
Amazon Lightsail : aiuta gli sviluppatori a iniziare 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 sviluppatori di Amazon Lightsail.

Servizi di container

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

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

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
Amazon EKS Distribuzione - Consente di creare cluster affidabili e sicuri ovunque vengano distribuite le applicazioni.	Tieni traccia degli aggiornamenti e delle patch di sicurezza per i cluster creati con Amazon EKS Distro. Per ulteriori informazioni, consulta Presentazione di Amazon EKS Distro - una distribuzione Kubernetes open source utilizzata da Amazon EKS .
Amazon Elastic Container Service (Amazon ECS) - Consente di eseguire, arrestare e gestire i contenitori in un cluster.	Ricevi notifiche quando è disponibile un nuovo AMI ottimizzato per Amazon ECS. Per ulteriori informazioni, consulta la sezione relativa alla sottoscrizione alle notifiche per l'aggiornamento di AMI Linux ottimizzate per Amazon ECS nella Guida per gli sviluppatori di Amazon Elastic Container Service.

Servizi di coinvolgimento dei clienti

La tabella seguente descrive come Amazon SNS migliora i servizi di coinvolgimento dei clienti grazie all'integrazione con Amazon Connect e Amazon Simple Email Service (SES) AWS End User Messaging SMS, consentendoti di ricevere avvisi e convalide, configurare messaggi SMS bidirezionali 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	Vantaggio dell'utilizzo con Amazon SNS
Amazon Connect - Consente di configurare un contact center cloud omnicanale per interagire con i clienti.	Ricevi avvisi e convalide. Per ulteriori informazioni, consulta La potenza di AWS Amazon Connect nella Amazon Connect Administrator Guide .
AWS End User Messaging SMS — Ti aiuta a coinvolgere i tuoi clienti inviando loro e-mail, SMS, messaggi vocali e notifiche push.	Configura SMS bidirezionali, consentendo così di ricevere messaggi dai clienti. Per ulteriori informazioni, consulta Messaggi SMS bidirezionali .

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
	nali nella Guida per l'AWS End User Messaging SMS utente.
Amazon Simple Email Service (Amazon SES) - Offre un metodo conveniente per inviare e ricevere e-mail usando domini e indirizzi e-mail personali.	Ricevi notifiche di mancati recapiti (bounce), reclami e consegne. Per maggiori informazioni, consulta Configurare le notifiche Amazon SNS per Amazon SES nella Guida per sviluppatori di Amazon Simple EmailService.

Servizi di database

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS database come AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon, Amazon ElastiCache Neptune, Amazon Redshift e Amazon Relational Database Service (RDS) 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	Vantaggio dell'utilizzo con Amazon SNS
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 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 - Combina prestazioni elevate e prevedibili con una scalabilità ottimale nel servizio database NoSQL interamente gestito.	Ricevi notifiche quando si verificano eventi di manutenzione. Per ulteriori informazioni, consulta Personalizzazione delle impostazioni del cluster DAX nella Guida per gli sviluppatori di Amazon DynamoDB.

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
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.
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 degli eventi RDS di Amazon. Per ulteriori informazioni, consulta la sezione Utilizzo della notifica eventi Amazon RDS nella Guida per l'utente di Amazon RDS.

Servizi e strumenti per sviluppatori

La tabella seguente descrive come Amazon SNS si integra con i servizi di strumenti per AWS sviluppatori, come AWS CodeDeploy Amazon AWS CodeBuild AWS CodeCommit CodeGuru, e AWS CodePipeline 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	Vantaggio dell'utilizzo con Amazon SNS
AWS CodeBuild - Compila il codice sorgente, esegue unit test e prepara artefatti pronti per essere distribuiti.	Ricevi notifiche quando le build hanno esito positivo, non riescono o si spostano da una fase di compilazione all'altra. Per ulteriori informazioni, consulta Build notification example for 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 argomento Amazon SNS nella Guida per l'AWS CodeCommit utente.
AWS CodeDeploy — Automatizza la distribuzione dell'applicazione a istanze Amazon EC2, istanze locali, funzioni Lambda serverless o servizi Amazon ECS.	Ricevi notifiche per CodeDeploy distribuzioni o eventi di istanza. 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.

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-mail, SMS, messaggi vocali e notifiche push, inclusa la possibilità di configurare SMS bidirezionali per la ricezione 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	Vantaggio dell'utilizzo con Amazon SNS
AWS End User Messaging SMS — Ti aiuta a coinvolgere i tuoi clienti inviando loro e-mail, SMS, messaggi vocali e notifiche push.	Configura SMS bidirezionali, consentendo così di ricevere messaggi dai clienti. Per ulteriori informazioni, consulta Messaggi SMS bidirezionali 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 Servers 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	Vantaggio dell'utilizzo con Amazon SNS
Amazon GameLift Servers : fornisce soluzioni per l'hosting di server di gioco multiplayer basati su sessioni nel cloud, incluso un servizio completamente gestito per l'implementazione, 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 Servers FlexMatch Developer Guide. • Per le notifiche sulla coda, consulta Configurare la notifica degli eventi per il posizionamento delle sessioni di gioco nella Amazon GameLift Servers Developer Guide.

Servizi Internet of Things

La tabella seguente descrive come Amazon SNS si integra AWS IoT con servizi AWS IoT Core come,, AWS IoT Greengrass e AWS IoT Device Defender AWS IoT Events, 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	Vantaggio 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 Creazione di una regola Amazon SNS nella AWS IoT Guida per gli sviluppatori.
AWS IoT Device Defender – Esegue l'auditing della configurazione dei dispositivi, monitorar e i dispositivi connessi per rilevare eventuali comportamenti anomali e mitigare i rischi di sicurezza.	Ricevi allarmi quando un dispositivo viola un comportamento. Per ulteriori 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 funzionamento 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 la sezione relativa al connettore SNS nella Guida per lo sviluppatore di AWS IoT Greengrass Version 1 .

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

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

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
Amazon CodeGuru : raccoglie i dati sulle prestazioni di runtime dalle tue applicazioni live e fornisce consigli che possono aiutarti a ottimizzare le prestazioni delle tue applicazioni.	Ricevi notifiche quando si verificano anomalie. Per ulteriori informazioni, consulta Lavorare con i report sulle anomalie e sui consigli nella Amazon CodeGuru User Guide.
Amazon DevOps Guru : genera informazioni operative utilizzando l'apprendimento automatico per aiutarti a migliorare le prestazioni delle tue applicazioni operative.	Proseguire approfondimenti e conferme. Per ulteriori informazioni, consulta Fornisci informazioni operative basate 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 rapidamente.	Ricevi notifiche di anomalie. Per ulteriori informazioni, consulta Utilizzo di Amazon SNS con Lookout for Metrics nella Guida per gli sviluppatori di Amazon Lookout for Metrics.
Amazon Rekognition - Consente di aggiungere 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 Rekognition.
Amazon SageMaker AI : consente ai data scientist e agli sviluppatori di creare e addestrare modelli di machine learning e quindi di distribuirli direttamente in un ambiente ospitato pronto per la produzione.	Ricevi notifiche quando un oggetto dati viene etichettato. Per ulteriori informazioni, consulta Creazione di un processo di etichettatura in streaming nella Amazon SageMaker AI Developer Guide.

Servizi di governance e gestione

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

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	Vantaggio dell'utilizzo con Amazon SNS
Amazon Q Developer nelle applicazioni di chat : 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	Distribuisce notifiche alle chat room. Per ulteriori informazioni, consulta Configurazione di Amazon Q Developer nelle applicazioni di chat nella Guida per l'amministratore delle applicazioni di chat di Amazon Q Developer.
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 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 CloudTrail I pubblicati nuovi file di log nel tuo bucket Amazon S3. Per ulteriori informazioni, consulta Configurazione delle notifiche di Amazon SNS nella Guida CloudTrail per AWS CloudTrail l'utente.
Amazon CloudWatch : monitora AWS le tue risorse e le applicazioni su cui esegui AWS in tempo reale.	Ricevi notifiche quando gli allarmi cambiano stato. Per ulteriori informazioni, consulta Using Amazon CloudWatch alarms nella Amazon CloudWatch User Guide.

Servizio AWS	Vantaggio 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 argomento SNS 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 Monitoraggio delle modifiche di stato di Systems Manager utilizzando le notifiche Amazon SNS nella AWS Systems Manager Guida per l'utente di.

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	Vantaggio 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) AWS Application Discovery Service, e per fornire notifiche per eventi come la raccolta di dati sul server AWS Snowball Edge, 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	Vantaggio 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, consulta Registrazione delle chiamate API dell'Application Discovery Service con AWS CloudTrail nella Guida per l'utente dell'Application Discovery Service.

Servizio AWS	Vantaggio 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.
AWS Snowball Edge — 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 di Snowball Edge. Per ulteriori informazioni, consulta Notifiche per i dispositivi Snow Family nella Guida per l'AWS Snowball Edge utente.

Reti e servizi di distribuzione di contenuti

La tabella seguente descrive come Amazon SNS si integra con i servizi di AWS rete e distribuzione di contenuti, come Amazon API Gateway, Amazon, Elastic Load Balancing CloudFront Direct Connect, Amazon Route 53 e Amazon VPC, per inviare notifiche per eventi come messaggi API, allarmi metrici, modifiche dello stato della connessione CloudFront, eventi di bilanciamento del carico, stati dei controlli di integrità 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	Vantaggio dell'utilizzo con Amazon SNS
Amazon API Gateway : consente di creare e distribuire il proprio REST WebSocket APIs su qualsiasi scala.	Ricevere messaggi inviati a un endpoint API Gateway. Per ulteriori informazioni, consulta Tutorial: Build an API Gateway REST API Gateway con AWS integrazione nella API Gateway Developer Guide.
Amazon CloudFront : velocizza la distribuzione di contenuti Web statici e dinamici,	Ricevi notifiche quando si verificano allarmi basati su metriche specificate. CloudFront

Servizio AWS	Vantaggio dell'utilizzo con Amazon SNS
come .html, .css, .php, immagini e file multimediali.	Per ulteriori informazioni, consulta la sezione Impostazione degli allarmi per ricevere notifiche nell'Amazon CloudFront Developer Guide.
Direct Connect — Collega la rete interna a una Direct Connect posizione tramite un cavo Ethernet standard in fibra ottica.	Ricevi notifiche quando allarmi che monitorano lo stato di un stato Direct Connect di modifica della connessione. Per ulteriori informazioni, vedere Creazione di CloudWatch allarmi per monitorare le Direct Connect connessioni nella Guida per l'utente Direct Connect.
Elastic Load Balancing — Distribuisce automaticamente il traffico in ingresso su più destinazioni, ad esempio su istanze Amazon EC2, contenitori e indirizzi IP, in più o più zone di disponibilità.	Ricevi notifiche degli allarmi creati per gli eventi di bilanciamento del carico. Per ulteriori informazioni, consulta Creare CloudWatch allarmi per il sistema di bilanciamento del carico nella Guida utente di Classic Load Balancers .
Amazon Route 53 : fornisce la registrazione del dominio, il routing DNS e il controllo dell'integrità.	Ricevere una notifica quando un controllo dello stato non è integro. Per ulteriori informazioni, consulta Come ricevere una notifica Amazon SNS quando un controllo dello stato non è integro (console) nella Guida per sviluppatori di Amazon Route 53.
Amazon Virtual Private Cloud (Amazon VPC) : consente di avviare AWS risorse in una rete virtuale definita dall'utente.	Ricevere notifiche per eventi specifici che si verificano nell'endpoint di interfaccia. Per ulteriori informazioni, consulta Come creare e gestire una notifica per un servizio endpoint nella Amazon VPC User Guide.

Servizi per sicurezza, identità e conformità

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

per le modifiche allo stato delle directory AWS Security Hub CSPM, i risultati di sicurezza, gli eventi di Inspector 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	Vantaggio dell'utilizzo con Amazon SNS
AWS Directory Service — Fornisce diversi modi per utilizzare Microsoft Active Directory (AD) con altri Servizi AWS.	Ricevere e-mail o messaggi di testo (SMS) quando lo stato della directory cambia. Per ulteriori informazioni, consulta Configurare le notifiche sullo stato della directory nella AWS Directory Service Guida di amministrazione.
Amazon GuardDuty : fornisce un monitoraggio continuo della sicurezza per aiutare a identificare 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'argomento Abbonamento agli GuardDuty annunci SNS nella Amazon User Guide . GuardDuty
Amazon Inspector - Esegue il test dell'accessibilità di rete delle istanze Amazon EC2 e dello stato di sicurezza delle applicazioni che vengono eseguite sulle istanze.	Ricevi notifiche per gli eventi Amazon Inspector . Per ulteriori informazioni, consulta Impostazioni di un argomento SNS per le notifiche di Amazon Inspector nella Guida dell'utente di Amazon Inspector.
AWS Security Hub CSPM : automatizza i controlli di sicurezza di AWS I e centralizza gli avvisi di sicurezza.	Ricevi notifiche sugli AWS Security Hub CSPM annunci, incluse notifiche su AWS Security Hub CSPM controlli o standard che sono stati aggiunti, modificati o ritirati. Per ulteriori informazioni, consulta Sottoscrizione agli AWS Security Hub CSPM annunci con Amazon SNS .

Servizi serverless

La tabella seguente descrive come Amazon SNS si integra con servizi come Amazon DynamoDB, Amazon EventBridge 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	Vantaggio dell'utilizzo con Amazon SNS
Amazon DynamoDB – Combina prestazioni elevate e prevedibili con una scalabilità ottimale nel servizio database NoSQL interamente gestito.	Ricevi notifiche quando si verificano eventi di manutenzione. Per ulteriori informazioni, consulta Personalizzazione delle impostazioni del cluster DAX nella Guida per gli sviluppatori di Amazon DynamoDB.
Amazon EventBridge : fornisce un flusso di dati in tempo reale dalle tue applicazioni 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 EventBridge gli obiettivi di Amazon nella Amazon EventBridge User Guide.
AWS Lambda - Consente di eseguire il codice senza effettuare il provisioning dei server o senza gestirli.	Ricevere i dati di output della funzione impostando un argomento SNS come una coda di lettere non recapitate Lambda o una destinazione Lambda. Per ulteriori informazioni, consulta Chiamata asincrona nella AWS Lambda Guida per gli sviluppatori.

Servizi di storage

La tabella seguente descrive come Amazon SNS si integra con servizi di AWS storage come Amazon Elastic File System (EFS) AWS Backup, Amazon Glacier, Amazon S3 AWS Snowball Edge e fornisce notifiche per vari eventi come attività di backup, allarmi del file system, processi di recupero dati, modifiche ai bucket e operazioni di trasferimento 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	Vantaggio dell'utilizzo con Amazon SNS
AWS Backup — Consente di centralizzare e automatizzare il backup dei dati nel cloud e Servizi AWS in locale	Ricevi notifiche di eventi. AWS Backup Per ulteriori informazioni, consulta Using Amazon SNS per tenere traccia AWS Backup degli eventi nella AWS Backup Developer Guide.
Amazon Elastic File System - Consente di archiviare i file per le istanze Amazon EC2.	Ricevi notifiche degli avvisi che hai creato per gli eventi EFS Amazon. Per ulteriori informazioni, consulta Strumenti di monitoraggio automatizzato nella Guida per l'utente di Amazon Elastic File System.
Amazon Glacier : fornisce lo storage per i dati utilizzati di rado.	Impostare la configurazione delle notifiche per un vault di modo che un messaggio venga inviato a un argomento di SNS al completamento del processo. Per ulteriori informazioni, consulta Configurazione delle notifiche di vault in Amazon Glacier nella Amazon Glacier Developer Guide .
Amazon Simple Storage Service (Amazon S3) — Fornisce l'archiviazione degli oggetti.	Ricevi notifiche quando si verificano modifiche a un bucket Amazon S3 o nel raro caso in cui gli oggetti non vengono replicati nella regione di destinazione. Per ulteriori informazioni, consulta Spiegazione passo per passo: come configurare un bucket per le notifiche (argomento SNS o coda SQS) e Monitoraggio dell'avanzamento con i parametri di replica e le notifiche di eventi Amazon S3 nella Guida per l'utente di Amazon Simple Storage Service.
AWS Snowball Edge — Utilizza dispositivi di archiviazione fisici per trasferire grandi quantità di dati tra Amazon S3 e la posizione di	Ricevi notifiche per i lavori di Snowball Edge. Per ulteriori informazioni, consulta Notifiche per

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

Origini eventi aggiuntivi

La tabella seguente descrive come utilizzare Amazon SNS 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 release di AWS servizio, sui tipi di istanze, sugli endpoint VPC 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 risorse in modo efficace.

Origine	Vantaggio dell'utilizzo con Amazon SNS
AWS Aggiornamenti giornalieri delle funzionalità	Ricevi tempestivamente informazioni dettagliate su release e aggiornamenti AWS tramite un argomento Amazon SNS. Queste versioni includono endpoint Amazon VPC Regioni AWS Servizi AWS, Servizi AWS integrati con AWS Service Quotas, tipi di istanze Amazon EC2, tipi di istanze Amazon SageMaker AI, tipi di istanze Amazon Nimble Studio, versioni del motore di database Amazon RDS e versioni di Amazon MSK Apache 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 argomento di Amazon SNS. 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	Vantaggio 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 AWS e servizi di elaborazione, storage, database e rete](#) sul blog di elaborazione AWS
- [Arricchimento delle architetture basate sugli eventi con le pipeline Event Fork sul blog](#) di Compute AWS AWS

Destinazioni eventi di Amazon SNS

Questo argomento elenca tutte le destinazioni degli eventi, organizzate in base alla [messaggistica application-to-application \(A2A\)](#) e alle notifiche [application-to-person \(A2P\)](#).

Note

Amazon SNS ha introdotto [Argomenti FIFO](#) nel mese di ottobre 2020. Attualmente, la maggior parte dei AWS servizi supporta la ricezione di eventi solo da argomenti standard di SNS. Amazon SQS supporta la ricezione di eventi sia dagli argomenti standard SNS che FIFO.

A2A destinazioni

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

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 di Amazon SNS
https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html	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 delle notifiche Amazon SNS alle funzioni Lambda per l'elaborazione automatizzata .
Amazon SQS	Distribuire gli eventi alle code per scopi di integrazione delle applicazioni. Per ulteriori informazioni, consulta Esegui l'eliminazione delle notifiche Amazon SNS alle code di Amazon SQS per l'elaborazione asincrona .
AWS Event Fork Pipelines	Distribuisce 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 degli eventi Amazon SNS su Event Fork AWS Pipelines .
HTTP/S	Distribuire eventi a webhook esterni. Per ulteriori informazioni, consulta Fanout delle notifiche Amazon SNS 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 e notifiche push native, caselle di posta elettronica, chat room di Amazon Chime, canali Slack e informazioni operative per i team in chiamata 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 di 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 e-mail Amazon SNS .
Endpoint platform	Distribuisce eventi ai telefoni cellulari come notifiche push native. Per ulteriori informazioni, consulta Invio di notifiche push per dispositivi mobili con Amazon SNS .
Amazon Q Developer nelle applicazioni di chat	Consegna eventi alle chat room Amazon Chime o ai canali Slack. Per ulteriori informazioni, consulta le seguenti pagine della Guida per gli amministratori delle applicazioni Amazon Q Developer in chat: • Configurazione di Amazon Q Developer nelle applicazioni di chat con Amazon Chime •

Destinazione di evento	Vantaggi dell'utilizzo di Amazon SNS
	Configurazione di Amazon Q Developer nelle applicazioni di chat con Slack Utilizzo di Amazon Q Developer nelle applicazioni di chat con altri AWS servizi
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.

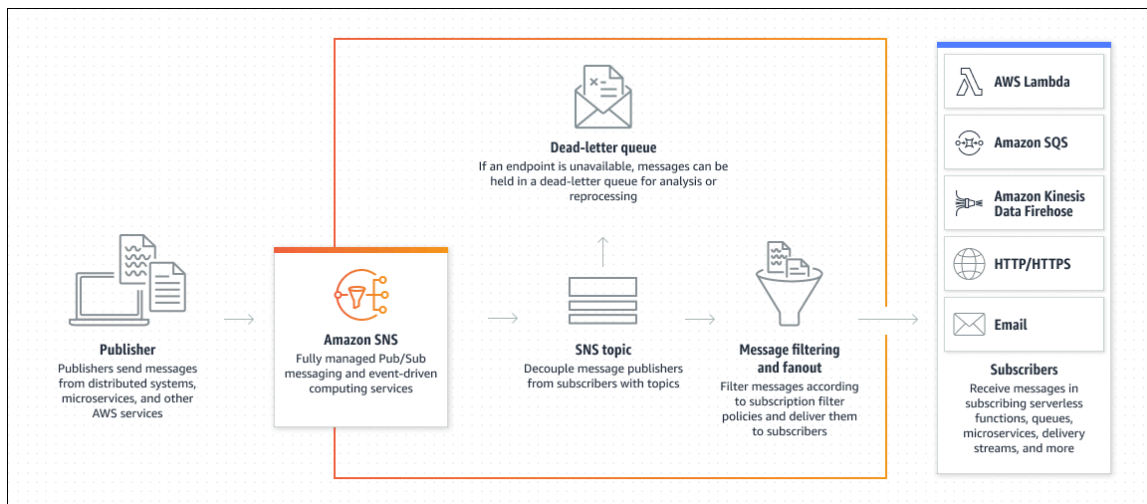
Note

Puoi offrire sia AWS eventi nativi che eventi personalizzati alle app di chat:

- AWS Eventi nativi:** puoi utilizzare Amazon Q Developer nelle applicazioni di chat per inviare AWS eventi nativi, tramite argomenti di Amazon SNS, ad Amazon Chime e Slack. Il set di AWS eventi nativi supportato include eventi di Gestione dei costi e fatturazione AWS, AWS Health, AWS CloudFormation CloudWatch, Amazon e altro ancora. Per ulteriori informazioni, consulta la sezione [Uso di Amazon Q Developer nelle applicazioni di chat con altri servizi](#) nella Guida per l'amministratore delle applicazioni di chat di Amazon Q Developer.
- Eventi personalizzati** – Puoi anche inviare i tuoi eventi personalizzati, tramite gli argomenti di Amazon SNS, ad Amazon Chime, Slack e Microsoft Teams. A tale scopo, è possibile pubblicare eventi personalizzati in un argomento SNS, che fornisce gli eventi a una funzione Lambda sottoscritta. La funzione Lambda utilizza quindi il webhook dell'app di chat per consegnare gli eventi ai destinatari. Per ulteriori informazioni, consulta [Come faccio a utilizzare i webhook per pubblicare messaggi Amazon SNS su Amazon Chime, Slack o Microsoft Teams?](#)

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

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 argomenti di Amazon SNS, dove possono essere filtrati e recapitati a sottoscrittori come Lambda, Amazon SQS o endpoint HTTP. Se la consegna non riesce, i messaggi vengono archiviati in una coda di lettere non scritte per ulteriori analisi o rielaborazioni.



Flussi di distribuzione da Fanout a Firehose

Puoi abbonare [i flussi di distribuzione](#) agli argomenti di Amazon SNS, in modo da inviare notifiche a endpoint di storage e analisi aggiuntivi. I messaggi pubblicati su un argomento di Amazon SNS vengono inviati al flusso di distribuzione Firehose sottoscritto e consegnati alla destinazione come configurato in Firehose. Il titolare di un abbonamento può abbonare fino a cinque stream di distribuzione Firehose a un argomento Amazon SNS. Ogni flusso di distribuzione Firehose ha una [quota predefinita](#) per le richieste e la velocità effettiva al secondo. Questa limitazione potrebbe comportare un numero maggiore di messaggi pubblicati (traffico in entrata) rispetto a quelli consegnati (traffico in uscita). Quando c'è più traffico in ingresso rispetto a quello in uscita, la sottoscrizione può accumulare un backlog di messaggi di grandi dimensioni, che potrebbe causare un'elevata latenza di recapito dei messaggi. Puoi richiedere un [aumento della quota](#) in base alla percentuale di pubblicazione per evitare impatti negativi sul carico di lavoro.

Tramite i flussi di distribuzione Firehose, puoi inviare le notifiche di Amazon SNS ad Amazon Simple Storage Service (Amazon S3), Amazon Redshift, OpenSearch Amazon Service (Service) e a fornitori di servizi di terze parti come Datadog, New Relic, MongoDB e Splunk. OpenSearch

Ad esempio, puoi utilizzare questa funzionalità per archiviare in modo permanente i messaggi inviati a un argomento in un bucket Amazon S3 a fini di conformità, archiviazione o altri scopi. A tale scopo, crea un flusso di distribuzione Firehose con una destinazione bucket Amazon S3 e sottoscrivi tale flusso di consegna all'argomento Amazon SNS. Come altro esempio, per eseguire analisi sui messaggi inviati a un argomento Amazon SNS, crea un flusso di distribuzione con una destinazione dell'indice di OpenSearch servizio. Puoi quindi sottoscrivere lo stream di distribuzione di Firehose all'argomento Amazon SNS.

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

Prerequisiti per la sottoscrizione dei flussi di distribuzione di Firehose agli argomenti di Amazon SNS

Per sottoscrivere un flusso di distribuzione a un argomento SNS, è Account AWS necessario disporre di:

- Un argomento SNS standard. Per ulteriori informazioni, consulta [Creare un argomento Amazon SNS](#).
- Un flusso di distribuzione Firehose. Per ulteriori informazioni, consulta [Creare un flusso di distribuzione](#) e [concedere 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 principale del servizio Amazon SNS e dispone dell'autorizzazione per scrivere nel flusso di distribuzione. Inserisci l'Amazon Resource Name (ARN) di questo ruolo come `SubscriptionRoleARN` quando crei la sottoscrizione. Amazon SNS assume questo ruolo, che consente ad Amazon SNS di inserire record nel flusso di distribuzione Firehose.

Di seguito viene illustrato un esempio di policy con autorizzazioni suggerite:

JSON

```
{  
  "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Action": [  
      "firehose:DescribeDeliveryStream",  
      "firehose:ListDeliveryStreams",  
      "firehose:ListTagsForDeliveryStream",  
      "firehose:PutRecord",  
      "firehose:PutRecordBatch"  
    ],  
    "Resource": [  
      "arn:aws:firehose:us-east-1:111111111111:deliverystream/firehose-sns-delivery-stream"  
    ],  
    "Effect": "Allow"  
  }  
]
```

Per fornire l'autorizzazione completa all'utilizzo di Firehose, puoi anche utilizzare la policy AWS gestita. `AmazonKinesisFirehoseFullAccess` In alternativa, per fornire autorizzazioni più rigorose per l'utilizzo di Firehose, è possibile creare una politica personalizzata. Come minimo, la policy deve dare l'autorizzazione per eseguire l'operazione `PutRecord` su un flusso di consegna specifico.

In tutti i casi, bisogna anche modificare la relazione di fiducia per includere il servizio principale Amazon SNS. Esempio:

JSON

```
{  
  "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'utente IAM](#).

Dopo aver completato questi requisiti, puoi [sottoscrivere il flusso di consegna all'argomento SNS](#).

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

[Per inviare le notifiche di Amazon SNS ai flussi di distribuzione, assicurati innanzitutto di aver soddisfatto tutti i prerequisiti](#). Per un elenco degli endpoint supportati, consulta [endpoint](#) e quote nel. Riferimenti generali di Amazon Web Services

Per abbonare uno stream di distribuzione di Firehose a un argomento

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione scegli Subscriptions (Sottoscrizioni).
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
 - a. Per Topic ARN (ARN argomento) scegli l'Amazon Resource Name (ARN) di un argomento standard.
 - b. Per Protocollo, scegliete Firehose.
 - c. Per Endpoint, scegli l'ARN di un flusso di distribuzione Firehose in grado di ricevere notifiche da Amazon SNS.
 - d. Per il ruolo di sottoscrizione ARN, specificate l'ARN del ruolo AWS Identity and Access Management (IAM) creato per la scrittura nei flussi di distribuzione Firehose. Per ulteriori informazioni, consulta [Prerequisiti per la sottoscrizione dei flussi di distribuzione di Firehose agli argomenti di Amazon SNS](#).
 - e. (Facoltativo) Per rimuovere i metadati di Amazon SNS dai messaggi pubblicati, scegli Enable raw message delivery (Abilita il recapito di messaggi). Per ulteriori informazioni, consulta [Consegna di messaggi non elaborati Amazon SNS](#).
5. (Facoltativo) Per configurare un criterio di filtro, espandere la sezione policy di filtro della sottoscrizione. Per ulteriori informazioni, consulta [Policy di filtro degli abbonamenti Amazon SNS](#).

6. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta [Code di lettere non ricevute di Amazon SNS](#).
7. Scegli Create Subscription (Crea sottoscrizione).

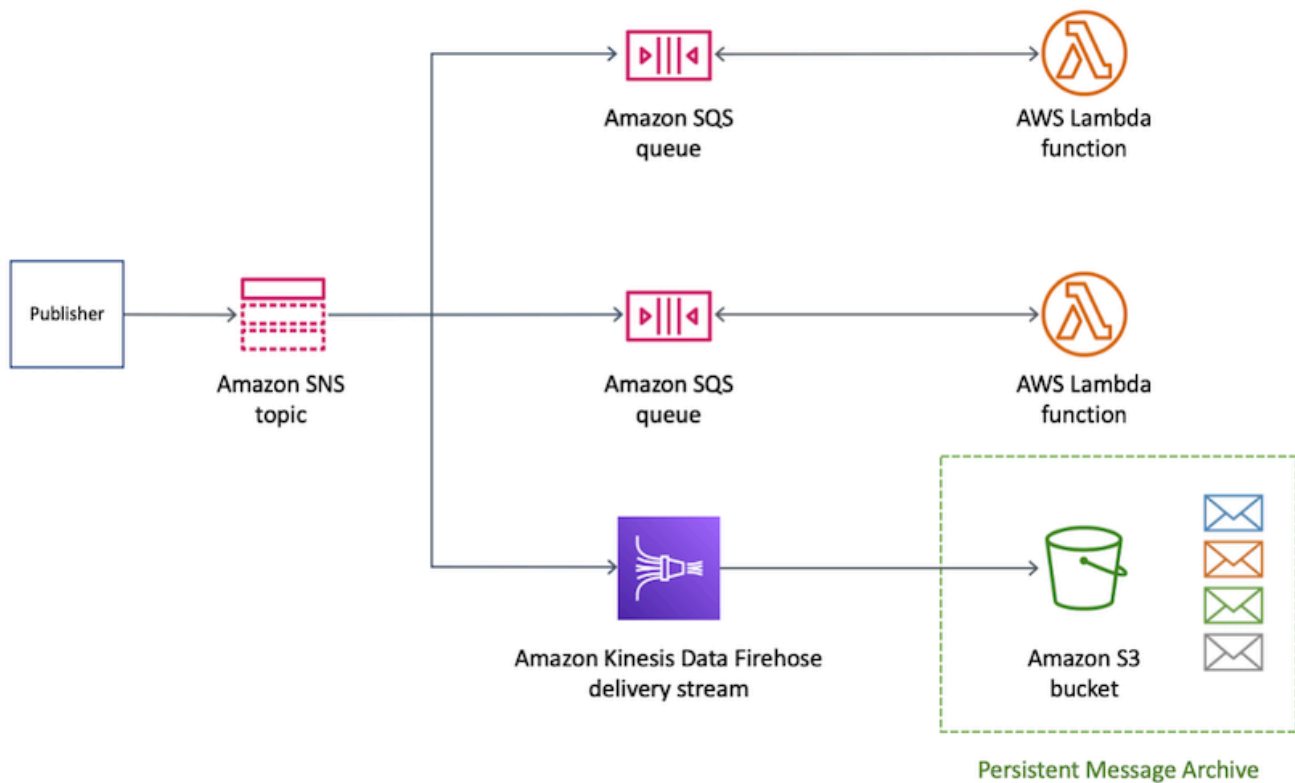
La console crea la sottoscrizione e apre la pagina dettagli della sottoscrizione.

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

[i flussi di consegna](#) consentono di gestire i messaggi Amazon SNS su più destinazioni, abilitando l'integrazione con Amazon S3, Amazon Service OpenSearch , Amazon Redshift e gli endpoint HTTP per lo storage, l'indicizzazione e l'analisi. Configurando correttamente la formattazione e la consegna dei messaggi, puoi archiviare le notifiche di Amazon SNS in Amazon S3 per elaborarle successivamente, analizzare i dati strutturati dei messaggi utilizzando Amazon Athena, indicizzare i messaggi per la ricerca e la visualizzazione in tempo reale e strutturare gli archivi OpenSearch in Amazon Redshift per l'interrogazione avanzata.

Archiviazione e analisi dei messaggi Amazon SNS nelle destinazioni Amazon S3

Questo argomento spiega come i flussi di distribuzione pubblicano i dati su Amazon Simple Storage Service (Amazon S3).



Argomenti

- [Formattazione delle notifiche Amazon SNS per lo storage nelle destinazioni Amazon S3](#)
- [Analisi dei messaggi Amazon SNS archiviati in Amazon S3 utilizzando Athena](#)

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

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

Note

In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito dei messaggi non elaborati è disabilitato, Amazon SNS aggiunge i metadati JSON al messaggio, incluse le seguenti proprietà:

- Type
- MessageId
- TopicArn

- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

Per ulteriori informazioni sulla distribuzione non elaborata, consulta [Consegna di messaggi non elaborati Amazon SNS](#).

```
{
  "Type": "Notification",
  "MessageId": "719a6bbf-f51b-5320-920f-3385b5e9aa56",
  "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-kinesis-test-topic",
  "Subject": "My 1st subject",
  "Message": "My 1st body",
  "Timestamp": "2020-11-26T23:48:02.032Z",
  "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:333333333333:my-kinesis-test-
topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5",
  "MessageAttributes": {
    "myKey1": {
      "Type": "String",
      "Value": "myValue1"
    },
    "myKey2": {
      "Type": "String",
      "Value": "myValue2"
    }
  }
}
```

L'esempio seguente mostra tre messaggi SNS inviati tramite un flusso di distribuzione allo stesso bucket Amazon S3. Viene applicato il buffering e le interruzioni di riga separano ogni messaggio.

```
{"Type":"Notification","MessageId":"d7d2513e-6126-5d77-
bbe2-09042bd0a03a","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 1st subject","Message":"My 1st
body","Timestamp":"2020-11-27T00:30:46.100Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
```

```
b59b-3b4aa6d8fcf5", "MessageAttributes": {"myKey1":
{"Type": "String", "Value": "myValue1"}, "myKey2": {"Type": "String", "Value": "myValue2"}}}
{"Type": "Notification", "MessageId": "0c0696ab-7733-5bfb-b6db-
ce913c294d56", "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic", "Subject": "My 2nd subject", "Message": "My 2nd
body", "Timestamp": "2020-11-27T00:31:22.151Z", "UnsubscribeURL": "https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5", "MessageAttributes": {"myKey1": {"Type": "String", "Value": "myValue1"}}}
{"Type": "Notification", "MessageId": "816cd54d-8cfa-58ad-91c9-8d77c7d173aa", "TopicArn": "arn:aws:s
east-1:333333333333:my-kinesis-test-topic", "Subject": "My 3rd subject", "Message": "My
3rd body", "Timestamp": "2020-11-27T00:31:39.755Z", "UnsubscribeURL": "https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5"}
}
```

Analisi dei messaggi Amazon SNS archiviati in Amazon S3 utilizzando Athena

Questa pagina spiega come analizzare i messaggi Amazon SNS inviati tramite flussi di consegna a destinazioni Amazon Simple Storage Service (Amazon S3).

Per analizzare i messaggi SNS inviati tramite i flussi di distribuzione Firehose verso destinazioni Amazon S3

1. Configura le risorse Amazon S3. Per istruzioni, consulta [Creazione di un bucket](#) nella Guida per l'utente di Amazon Simple Storage Service e [Utilizzo dei bucket Amazon S3](#) nella Guida per l'utente di Amazon Simple Storage Service.
2. Configura il flusso di consegna. Per istruzioni, consulta [Scegli Amazon S3 per la tua destinazione](#) nella Amazon Data Firehose Developer Guide.
3. Utilizza [Amazon Athena](#) per eseguire query sugli oggetti Amazon S3 utilizzando SQL standard. Per ulteriori informazioni, consulta l'argomento relativo alle [nozioni di base](#) nella Guida per l'utente di Amazon Athena.

Query di esempio

Per questa query di esempio, supponiamo quanto segue:

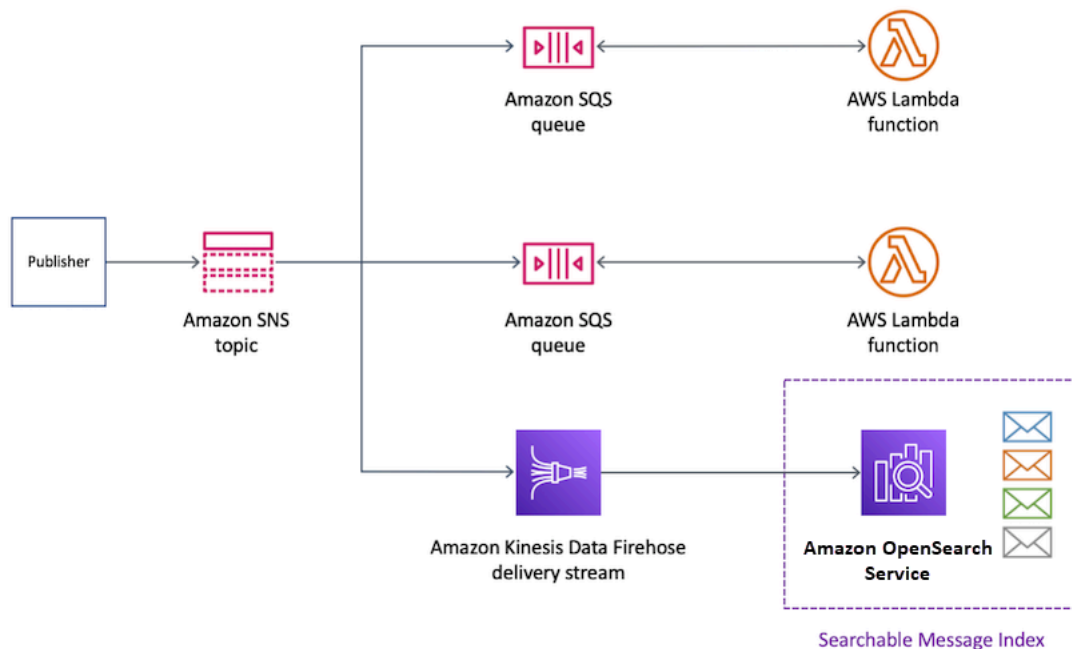
- I messaggi vengono archiviati nella tabella `notifications` nello schema `default`.
- La tabella `notifications` include una colonna `timestamp` con un tipo di `string`.

La query seguente restituisce tutti i messaggi SNS ricevuti nell'intervallo di date specificato:

```
SELECT *
FROM default.notifications
WHERE from_iso8601_timestamp(timestamp) BETWEEN TIMESTAMP '2020-12-01 00:00:00' AND
TIMESTAMP '2020-12-02 00:00:00';
```

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

Questa sezione spiega come i flussi di distribuzione pubblicano i dati su Amazon OpenSearch Service (OpenSearch Service).



Argomenti

- [Archiviazione e formattazione delle notifiche OpenSearch Amazon SNS negli indici di servizio](#)
- [Analisi dei messaggi OpenSearch Amazon SNS per le destinazioni dei servizi](#)

Archiviazione e formattazione delle notifiche OpenSearch Amazon SNS negli indici di servizio

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

 Note

In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito dei messaggi non elaborati è disabilitato, Amazon SNS aggiunge i metadati JSON al messaggio, incluse le seguenti proprietà:

- Type
- MessageId
- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

Per ulteriori informazioni sulla distribuzione non elaborata, consulta [Consegna di messaggi non elaborati Amazon SNS](#).

```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "49613100963111323203250405402193283794773886550985932802.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  }
}
```

```
    }  
  }  
},  
"fields": {  
  "Timestamp": [  
    "2020-12-02T22:29:21.189Z"  
  ]  
},  
"sort": [  
  1606948161189  
]  
}
```

Analisi dei messaggi OpenSearch Amazon SNS per le destinazioni dei servizi

Questo argomento spiega come analizzare i messaggi Amazon SNS inviati tramite flussi di consegna verso destinazioni Amazon OpenSearch Service (OpenSearch Service).

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

1. Configura le risorse del tuo servizio. OpenSearch Per istruzioni, consulta la sezione Guida [introduttiva ad Amazon OpenSearch Service](#) nella Amazon OpenSearch Service Developer Guide.
2. Configura il flusso di consegna. Per istruzioni, consulta [Choose OpenSearch Service for Your Destination](#) nella Amazon Data Firehose Developer Guide.
3. Esegui una query utilizzando OpenSearch Service queries e Kibana. Per ulteriori informazioni, consulta [Step 3: Search Documents in an OpenSearch Service Domain](#) e [Kibana](#) nella Amazon OpenSearch Service Developer Guide.

Query di esempio

Nell'esempio seguente viene eseguita una query sull'indice `my-index` per tutti i messaggi SNS ricevuti nell'intervallo di date specificato:

```
POST https://search-my-domain.us-east-1.es.amazonaws.com/my-index/_search  
{  
  "query": {  
    "bool": {  
      "filter": [  

```

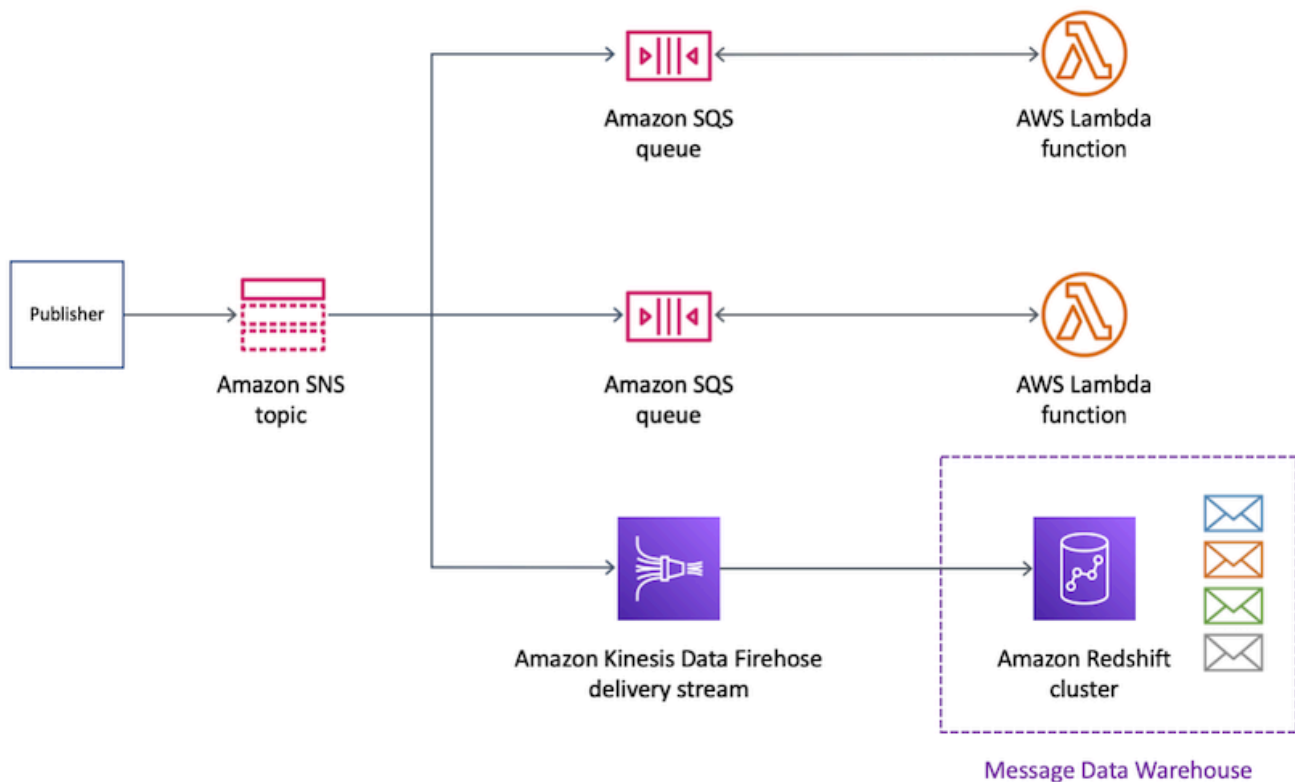
```

{
  "range": {
    "Timestamp": {
      "gte": "2020-12-08T00:00:00.000Z",
      "lte": "2020-12-09T00:00:00.000Z",
      "format": "strict_date_optional_time"
    }
  }
}
]
}
}
}

```

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

Questo argomento spiega come suddividere le notifiche di Amazon SNS in un flusso di distribuzione, che quindi pubblica i dati su Amazon Redshift. Con questa configurazione, puoi connetterti al database Amazon Redshift e utilizzare uno strumento di query SQL per recuperare messaggi Amazon SNS che soddisfano criteri specifici.



Argomenti

- [Strutturazione degli archivi di messaggi Amazon SNS nelle tabelle di Amazon Redshift](#)
- [Analisi dei messaggi Amazon SNS archiviati nelle destinazioni Amazon Redshift](#)

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

Per gli endpoint Amazon Redshift, i messaggi Amazon SNS vengono archiviati come righe in una tabella. Ecco un esempio di come vengono archiviati i dati:

Note

In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Quando il recapito dei messaggi non elaborati è disabilitato, Amazon SNS aggiunge i metadati JSON al messaggio, incluse le seguenti proprietà:

- Type
- MessageId
- TopicArn
- Subject
- Message
- Timestamp
- UnsubscribeURL
- MessageAttributes

Per ulteriori informazioni sulla distribuzione non elaborata, consulta [Consegna di messaggi non elaborati Amazon SNS](#).

Sebbene Amazon SNS aggiunga proprietà al messaggio utilizzando le maiuscole mostrate in questo elenco, i nomi delle colonne nelle tabelle Amazon Redshift vengono visualizzati con caratteri minuscoli. Per trasformare i metadati JSON per l'endpoint Amazon Redshift, puoi utilizzare il comando SQL COPY. Per ulteriori informazioni, consulta [Esempi di copia da JSON](#) e [Caricamento da dati JSON utilizzando l'opzione "auto ignorecase"](#) nella Guida per gli sviluppatori di database di Amazon Redshift.

tipo	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageAttributes
Notificazione	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&SubscriptionArn=arn:aws:sns:us-east-1:111111:my-topic:326deebcbf4-45dab92b-ca77a247813b	{\"my_attribute\": {\"Type\": \"String\", \"Value\": \"my_value\"}}
Notificazione	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111	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-1:111111:my-topic:326deebcbf4-45dab92b-ca77a247813b	{\"my_attribute2\": {\"Type\": \"String\", \"Value\": \"my_value2\"}}

tipo	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageAttributes
		arn:aws:sns:us-east-1:1111111111:my-topic				naws.com/?action=Annulla l'iscrizione&SubscriptionArn=arn:aws:sns:us-east-1:1111111111:my-topic:326deeeb-cb f4-45da-b92b-ca77a247813b	gl","Value":\my_value\}}

tipo	messageId	topicArn	subject	message	timestamp	unsubscribeURL (URL annullamento sottoscrizione)	messageAttributes
Notificazione	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=NullifySubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeebcbf4-45dab92b-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 messaggi Amazon SNS nelle destinazioni Amazon Redshift](#).

Analisi dei messaggi Amazon SNS archiviati nelle destinazioni Amazon Redshift

Questo argomento descrive come analizzare i messaggi Amazon SNS inviati tramite flussi di distribuzione verso destinazioni Amazon Redshift.

Per analizzare i messaggi SNS inviati tramite i flussi di distribuzione Firehose verso destinazioni Amazon Redshift

1. Configura le risorse Amazon Redshift. Per istruzioni, consulta [Nozioni di base su Amazon Redshift](#) nella Guida alle operazioni di Amazon Redshift.
2. Configura il flusso di consegna. Per istruzioni, consulta [Scegli Amazon Redshift per la tua destinazione](#) nella Amazon Data Firehose Developer Guide.
3. Eseguire una query. Per ulteriori informazioni, consulta [Esecuzione di query su un database con l'editor di query](#) nella Guida alla gestione di Amazon Redshift.

Query di esempio

Per questa query di esempio, supponiamo quanto segue:

- I messaggi vengono archiviati nella tabella `notifications` nello schema predefinito `public`.
- La proprietà `Timestamp` del messaggio SNS viene memorizzata nella colonna `timestamp` della tabella con un tipo `timestampz` di dati colonna.

Note

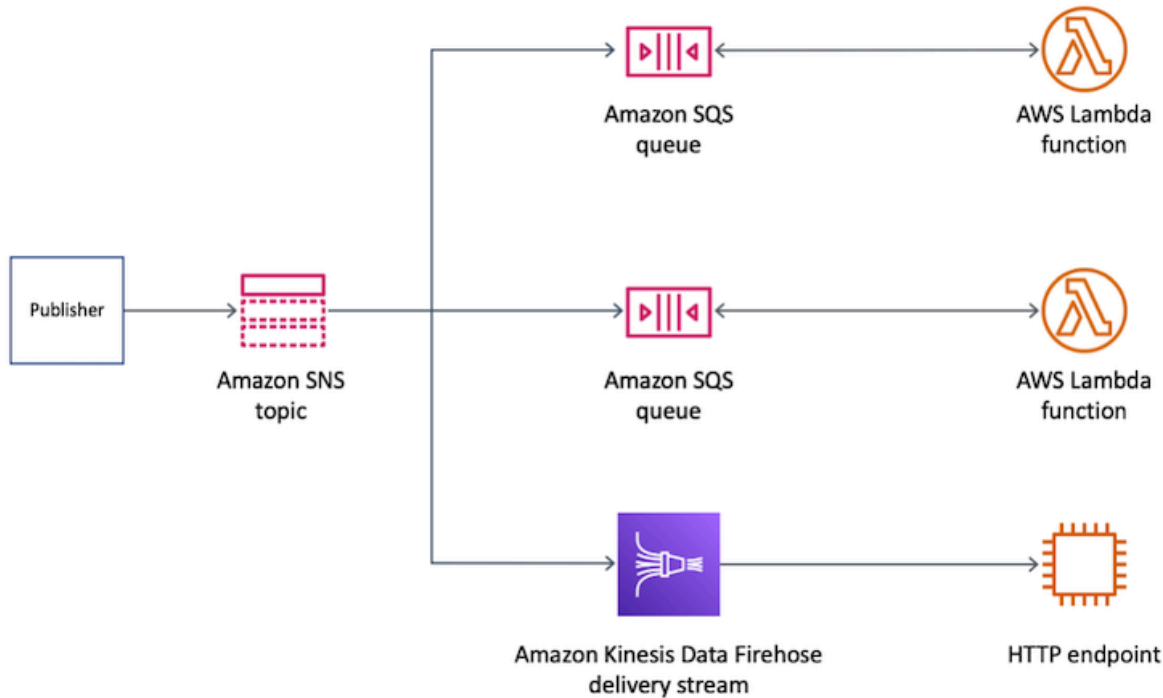
Per trasformare i metadati JSON per l'endpoint Amazon Redshift, puoi utilizzare il comando SQL `COPY`. Per ulteriori informazioni, consulta [Esempi di copia da JSON](#) e [Caricamento da dati JSON utilizzando l'opzione "auto ignorecase"](#) nella Guida per sviluppatori di Amazon Redshift.

La query seguente restituisce tutti i messaggi SNS ricevuti nell'intervallo di date specificato:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
  '2020-12-02T09:00:00.000Z';
```

Configurazione del recapito dei messaggi Amazon SNS a destinazioni HTTP utilizzando

Questo argomento spiega come i flussi di distribuzione pubblicano i dati sugli endpoint HTTP.



Argomenti

- [Formato di notifica Amazon SNS per la consegna a destinazioni HTTP](#)

Formato di notifica Amazon SNS per la consegna a destinazioni HTTP

Ecco un esempio di corpo di richiesta HTTP POST da Amazon SNS, inviato tramite un flusso di distribuzione a un endpoint HTTP. La notifica Amazon SNS è codificata come payload base64 all'interno della proprietà records.

Note

In questo esempio, il recapito dei messaggi non elaborati è disabilitato per il messaggio pubblicato. Per ulteriori informazioni sulla distribuzione non elaborata, consulta [Consegna di messaggi non elaborati Amazon SNS](#).

```
"body": {
  "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
  "timestamp": 1606255960435,
  "records": [
    {
      "data":
"eyJUeXB1IjoiTm90aWZpY2F0aW9uIiwiaWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFFhYzg0M"
    }
  ]
}
```

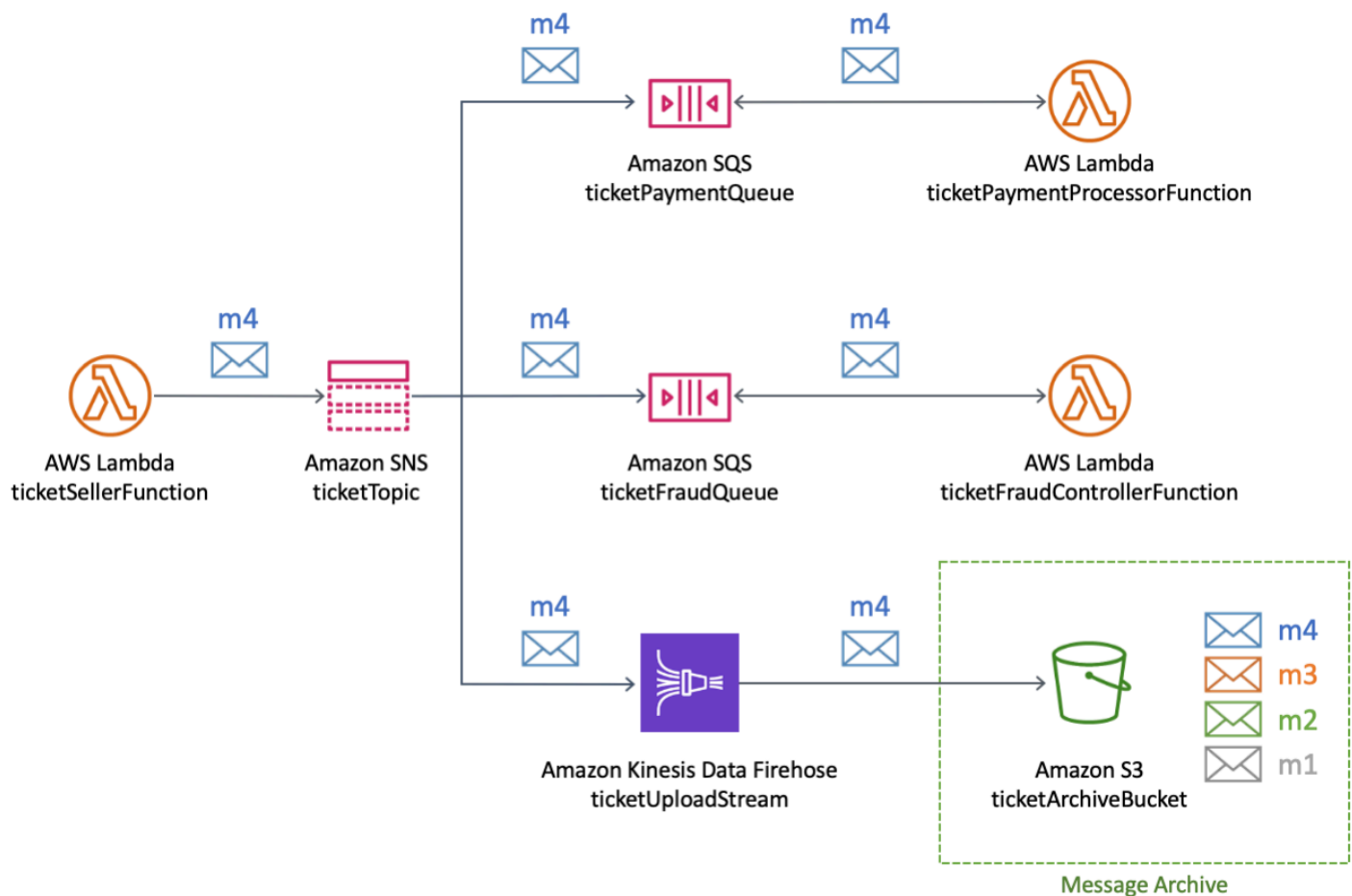
Archiviazione e analisi dei messaggi Amazon SNS: un esempio di utilizzo per le piattaforme di biglietteria aerea

Questo argomento fornisce un tutorial per un caso d'uso comune di archiviazione e analisi dei messaggi Amazon SNS.

L'impostazione di questo caso d'uso è una piattaforma di biglietteria aerea che opera in un ambiente regolamentato.

1. La piattaforma è soggetta a un framework di conformità che richiede all'azienda di archiviare tutte le vendite dei biglietti per almeno cinque anni.
2. Per raggiungere l'obiettivo di conformità relativo alla conservazione dei dati, l'azienda sottoscrive un flusso di distribuzione su un argomento esistente di Amazon SNS.
3. La destinazione per il flusso di distribuzione è un bucket Amazon Simple Storage Service (Amazon S3). Con questa configurazione, tutti gli eventi pubblicati nell'argomento SNS vengono archiviati nel bucket Amazon S3.

Nel diagramma seguente viene illustrata l'architettura di questa configurazione.



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

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

Argomenti

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

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

Questo argomento descrive come creare le risorse necessarie per il [caso d'uso di esempio di archiviazione e analisi dei messaggi](#):

- Un bucket Amazon Simple Storage Service (Amazon S3).
- Due code di Amazon Simple Queue Service (Amazon SQS)
- Un argomento Amazon SNS
- Due abbonamenti Amazon SQS all'argomento Amazon SNS

Per creare le risorse iniziali

1. Crea il bucket Amazon S3:

- a. Aprire la [console Amazon S3](#).
- b. Scegliere Create bucket (Crea bucket).
- c. In Bucket name (Nome bucket), immettere un nome univoco globale. Mantenere gli altri campi come valori predefiniti.
- d. Scegliere Create bucket (Crea bucket).

Per ulteriori informazioni sui bucket Amazon S3, consulta [Creazione di un bucket](#) nella Guida per l'utente di Amazon Simple Storage Service e [Utilizzo dei bucket Amazon S3](#) nella Guida per l'utente di Amazon Simple Storage Service.

2. Crea le due code Amazon SQS:

- a. Apri la [console Amazon SQS](#).
- b. Scegliere Crea coda.
- c. Per Tipo, scegliere Standard.
- d. In Nome, inserisci **ticketPaymentQueue**.
- e. Su Policy di accesso, per Scegli il metodo, scegliere Avanzato.
- f. Nella casella Criteri JSON, incollare la policy seguente:

JSON

```
{
  "Version": "2012-10-17",
  "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 modifica la AWS regione (**us-east-1**) di conseguenza.

- g. Scegliere Crea coda.
- h. Ripetere questi passaggi per creare una seconda coda SQS denominata **ticketFraudQueue**.

Per ulteriori informazioni sulla creazione delle code SQS, consulta [Creazione di una coda Amazon SQS \(console\)](#) nella Guida per sviluppatori Amazon Simple Queue Service.

3. Creazione dell'argomento SNS:
 - a. Aprire la [pagina Topics \(Argomenti\)](#) nella console Amazon SNS.
 - b. Scegliere Create topic (Crea argomento).
 - c. Su Details (Dettagli), per Type (Tipo), scegliere Standard.
 - d. Per Nome, immetti **ticketTopic**.

- e. Scegliere Create topic (Crea argomento).

Per ulteriori informazioni sulla creazione di argomenti SNS, consulta [Creare un argomento Amazon SNS](#).

4. Sottoscrizione della coda SQS all'argomento SNS:

- a. Nella [Console Amazon SNS](#), nella pagina ticketTopic dei dettagli dell'argomento, scegli Creazione di una sottoscrizione.
- b. Su Details (Dettagli), per Protocol (Protocollo), scegliere Amazon SQS.
- c. Per Endpoint, scegli l'Amazon Resource Name (ARN) della ticketPaymentQueue coda.
- d. Scegli Create Subscription (Crea sottoscrizione).
- e. Ripeti questi passaggi per creare un secondo abbonamento utilizzando l'ARN della ticketFraudQueue coda.

Per ulteriori informazioni su come sottoscrivere argomenti SNS, consulta [Creazione di un abbonamento a un argomento di Amazon SNS](#). Puoi anche sottoscrivere code SQS agli argomenti SNS dalla console Amazon SQS. Per ulteriori informazioni, consulta [Sottoscrizione di una coda Amazon SQS a un argomento Amazon SNS \(console\)](#) nella Guida per sviluppatori di Amazon Simple Queue Service.

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

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

Questo argomento spiega 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 Amazon Data Firehose

1. Apri la console dei [servizi Amazon Data Firehose](#).
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 Amazon Data Firehose, scegli il flusso di distribuzione che hai appena creato (). ticketUploadStream Nella tabella Dettagli, annota l'Amazon Resource Name (ARN) del relativo flusso per un momento successivo.

Per ulteriori informazioni sulla creazione di flussi di distribuzione, consulta [Creating an Amazon Data Firehose Delivery Stream nella Amazon Data Firehose Developer Guide](#). Per ulteriori informazioni sulla creazione di ruoli IAM, consulta [Creating a role to delegate permissions to an AWS service](#) nella IAM User Guide.

Hai creato il flusso di distribuzione di Firehose con le autorizzazioni richieste. Per continuare, consulta [Sottoscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS](#).

Sottoscrizione dello stream di distribuzione di Firehose all'argomento Amazon SNS

Questo argomento spiega come creare le seguenti risorse per il [caso d'uso di esempio di archiviazione e analisi dei messaggi](#):

- Il ruolo AWS Identity and Access Management (IAM) che consente all'abbonamento Amazon SNS di inserire record nel flusso di distribuzione.
- L'abbonamento Firehose Delivery Stream all'argomento Amazon SNS.

Per creare il ruolo IAM per l'abbonamento Amazon SNS

1. Aprire la [pagina Roles \(Ruoli\)](#) della console IAM.
2. Scegliere Create role (Crea ruolo).
3. Per Select type of trusted entity (Seleziona tipo di entità attendibile), seleziona AWS service (Servizio AWS).
4. Per Scegli un caso d'uso, scegliere SNS. Quindi scegliere Next: Permissions (Successivo: Autorizzazioni).
5. Scegliere Next: Tags (Successivo: Tag).
6. Scegliere Next:Review (Successivo: Rivedi).
7. Nella pagina Review (Rivedi), per Role name (Nome ruolo), immettere **ticketUploadStreamSubscriptionRole**. Quindi selezionare Create role (Crea ruolo).
8. Quando il ruolo viene creato, scegli il suo nome () ticketUploadStreamSubscriptionRole.
9. Nella pagina Riepilogo del ruolo, scegliere Aggiunta di policy inline.
10. Alla pagina Create policy (Crea policy), selezionare la scheda JSON, poi copiare la policy nella casella di testo.

JSON

```
{
  "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 modifica la AWS Region (**us-east-1**) di conseguenza.

11. Scegliere Review policy (Esamina policy).
12. Nella pagina Review policy (Esamina policy), per Name (Nome), immettere **FirehoseSnsPolicy**. Quindi scegliere Create policy (Crea policy).
13. Nella pagina Riepilogo del ruolo, prendere nota dell'ARN ruolo per dopo.

Per ulteriori informazioni sulla creazione di ruoli IAM, consulta [Creare un ruolo per delegare le autorizzazioni a un AWS servizio](#) nella Guida per l'utente IAM.

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

1. Aprire la [pagina Topics \(Argomenti\)](#) nella console Amazon SNS.
2. Nella tabella Subscriptions (Sottoscrizioni) scegliere Create subscription (Crea sottoscrizione).
3. In Dettagli, per Protocollo, scegli.
4. Per Endpoint, inserisci l'Amazon Resource Name (ARN) ticketUploadStreamdel flusso di distribuzione creato in precedenza. Ad esempio, specifica **arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream**.
5. Per ARN del ruolo di sottoscrizione, inserisci l'ARN del ruolo ticketUploadStreamSubscriptionRoleIAM che hai creato in precedenza. Ad esempio, specifica **arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole**.
6. Selezionare l'Abilitazione del recapito di messaggi.
7. Scegli Create Subscription (Crea sottoscrizione).

È stato creato il ruolo IAM e la sottoscrizione all'argomento SNS. Per continuare, consulta [Test e interrogazione di una configurazione Amazon SNS per una gestione efficace dei dati](#).

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

Questo argomento spiega come testare il [caso d'uso di esempio di archiviazione e analisi dei messaggi](#) pubblicando un messaggio sull'argomento Amazon SNS. Le istruzioni includono una query di esempio che è possibile eseguire e adattare alle proprie esigenze.

Per testare la configurazione

1. Aprire la [pagina Topics \(Argomenti\)](#) nella console Amazon SNS.
2. Seleziona l'argomento **ticketTopic**.
3. Seleziona Publish message (Pubblica messaggio).
4. Nella pagina Pubblica il messaggio nell'argomento inserisci quanto segue per il corpo del messaggio. Aggiungi un carattere di nuova riga alla fine del messaggio.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15  
04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

Mantenere tutte le altre opzioni come valori predefiniti.

5. Seleziona Publish message (Pubblica messaggio).

Per ulteriori informazioni sulla pubblicazione dei messaggi, consulta [Pubblicazione di un messaggio Amazon SNS](#).

6. Dopo l'intervallo di flusso di consegna di 60 secondi, aprire la finestra [Console Amazon Simple Storage Service \(Amazon S3\)](#) e scegliere il bucket Amazon S3 [creato inizialmente](#).

Il messaggio pubblicato appare nel bucket.

Per eseguire una query sui dati

1. Aprire la [console Amazon Athena](#).
2. Eseguire una query.

Ad esempio, si supponga che la tabella `notifications` nello schema `default` contenga i seguenti dati:

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15  
04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15  
11:30:15","Destination":"Miami","FlyingFrom":"Omaha","TicketNumber":"efgh5678"}  
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15  
3:30:10","Destination":"Miami","FlyingFrom":"NewYork","TicketNumber":"ijkl9012"}  
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15  
12:30:05","Destination":"Delhi","FlyingFrom":"Omaha","TicketNumber":"mnop3456"}
```

Per trovare la destinazione principale, eseguire la seguente query:

```
SELECT destination  
FROM default.notifications  
GROUP BY destination  
ORDER BY count(*) desc  
LIMIT 1;
```

Per eseguire una query per i ticket venduti in un intervallo di data e ora specifico, eseguire una query simile alla seguente:

```
SELECT *  
FROM default.notifications  
WHERE bookingtime  
    BETWEEN TIMESTAMP '2020-12-15 10:00:00'  
    AND TIMESTAMP '2020-12-15 12:00:00';
```

È possibile adattare entrambe le query di esempio per le proprie esigenze. Per ulteriori informazioni sull'utilizzo di Athena per eseguire query, consulta [Nozioni di base](#) nella Guida per l'utente di Amazon Athena.

Pulizia

Per evitare di incorrere in costi di utilizzo dopo aver terminato il test, eliminare le seguenti risorse create durante l'esercitazione:

- Abbonamenti Amazon SNS
- Argomento Amazon SNS
- Code di Amazon Simple Queue Service (Amazon SQS)
- Bucket Amazon S3
- flusso di consegna

- AWS Identity and Access Management ruoli e politiche (IAM)

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

Per automatizzare l'implementazione [caso d'uso d'esempio di archiviazione dei messaggi e analisi dei dati](#) di Amazon SNS, è possibile utilizzare il seguente modello YAML:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Template for creating an SNS archiving use case
Resources:
  ticketUploadStream:
    DependsOn:
      - ticketUploadStreamRolePolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN: !Sub 'arn:${AWS::Partition}:s3:::${ticketArchiveBucket}'
        BufferingHints:
          IntervalInSeconds: 60
          SizeInMBs: 1
        CompressionFormat: UNCOMPRESSED
        RoleARN: !GetAtt ticketUploadStreamRole.Arn
  ticketArchiveBucket:
    Type: AWS::S3::Bucket
  ticketTopic:
    Type: AWS::SNS::Topic
  ticketPaymentQueue:
    Type: AWS::SQS::Queue
  ticketFraudQueue:
    Type: AWS::SQS::Queue
  ticketQueuePolicy:
    Type: AWS::SQS::QueuePolicy
    Properties:
      PolicyDocument:
        Statement:
          Effect: Allow
          Principal:
            Service: sns.amazonaws.com
          Action:
            - sqs:SendMessage
```

```

    Resource: '*'
    Condition:
      ArnEquals:
        aws:SourceArn: !Ref ticketTopic
  Queues:
    - !Ref ticketPaymentQueue
    - !Ref ticketFraudQueue
ticketUploadStreamSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketUploadStream.Arn
    Protocol: firehose
    SubscriptionRoleArn: !GetAtt ticketUploadStreamSubscriptionRole.Arn
ticketPaymentQueueSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketPaymentQueue.Arn
    Protocol: sqs
ticketFraudQueueSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketFraudQueue.Arn
    Protocol: sqs
ticketUploadStreamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Sid: ''
          Effect: Allow
          Principal:
            Service: firehose.amazonaws.com
          Action: sts:AssumeRole
ticketUploadStreamRolePolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: FirehoseTicketUploadStreamRolePolicy
    PolicyDocument:
      Version: '2012-10-17'
      Statement:

```



```

    - Effect: Allow
    Action:
      - s3:AbortMultipartUpload
      - s3:GetBucketLocation
      - s3:GetObject
      - s3:ListBucket
      - s3:ListBucketMultipartUploads
      - s3:PutObject
    Resource:
      - !Sub 'arn:aws:s3:::${ticketArchiveBucket}'
      - !Sub 'arn:aws:s3:::${ticketArchiveBucket}/*'
  Roles:
    - !Ref ticketUploadStreamRole
ticketUploadStreamSubscriptionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - sns.amazonaws.com
          Action:
            - sts:AssumeRole
    Policies:
      - PolicyName: SNSKinesisFirehoseAccessPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Action:
                - firehose:DescribeDeliveryStream
                - firehose:ListDeliveryStreams
                - firehose:ListTagsForDeliveryStream
                - firehose:PutRecord
                - firehose:PutRecordBatch
              Effect: Allow
              Resource:
                - !GetAtt ticketUploadStream.Arn

```

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

Amazon SNS si integra con AWS Lambda, consentendoti di attivare le funzioni Lambda in risposta alle notifiche di Amazon SNS. Quando un messaggio viene pubblicato su un argomento di Amazon SNS 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, pubblicare il messaggio su altri argomenti di Amazon SNS o inviarlo ad altri servizi. AWS

Inoltre, Amazon SNS supporta gli attributi relativi allo stato di consegna dei messaggi per le notifiche dei messaggi inviate agli endpoint Lambda. Per ulteriori informazioni, consulta [Stato di consegna dei messaggi Amazon SNS](#).

Argomenti

- [Prerequisiti per l'integrazione di Amazon SNS con le funzioni Lambda in tutte le regioni](#)
- [Sottoscrizione di una funzione Lambda a un argomento di Amazon SNS](#)

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

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

- Una funzione Lambda
- Un argomento di Amazon SNS

Per informazioni sulla creazione di una funzione Lambda da utilizzare con Amazon SNS, consulta [Uso di Lambda con Amazon SNS](#). Per ulteriori informazioni sulla creazione di un argomento Amazon SNS, consultare [Creazione di un argomento](#).

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

Ad esempio, se desideri sottoscrivere una funzione Lambda negli Stati Uniti orientali (Virginia settentrionale) a un argomento di Amazon SNS in Asia Pacifico (Hong Kong), modifica il principale

nella politica della funzione 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 messaggi Amazon SNS da regioni opt-in ad altre regioni opt-in non è supportato.

Sottoscrizione di una funzione Lambda a un argomento di Amazon SNS

Questo argomento spiega come sottoscrivere una funzione Lambda a un argomento Amazon SNS, abilitando l'attivazione della funzione dai messaggi pubblicati.

1. Accedi alla [console Amazon SNS](#).
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. Verificare il valore Topic ARN (ARN argomento) scelto.
 - b. Per Protocol scegli. AWS Lambda
 - c. Per Endpoint immettere l'ARN di una funzione.
 - d. Scegli Create Subscription (Crea sottoscrizione).

Quando un messaggio viene pubblicato su un argomento di Amazon SNS 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 Amazon SNS, incluso un tutorial, consulta [Using AWS Lambda with Amazon SNS](#).

Esegui l'eliminazione delle notifiche Amazon SNS alle code di Amazon SQS per l'elaborazione asincrona

[Amazon SNS](#) funziona a stretto contatto con Amazon Simple Queue Service (Amazon SQS). Questi servizi offrono numerosi vantaggi agli sviluppatori. Amazon SNS consente alle applicazioni di inviare messaggi con vincoli temporistici a più sottoscrittori grazie a un meccanismo "push", eliminando la necessità di cercare periodicamente gli aggiornamenti o di "eseguirne il polling". Amazon SQS; è un servizio di accodamento di messaggi utilizzato dalle applicazioni distribuite per scambiare messaggi mediante un modello di polling. Può essere utilizzato per separare i componenti di invio e ricezione, senza richiedere la disponibilità simultanea di tutti i componenti. Grazie all'integrazione di Amazon SNS con Amazon SQS, è possibile recapitare i messaggi alle applicazioni che richiedono la notifica immediata di un evento e renderli persistenti in una coda Amazon SQS in modo da essere elaborati in seguito da altre applicazioni.

Quando esegui la sottoscrizione di una coda Amazon SQS; a un argomento Amazon SNS, puoi pubblicare un messaggio nell'argomento e Amazon SNS invia un messaggio Amazon SQS alla coda per la quale è stata eseguita la sottoscrizione. Il messaggio Amazon SQS contiene l'oggetto e il messaggio pubblicati nell'argomento insieme ai metadati relativi al messaggio in un documento JSON. Il messaggio Amazon SQS sarà simile al documento JSON seguente.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:MyTopic:c7fe3a54-
ab0e-4ec2-88e0-db410a0f2bee"
}
```

Iscrizione di una coda Amazon SQS a un argomento Amazon SNS

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

- Utilizzo del [Console Amazon SQS](#), che semplifica il processo. Per ulteriori informazioni, consulta Tutorial: [Sottoscrizione di una coda Amazon SQS a un argomento Amazon SNS](#) nella Guida per sviluppatori di Amazon Simple Queue Service.
- Utilizza le fasi seguenti:
 1. [Ottieni l'Amazon Resource Name \(ARN\) della coda a cui desideri inviare i messaggi e l'argomento a cui intendi sottoscrivere la coda.](#)
 2. [Concedi l'autorizzazione sqs : SendMessage all'argomento Amazon SNS in modo che possa inviare messaggi alla coda.](#)
 3. [Sottoscrizione della coda all'argomento Amazon SNS.](#)
 4. [Concedi agli utenti IAM o agli Account AWS le autorizzazioni appropriate per pubblicare nell'argomento Amazon SNS e leggere i messaggi nella coda Amazon SQS.](#)
 5. [Verifica la procedura pubblicando un messaggio nell'argomento e leggendo il messaggio dalla coda.](#)

Per informazioni sulla configurazione di un argomento per l'invio di messaggi a una coda che si trova in un account AWS differente, consultare [Invio di messaggi Amazon SNS a una coda Amazon SQS; in un altro account.](#)

Per visualizzare un CloudFormation modello che crea un argomento che invia messaggi a due code, consulta. [Automatizza la messaggistica da Amazon SNS ad Amazon SQS con AWS CloudFormation](#)

Fase 1: ottenere l'ARN della coda e dell'argomento

Quando esegui la sottoscrizione di una coda al tuo argomento, necessiti di una copia dell'ARN della coda. Analogamente, quando autorizzi l'argomento a inviare messaggi, necessiti di una copia dell'ARN dell'argomento.

Per ottenere l'ARN della coda, puoi utilizzare la console Amazon SQS o l'azione API. [GetQueueAttributes](#)

Ottenimento dell'ARN della coda dalla console Amazon SQS

1. Accedi Console di gestione AWS e apri la console Amazon SQS all'indirizzo. <https://console.aws.amazon.com/sqs/>
2. Seleziona la casella per la coda di cui intendi ottenere l'ARN.
3. Nella scheda Details (Dettagli), copia il valore dell'ARN in modo da utilizzarlo per la sottoscrizione all'argomento Amazon SNS.

Per ottenere l'ARN dell'argomento, puoi utilizzare la console Amazon SNS, il comando [sns-get-topic-attributes](#) o l'operazione API [GetQueueAttributes](#).

Ottenimento dell'ARN dell'argomento dalla console Amazon SNS

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, scegliere l'argomento di cui si desidera ottenere l'ARN.
3. Nella sezione Details (Dettagli), copiare il valore di ARN in modo da poterlo utilizzare per autorizzare l'argomento Amazon SNS a inviare messaggi alla coda.

Fase 2: Concedere all'argomento Amazon SNS l'autorizzazione a inviare messaggi alla coda Amazon SQS

Per consentire a un argomento Amazon SNS di inviare messaggi a una coda, devi definire una policy sulla coda che consenta all'argomento Amazon SNS di eseguire l'operazione `sqs:SendMessage`.

Prima di eseguire la sottoscrizione di una coda a un argomento, devi creare un argomento e una coda. Se non lo hai già fatto, creali adesso. Per ulteriori informazioni, consultare [Creazione di un argomento](#) e [Creare una coda](#) nella Guida per sviluppatori di Amazon Simple Queue Service.

Per impostare una policy su una coda, puoi utilizzare la console Amazon SQS o [SetQueueAttributes](#) l'azione API. Prima di iniziare, assicurati di avere l'ARN dell'argomento a cui intendi concedere l'autorizzazione a inviare messaggi alla coda. Se stai sottoscrivendo una coda a più argomenti, la policy deve contenere un elemento `Statement` per ogni argomento.

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

1. Accedi Console di gestione AWS e apri la console Amazon SQS all'indirizzo. <https://console.aws.amazon.com/sqs/>

2. Seleziona la casella della coda per la quale intendi impostare la policy, scegli la scheda Policy di accesso, quindi scegli Modifica.
3. Nella Policy di accesso, definire chi può accedere alla coda.
 - Aggiungi una condizione che autorizza l'operazione per l'argomento.
 - Impostare `Principal` come servizio Amazon SNS, come mostrato nell'esempio seguente.
 - Utilizzare le chiavi di condizione globali [aws:SourceArn](#) o [aws:SourceAccount](#) per proteggersi dallo scenario [Confused deputy](#). Per utilizzare queste chiavi di condizione, impostare il valore sull'ARN del proprio argomento. Se la coda è sottoscritta a più argomenti, è possibile usare invece `aws:SourceAccount`.

Ad esempio, la seguente politica consente di MyTopic 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"
        }
      }
    }
  ]
}
```

Fase 3: eseguire la sottoscrizione della coda all'argomento Amazon SNS

Per inviare messaggi a una coda tramite un argomento, devi eseguire la sottoscrizione della coda all'argomento Amazon SNS. La coda viene specificata mediante il relativo ARN. Per effettuare la sottoscrizione a un argomento, puoi utilizzare la console Amazon SNS, il comando della CLI [sns-subscribe](#) o l'operazione API [Subscribe](#). Prima di iniziare, assicurati di avere l'ARN della coda per la quale intendi eseguire la sottoscrizione.

1. Accedi a [Amazon SNS console](#)
2. Nel pannello di navigazione, scegliere Argomenti.
3. Nella pagina Topics (Argomenti) scegliere un argomento.
4. Nella **MyTopic** pagina, nella pagina Abbonamenti, scegli Crea abbonamento.
5. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
 - a. Verificare il valore Topic ARN (ARN argomento).
 - b. Per Protocollo, scegliere Amazon SQS.
 - c. Per Endpoint immettere l'ARN di una coda Amazon SQS.
 - d. Scegli Create Subscription (Crea sottoscrizione).

Dopo la conferma della sottoscrizione, il campo Subscription ID (ID sottoscrizione) della nuova sottoscrizione visualizza il relativo ID. Se il proprietario della coda crea la sottoscrizione, questa viene automaticamente confermata e dovrebbe essere attiva quasi immediatamente.

In genere, esegui la sottoscrizione della tua coda al tuo argomento nel tuo account. Tuttavia, puoi anche eseguire la sottoscrizione di una coda in un altro account al tuo argomento. Se l'utente che crea la sottoscrizione non è il proprietario della coda (ad esempio, se un utente dell'account A esegue la sottoscrizione di una coda nell'account B a un argomento nell'account A), la sottoscrizione deve essere confermata. Per ulteriori informazioni sulla sottoscrizione di una coda in un account differente e sulla conferma della sottoscrizione, consulta [Invio di messaggi Amazon SNS a una coda Amazon SQS; in un altro account](#).

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

È necessario utilizzare AWS Identity and Access Management (IAM) per consentire solo agli utenti appropriati di pubblicare sull'argomento Amazon SNS e sui read/delete messaggi dalla coda Amazon SQS. Per ulteriori informazioni sul controllo delle operazioni sugli argomenti e le code per gli utenti IAM, consulta [Utilizzo di policy basate su identità con Amazon SNS](#), e [Identity and Access Management in Amazon SQS](#) nella Guida per sviluppatori di Amazon Simple Queue Service.

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

- [Aggiungere una policy a un utente o gruppo IAM](#). Il modo più semplice di concedere agli utenti le autorizzazioni per argomenti o code è di creare un gruppo e aggiungere a quel gruppo dapprima

la policy appropriata e quindi gli utenti. È molto più semplice aggiungere e rimuovere utenti da un gruppo anziché tenere traccia delle policy impostate su singoli utenti.

- [Aggiungere una policy a un argomento o a una coda](#). Se desideri concedere autorizzazioni a un argomento o creare una coda a un altro AWS account, l'unico modo per farlo è aggiungere una policy che abbia come principale quella a cui Account AWS 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.

Aggiunta di una policy a un utente o gruppo IAM

Se aggiungessi la seguente policy a un utente o gruppo IAM, daresti a quell'utente o ai membri di quel gruppo il permesso di eseguire l'`sns:Publish` sull'argomento `MyTopic`.

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

Se aggiungessi la seguente policy a un utente o gruppo IAM, daresti a quell'utente o ai membri di quel gruppo il permesso di 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 utenti IAM che dispongono di autorizzazioni di accesso a livello di amministratore (accesso con wildcard) a quella risorsa. L'accesso alla risorsa viene automaticamente negato a tutti gli altri utenti IAM nell'altro account. Se intendi concedere l'accesso alla risorsa a specifici utenti IAM in quel Account AWS, l'account o un utente IAM con accesso di livello amministratore deve delegare le autorizzazioni per la risorsa a quegli utenti IAM. Per ulteriori informazioni sulla delega multiaccount, consulta la sezione relativa all'[abilitazione dell'accesso multiaccount](#) nella Guida all'uso di IAM.

Se 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 `sqs:DeleteMessage` e `sqs:ReceiveMessage` su quella coda.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Fase 5: eseguire la verifica delle sottoscrizioni della coda all'argomento

Puoi eseguire la verifica delle sottoscrizioni di una coda a un argomento pubblicando nell'argomento e visualizzando il messaggio che l'argomento invia alla coda.

Per pubblicare un argomento utilizzando la console Amazon SNS

1. Utilizzando le credenziali dell'utente Account AWS o dell'utente IAM con autorizzazione alla pubblicazione sull'argomento, accedi Console di gestione AWS e apri la console Amazon SNS 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).

Visualizzazione del messaggio dall'argomento utilizzando la console Amazon SQS

1. Utilizzando le credenziali dell'utente Account AWS o dell'utente IAM con autorizzazione a visualizzare i messaggi in coda, accedi Console di gestione AWS e apri la console Amazon SQS 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 Message Details (Dettagli messaggio) contiene un documento JSON con l'oggetto e il messaggio che hai pubblicato nell'argomento. Il messaggio risulta simile al documento JSON seguente.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-db410a0f2bee"
}
```

5. Scegliere Close (Chiudi). La pubblicazione in un argomento che invia messaggi di notifica a una coda è completata.

Automatizza la messaggistica da Amazon SNS ad Amazon SQS con AWS CloudFormation

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 eseguono automaticamente la procedura di configurazione creando due code e un argomento con sottoscrizioni

alle code, aggiungendo una policy alle code affinché l'argomento possa inviare messaggi alle code e creando utenti e gruppi IAM per controllare l'accesso a tali risorse.

Per ulteriori informazioni sulla distribuzione AWS delle risorse utilizzando un CloudFormation modello, consulta la Guida [introduttiva](#) nella Guida per l'CloudFormation utente.

Utilizzo di un CloudFormation modello per configurare argomenti e code all'interno di un Account AWS

Il modello di esempio crea un argomento Amazon SNS in grado di inviare messaggi a due code Amazon SQS con autorizzazioni appropriate per consentire ai membri di un gruppo IAM di pubblicare nell'argomento e a un altro gruppo di leggere messaggi dalle code. Il modello crea inoltre utenti IAM che vengono aggiunti a ogni gruppo.

Copiare il contenuto del modello in un file. [Puoi anche scaricare il modello dalla pagina Modelli.AWS CloudFormation](#) Nella pagina dei modelli, scegli Sfoglia modelli di esempio per AWS servizio, quindi scegli Amazon Simple Queue Service.

My SNSTopic è configurato per pubblicare su due endpoint sottoscritti, che sono due code Amazon SQS MyQueue (1 e 2). MyQueue MyPublishTopicGroup [è un gruppo IAM i cui membri sono autorizzati a pubblicare su My SNSTopic utilizzando l'azione Publish API o il comando sns-publish.](#)

Il modello crea gli utenti IAM 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 utenti IAM con MyPublishUserKey e MyQueueUserKey. AddUserToMyPublishTopicGroup si aggiunge MyPublishUser a MyPublishTopicGroup in modo che all'utente vengano assegnate le autorizzazioni al gruppo.

My RDMMessage QueueGroup è un gruppo IAM i cui membri sono autorizzati a leggere ed eliminare messaggi dalle due code di Amazon SQS utilizzando le azioni [ReceiveMessage](#) e [DeleteMessageAPI](#). AddUserToMyQueueGroup si aggiunge MyQueueUser a My RDMMessage QueueGroup in modo che all'utente vengano assegnate le autorizzazioni al gruppo. MyQueuePolicy assegna il permesso SNSTopic a My di pubblicare le sue notifiche nelle due code.

L'elenco seguente mostra il contenuto del 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 delle notifiche Amazon SNS agli endpoint HTTPS

Puoi utilizzare Amazon SNS per inviare messaggi di notifica a uno o più endpoint HTTP o HTTPS. Quando effettui la sottoscrizione di un endpoint a un argomento, puoi pubblicare una notifica nell'argomento e Amazon SNS provvederà a inviare una richiesta HTTP POST per consegnare il contenuto della notifica all'endpoint dotato di sottoscrizione. Quando effettui la sottoscrizione dell'endpoint, scegli se Amazon SNS deve utilizzare HTTP o HTTPS per inviare la richiesta POST all'endpoint. Se utilizzi HTTPS, puoi sfruttare il supporto di Amazon SNS per quanto segue:

- **Server Name Indication (SNI):** consente a Amazon SNS di supportare gli endpoint HTTPS che richiedono l'estensione di protocollo SNI, ad esempio un server che richiede più certificati per ospitare più domini. Per ulteriori informazioni sull'uso di SNI, consulta [Server Name Indication \(SNI\)](#).
- **Autenticazione di accesso di base e digest—** Questo consente di specificare un nome utente e una password nell'URL HTTPS della richiesta HTTP POST, come `https://user:password@domain.com` o `https://user@domain.com`. Il nome utente e la password vengono crittografati nella connessione SSL stabilita quando usi HTTPS. Solo il nome di dominio viene inviato come testo normale. Per ulteriori informazioni su Basic e Digest Access Authentication, consulta [RFC-2617](#).

Important

Amazon SNS attualmente non supporta endpoint HTTP(S) privati.
URLs Gli HTTPS sono recuperabili solo dall'azione dell'API Amazon
GetSubscriptionAttributes SNS, per i principali a cui hai concesso l'accesso all'API.

Note

Il servizio client deve essere in grado di supportare la risposta con intestazione HTTP/1.1 401 Unauthorized.

La richiesta contiene l'oggetto e il messaggio pubblicati nell'argomento insieme ai metadati relativi alla notifica in un documento JSON. La richiesta risulterà simile alla richiesta HTTP POST seguente. Per i dettagli sull'intestazione HTTP e il formato JSON del corpo della richiesta, consulta [Intestazioni HTTP/HTTPS](#) e [Notifica HTTP/HTTPS in formato JSON](#).

Note

Amazon SNS considera risolvibili tutti gli errori 5XX e 429 (troppe richieste inviate) errori. Questi errori sono soggetti alla politica di consegna. Tutti gli altri errori sono considerati errori permanenti e non verranno tentati nuovi tentativi.

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" :  
"EXAMPLE1DMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVS7Xn0n/49IkxDKz8Yr1H2qJXj2iZB0Zo2071c4"  
"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"  
}
```

Sottoscrizione di un endpoint HTTPS a un argomento di Amazon SNS

Questo argomento spiega come sottoscrivere gli HTTP/S endpoint agli argomenti di Amazon SNS.

Argomenti

- [Fase 1: Verifica della capacità dell'endpoint di elaborare messaggi Amazon SNS](#)
- [Fase 2: sottoscrivere l' HTTP/HTTPS endpoint all'argomento Amazon SNS](#)
- [Passaggio 3: conferma l'abbonamento ad Amazon SNS](#)
- [Passaggio 4: Facoltativo: imposta la politica di consegna per l'abbonamento Amazon SNS](#)
- [Passaggio 5: Facoltativo: concedere agli utenti le autorizzazioni per la pubblicazione sull'argomento Amazon SNS](#)
- [Passaggio 6: invio di messaggi Amazon SNS all'endpoint HTTP/HTTPS](#)

Fase 1: Verifica della capacità dell'endpoint di elaborare messaggi Amazon SNS

Prima di effettuare la sottoscrizione dell'endpoint HTTP o HTTPS a un argomento, devi verificare che l'endpoint sia in grado di gestire le richieste HTTP POST che Amazon SNS usa per inviare la conferma della sottoscrizione e i messaggi di notifica. In genere questo significa creare e distribuire un'applicazione Web (ad esempio, un servlet Java se l'host dell'endpoint esegue Linux con Apache e Tomcat) che elabori le richieste HTTP di Amazon SNS. Quando effettui la sottoscrizione di un endpoint HTTP, Amazon SNS invia una richiesta di conferma della sottoscrizione. L'endpoint deve essere preparato a ricevere ed elaborare la richiesta quando crei la sottoscrizione perché Amazon SNS la invia in quel momento. Amazon SNS non invierà notifiche all'endpoint fino alla conferma della sottoscrizione. Una volta confermata la sottoscrizione, Amazon SNS invierà le notifiche all'endpoint quando verrà eseguita un'operazione di pubblicazione sull'argomento sottoscritto.

Per impostare l'endpoint in modo che elabori i messaggi di conferma della sottoscrizione e di notifica

1. Il tuo codice deve leggere le intestazioni HTTP delle richieste HTTP POST che Amazon SNS invia all'endpoint. Il codice deve cercare il campo di intestazione `x-amz-sns-message-type` che indica il tipo di messaggio inviato da Amazon SNS. In base all'intestazione puoi determinare il tipo di messaggio senza dover analizzare il corpo della richiesta HTTP. Sono due i tipi che devi gestire: `SubscriptionConfirmation` e `Notification`. Il messaggio `UnsubscribeConfirmation` viene utilizzato solo quando la sottoscrizione viene eliminata dall'argomento.

Per i dettagli sull'intestazione HTTP, consulta [Intestazioni HTTP/HTTPS](#). La seguente richiesta HTTP POST è un esempio di messaggio di conferma della sottoscrizione.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37f...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH+...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

2. Il codice deve analizzare il documento JSON nel corpo della richiesta HTTP POST e nel tipo di contenuto `text/plain` per leggere le coppie nome-valore che costituiscono il messaggio Amazon SNS. Usa un parser JSON in grado di gestire la conversione della rappresentazione con escape dei caratteri di controllo nei corrispondenti valori ASCII (ad esempio, la conversione di `\n` in un carattere di nuova riga). Puoi utilizzare un parser JSON esistente come il [processore Jackson JSON](#) o scriverne uno personalizzato. Per inviare il testo nei campi dell'oggetto e del messaggio come codice JSON valido, Amazon SNS deve convertire alcuni caratteri di controllo in rappresentazioni con escape che possono essere incluse nel documento JSON. Quando ricevi il documento JSON nel corpo della richiesta POST inviata all'endpoint, devi riconvertire i caratteri con escape nei valori originali se vuoi una rappresentazione esatta dell'oggetto e dei messaggi originali pubblicati nell'argomento. Questo è un aspetto critico se desideri verificare la firma di una notifica, perché la firma usa il messaggio e l'oggetto nel formato originale come parte della stringa di firma.
3. Il codice deve verificare l'autenticità di un messaggio di notifica, di conferma della sottoscrizione o di conferma di annullamento della sottoscrizione inviato da Amazon SNS. Utilizzando le informazioni contenute nel messaggio Amazon SNS, l'endpoint può ricreare la firma per consentirti di verificare i contenuti del messaggio mettendo in corrispondenza la tua firma con la firma che Amazon SNS ha inviato con il messaggio. Per ulteriori informazioni sulla verifica della firma di un messaggio, consulta [Verifica delle firme dei messaggi Amazon SNS](#).
4. In base al tipo specificato dal campo di intestazione `x-amz-sns-message-type`, il codice deve leggere il documento JSON contenuto nel corpo della richiesta HTTP ed elaborare il messaggio. Di seguito sono riportate le linee guida per gestire i due tipi di messaggi principali:

SubscriptionConfirmation

Leggi il valore di `SubscribeURL` e visita l'URL. Per confermare la sottoscrizione e iniziare a ricevere notifiche presso l'endpoint, devi visitare l'`SubscribeURL` (ad esempio inviando all'URL una richiesta HTTP GET). Fai riferimento all'esempio di richiesta HTTP nella fase precedente per vedere l'aspetto di `SubscribeURL`. Per ulteriori informazioni sul formato del messaggio `SubscriptionConfirmation`, consulta [Conferma sottoscrizione HTTP/HTTPS in formato JSON](#). Quando visiterai l'URL, otterrai una risposta simile al documento XML seguente. Il documento restituisce l'ARN della sottoscrizione dell'endpoint nell'elemento `ConfirmSubscriptionResult`.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
  <ConfirmSubscriptionResult>
```



```
<SubscriptionArn>arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55</
SubscriptionArn>
</ConfirmSubscriptionResult>
<ResponseMetadata>
  <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
</ResponseMetadata>
</ConfirmSubscriptionResponse>
```

In alternativa alla visita del `SubscribeURL`, puoi confermare l'iscrizione utilizzando l'[ConfirmSubscription](#) Token impostando il valore corrispondente nel messaggio. `SubscriptionConfirmation` Se desideri che solo il proprietario dell'argomento e il proprietario della sottoscrizione possano annullare la sottoscrizione dell'endpoint, chiama l'operazione `ConfirmSubscription` con una firma AWS .

Notifica

Leggi i valori di `Subject` e `Message` per ottenere le informazioni sulla notifica pubblicate nell'argomento.

Per i dettagli sul formato del messaggio `Notification`, consulta [Intestazioni HTTP/HTTPS](#). La richiesta HTTP POST seguente è un esempio di messaggio di notifica inviato all'endpoint `example.com`.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
```

```
"Timestamp" : "2012-05-02T00:54:06.655Z",
"SignatureVersion" : "1",
"Signature" : "EXAMPLEw6JRN...",
"SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
"UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

5. Assicurati che l'endpoint risponda al messaggio HTTP POST ricevuto da Amazon SNS con il codice di stato appropriato. La connessione scadrà tra circa 15 secondi. Se l'endpoint non risponde prima del timeout della connessione o se restituisce un codice di stato non compreso tra 200 e 4xx, Amazon SNS considererà la consegna del messaggio come un tentativo fallito.
6. Assicurati che il codice possa gestire tentativi ripetuti di consegna del messaggio da parte di Amazon SNS. Se Amazon SNS non riceve una risposta positiva dall'endpoint, tenta di consegnare nuovamente il messaggio. Ciò vale per tutti i messaggi, incluso il messaggio di conferma della sottoscrizione. Per default, se la consegna iniziale del messaggio non riesce, Amazon SNS ritenta fino a tre volte, con un intervallo di 20 secondi tra tentativi non riusciti.

Note

La richiesta del messaggio scade dopo circa 15 secondi. Questo significa che, se la mancata consegna del messaggio è stata causata da un timeout, Amazon SNS riprova circa 35 secondi dopo il precedente tentativo non riuscito. È possibile impostare una policy di consegna diversa per l'endpoint.

Amazon SNS utilizza il campo di intestazione `x-amz-sns-message-id` per identificare in modo univoco ogni messaggio pubblicato su un argomento Amazon SNS. Confrontando IDs i messaggi elaborati con quelli in arrivo, è possibile determinare se si tratta di un nuovo tentativo.

7. Se sottoscrivi un endpoint HTTPS, assicurati che l'endpoint disponga di un certificato server di un'autorità di certificazione (CA) attendibile. Amazon SNS invia messaggi solo agli endpoint HTTPS che dispongono di un certificato server firmato da una CA che Amazon SNS ritiene attendibile.
8. Distribuisci il codice creato per ricevere i messaggi Amazon SNS. Quando effettui la sottoscrizione dell'endpoint, questo deve essere pronto a ricevere almeno il messaggio di conferma della sottoscrizione.

Fase 2: sottoscrivere l' HTTP/HTTPS endpoint all'argomento Amazon SNS

Per inviare messaggi a un endpoint HTTP o HTTPS tramite un argomento, devi effettuare la sottoscrizione dell'endpoint all'argomento Amazon SNS. L'endpoint deve essere specificato tramite il relativo URL. Per effettuare la sottoscrizione a un argomento, puoi utilizzare la console Amazon SNS, il comando [sns-subscribe](#) o l'operazione API [Subscribe](#). Prima di iniziare, verifica di disporre dell'URL dell'endpoint di cui effettuare la sottoscrizione e che l'endpoint sia pronto a ricevere i messaggi di conferma e notifica, come descritto nella fase 1.

Per effettuare la sottoscrizione di un endpoint HTTP o HTTPS a un argomento utilizzando la console Amazon SNS

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, scegli Sottoscrizioni.
3. Scegli l'opzione Create Subscription (Crea sottoscrizione).
4. Nell'elenco a discesa Protocol (Protocollo) seleziona HTTP o HTTPS.
5. Nella casella Endpoint incolla l'URL dell'endpoint a cui l'argomento invierà messaggi e scegli Create subscription (Crea sottoscrizione).
6. Viene visualizzato il messaggio di conferma. Scegli Chiudi.

Viene visualizzato l'ID di abbonamento del nuovo abbonamento PendingConfirmation. Dopo la conferma della sottoscrizione, il campo Subscription ID (ID sottoscrizione) visualizzerà il relativo ID.

Passaggio 3: conferma l'abbonamento ad Amazon SNS

Per confermare l'abbonamento ad Amazon SNS, segui questi passaggi per assicurarti che l'endpoint possa ricevere correttamente i messaggi. Questo processo prevede la configurazione dell'endpoint per gestire i messaggi di conferma in arrivo, il recupero dell'URL di conferma e la conferma dell'abbonamento. Puoi confermare l'abbonamento automaticamente o manualmente, a seconda della configurazione.

1. Dopo la sottoscrizione a un argomento di Amazon SNS, Amazon SNS invia un messaggio di conferma al tuo endpoint. Questo messaggio contiene un messaggio `SubscribeURL` che devi utilizzare per confermare l'abbonamento.
2. L'endpoint deve essere configurato per ascoltare i messaggi in arrivo da Amazon SNS. Quando arriva il messaggio di conferma, estrailo **SubscribeURL** dal messaggio.

3. Una volta ottenuto il `SubscribeURL`, puoi confermare l'abbonamento in due modi:

- Conferma automatica: l'endpoint può confermare automaticamente l'abbonamento inviando una richiesta HTTP GET a `SubscribeURL`.

Questo metodo non richiede un intervento manuale.

- Conferma manuale: se la conferma automatica non è impostata, copia il messaggio **`SubscribeURL`** di conferma e incollalo nella barra degli indirizzi del browser.

Ciò confermerà l'iscrizione manualmente.

4. Puoi anche verificare lo stato dell'abbonamento utilizzando la console Amazon SNS:

- a. Accedi alla [console Amazon SNS](#).
- b. Nel riquadro di navigazione scegli Subscriptions (Sottoscrizioni).
- c. Trova il tuo abbonamento nell'elenco.
 - Se confermato, `SubscriptionArn` verrà visualizzato.
 - Se non è ancora confermato, verrà visualizzato come `PendingConfirmation`.

Passaggio 4: Facoltativo: imposta la politica di consegna per l'abbonamento Amazon SNS

Per default, se la consegna iniziale del messaggio non riesce, Amazon SNS ritenta fino a tre volte, con un intervallo di 20 secondi tra tentativi non riusciti. Come descritto nella [fase 1](#), l'endpoint deve disporre di codice in grado di gestire i tentativi ripetuti di consegna dei messaggi. Impostando la policy di consegna per un argomento o una sottoscrizione puoi determinare la frequenza e l'intervallo tra i tentativi di consegna non riusciti di Amazon SNS. Puoi anche specificare il tipo di contenuto HTTP/S in `DeliveryPolicy`. Per ulteriori informazioni, consulta [Creazione di una politica di consegna HTTP/S](#).

Passaggio 5: Facoltativo: concedere 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, è necessario utilizzare AWS Identity and Access Management (IAM) per concedere l'autorizzazione alla pubblicazione

dell'argomento. Per informazioni sull'assegnazione di autorizzazioni per le operazioni Amazon SNS agli utenti IAM, consulta [Utilizzo di policy basate su identità con Amazon SNS](#).

Vi sono due modi di controllare l'accesso a un argomento:

- Aggiungere una policy a un utente o gruppo IAM. Il modo più semplice di concedere agli utenti le autorizzazioni per gli argomenti è creare un gruppo e aggiungere la policy appropriata al gruppo e quindi aggiungere gli utenti a quel gruppo. È molto più semplice aggiungere e rimuovere utenti da un gruppo anziché tenere traccia delle policy impostate su singoli utenti.
- Aggiungere una policy all'argomento. Se desideri concedere le autorizzazioni per un argomento a un altro account AWS, l'unico modo consiste nell'aggiungere una policy che abbia come principale il Account AWS a cui concedere le autorizzazioni.

Il primo metodo deve essere utilizzato nella maggior parte dei casi (applicare policy a gruppi e gestire le autorizzazioni per gli utenti aggiungendo o rimuovendo gli utenti appropriati ai gruppi). Se hai la necessità di concedere autorizzazioni a un utente in un altro account, utilizza il secondo metodo.

Se aggiungessi la seguente policy a un utente o gruppo IAM, daresti a quell'utente o ai membri di quel gruppo il permesso di eseguire l'`sns:Publish` sull'argomento `MyTopic`.

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

La policy di esempio seguente mostra come concedere a un altro account le autorizzazioni per un argomento.

Note

Quando concedi a un'altra persona Account AWS l'accesso a una risorsa del tuo account, concedi anche agli utenti IAM che dispongono di autorizzazioni di accesso a livello di amministratore (accesso con wildcard) a quella risorsa. L'accesso alla risorsa viene automaticamente negato a tutti gli altri utenti IAM nell'altro account. Se desideri consentire a utenti IAM specifici di Account AWS accedere alla tua risorsa, l'account o un utente IAM

con accesso a livello di amministratore deve delegare le autorizzazioni per la risorsa a quegli utenti IAM. Per ulteriori informazioni sulla delega multiaccount, consulta la sezione relativa all'[abilitazione dell'accesso multiaccount](#) nella Guida all'uso di IAM.

Se 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 messaggi Amazon SNS all'endpoint HTTP/HTTPS

Puoi inviare un messaggio alle sottoscrizioni di un argomento pubblicando nell'argomento. Per pubblicare in un argomento, puoi utilizzare la console Amazon SNS, il comando della CLI [sns-publish](#) o l'operazione API [Publish](#).

Se hai seguito le istruzioni nella [fase 1](#), il codice distribuito all'endpoint è in grado di elaborare la notifica.

Pubblicazione di un argomento utilizzando la console Amazon SNS

1. Utilizzando le credenziali dell'utente Account AWS o IAM con autorizzazione alla pubblicazione sull'argomento, accedi Console di gestione AWS e apri la console Amazon SNS all'indirizzo. <https://console.aws.amazon.com/sns/>
2. Nel riquadro di navigazione, selezionare Topics (Argomenti) e scegli un argomento.
3. Scegliere il pulsante Publish message (Pubblica messaggio).
4. Nella casella Subject (Oggetto), immetti un oggetto (ad esempio **Testing publish to my endpoint**).

5. Nella casella Message (Messaggio), immetti del testo (ad esempio, **Hello world!**) e scegli Publish message (Pubblica messaggio).

Viene visualizzato il messaggio "Your message has been successfully published" (Il messaggio è stato pubblicato).

Verifica delle firme dei messaggi Amazon SNS

Amazon SNS utilizza le firme dei messaggi per confermare l'autenticità dei messaggi inviati al tuo endpoint HTTP. Per garantire l'integrità dei messaggi e prevenire lo spoofing, devi verificare la firma prima di elaborare qualsiasi messaggio Amazon SNS.

Quando è necessario verificare le firme di Amazon SNS?

È necessario verificare le firme dei messaggi Amazon SNS nei seguenti scenari:

- Quando Amazon SNS invia un messaggio di notifica al tuo endpoint HTTP (S).
- Quando Amazon SNS invia un messaggio di conferma al tuo endpoint dopo una chiamata [Subscribe](#) o [Unsubscribe](#) API.

Amazon SNS supporta due versioni di firma:

- SignatureVersion1 — Utilizza un SHA1 hash del messaggio.
- SignatureVersion2 — Utilizza un SHA256 hash del messaggio. Ciò offre una maggiore sicurezza ed è l'opzione consigliata.

Per verificare correttamente le firme dei messaggi SNS, segui queste best practice:

- Recupera sempre il certificato di firma utilizzando HTTPS per prevenire attacchi di intercettazione non autorizzati.
- Verifica che il certificato sia emesso da Amazon SNS.
- Verifica che la catena di fiducia del certificato sia valida.
- Il certificato deve provenire da un URL firmato da SNS.
- Non fidatevi dei certificati forniti nel messaggio senza convalida.
- Rifiuta qualsiasi messaggio con un messaggio imprevisto TopicArn per evitare lo spoofing.

- I servizi AWS SDKs per Amazon SNS forniscono una logica di convalida integrata, che riduce il rischio di implementazione errata.

Configurazione della versione della firma dei messaggi sugli argomenti di Amazon SNS

La configurazione della versione della firma dei messaggi sugli argomenti di Amazon SNS consente di migliorare la sicurezza e la compatibilità del processo di verifica dei messaggi.

Seleziona tra `SignatureVersion 1 (SHA1)` e `SignatureVersion 2 (SHA256)` per controllare l'algoritmo di hashing utilizzato per firmare i messaggi. L'impostazione predefinita degli argomenti di Amazon SNS è 1. **SignatureVersion** Puoi configurare questa impostazione utilizzando l'azione [SetTopicAttributes](#) API.

Utilizzate l'esempio seguente per impostare l'attributo topic `SignatureVersion` utilizzando AWS CLI:

```
aws sns set-topic-attributes \
  --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
  --attribute-name SignatureVersion \
  --attribute-value 2
```

Verifica della firma di un messaggio Amazon SNS quando si utilizzano richieste basate su query HTTP

La verifica della firma di un messaggio Amazon SNS quando si utilizzano richieste basate su query HTTP garantisce l'autenticità e l'integrità del messaggio. Questo processo conferma che il messaggio proviene da Amazon SNS e non è stato manomesso durante il transito. Analizzando il messaggio, creando la stringa corretta da firmare e convalidando la firma con una chiave pubblica affidabile, proteggi il sistema da falsificazioni e alterazioni non autorizzate dei messaggi.

1. Estrai coppie chiave-valore dal documento JSON nel corpo della richiesta HTTP POST inviato da Amazon SNS. Questi campi sono necessari per costruire la stringa da firmare.
 - Message
 - Subject(se presente)
 - MessageId
 - Timestamp

- TopicArn
- Type

Esempio:

```
MESSAGE_FILE="message.json"
FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
```

 Note

Se un campo contiene caratteri in escape (ad esempio, \n), convertili nella forma originale per garantire una corrispondenza esatta.

2. Individua il SigningCertURL campo nel messaggio Amazon SNS. Questo certificato contiene la chiave pubblica necessaria per verificare la firma del messaggio. Esempio:

```
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
```

3. Assicurati SigningCertURL che provenga da un AWS dominio affidabile (ad esempio, <https://sns.us-east-1.amazonaws.com>). Rifiuta qualsiasi AWS dominio URLs esterno per motivi di sicurezza.
4. Scarica il certificato X.509 dall'URL fornito. Esempio:

```
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
```

5. Estrai la chiave pubblica dal certificato X.509 scaricato. La chiave pubblica consente di decrittografare la firma del messaggio e verificarne l'integrità. Esempio:

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. Tipi di messaggi diversi richiedono coppie chiave-valore diverse nella stringa da firmare. Identifica il tipo di messaggio (Type campo nel messaggio Amazon SNS) per determinare quali coppie chiave-valore includere:
 - Messaggio di notifica: include MessageId, Subject (se presente), Timestamp, TopicArn e. Type

- SubscriptionConfirmation UnsubscribeConfirmation messaggio: includeMessage, MessageIdSubscribeURL, Timestamp, Token, TopicArn, eType.

7. Amazon SNS richiede che la stringa da firmare segua un ordine rigoroso e fisso di campi per la verifica. Devono essere inclusi solo i campi esplicitamente richiesti, non è possibile aggiungere campi aggiuntivi. I campi opzionaliSubject, ad esempio, devono essere inclusi solo se presenti nel messaggio e devono apparire nella posizione esatta definita dall'ordine dei campi richiesto. Esempio:

```
KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
```

Important

Non aggiungete un carattere di nuova riga alla fine della stringa.

8. Disporre le coppie chiave-valore in ordine di byte (alfabetico per nome della chiave).
9. Costruisci la stringa da firmare utilizzando il seguente esempio di formato:

```
STRING_TO_SIGN=""
for FIELD in "${FIELDS[@]}"; do
    VALUE=$(jq -r --arg field "$FIELD" '.[${field}]' "$MESSAGE_FILE")
    STRING_TO_SIGN+="$FIELD\n$VALUE"
    # Append a newline after each field except the last one
    if [[ "$FIELD" != "Type" ]]; then
        STRING_TO_SIGN+="\n"
    fi
done
```

Esempio di messaggio di notifica:

```
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-1G1WEFC0XTC0P
```

```
Type
Notification
```

SubscriptionConfirmation esempio:

```
Message
Please confirm your subscription
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/...
Timestamp
2024-01-01T00:00:00.000Z
Token
abc123...
TopicArn
arn:aws:sns:us-east-2:123456789012:MyTopic
Type
SubscriptionConfirmation
```

10. Il Signature campo nel messaggio è codificato in Base64. È necessario decodificarlo per confrontare la sua forma binaria grezza con l'hash derivato. Esempio:

```
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
echo "$SIGNATURE" | base64 -d > signature.bin
```

11. Usa il SignatureVersion campo per selezionare l'algoritmo di hash:

- Per SignatureVersion 1, usa SHA1(ad esempio, -sha1).
- Per SignatureVersion 2, usa SHA256(ad esempio, -sha256).

12. Per confermare l'autenticità del messaggio Amazon SNS, genera un hash della stringa costruita e verifica la firma utilizzando la chiave pubblica.

```
openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
"$STRING_TO_SIGN"
```

Se la firma è valida, l'output è. Verified OK Altrimenti, l'output èVerification Failure.

Script di esempio con gestione degli errori

Il seguente script di esempio automatizza il processo di verifica:

```
#!/bin/bash

# Path to the local message file
MESSAGE_FILE="message.json"

# Extract the SigningCertURL and Signature from the message
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")

# Fetch the X.509 certificate
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem

# Extract the public key from the certificate
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem

# Define the fields to include in the string to sign
FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")

# Initialize the string to sign
STRING_TO_SIGN=""

# Iterate over the fields to construct the string to sign
for FIELD in "${FIELDS[@]}; do
    VALUE=$(jq -r --arg field "$FIELD" '[$field]' "$MESSAGE_FILE")
    STRING_TO_SIGN+="$FIELD\n$VALUE"
    # Append a newline after each field except the last one
    if [[ "$FIELD" != "Type" ]]; then
        STRING_TO_SIGN+="\n"
    fi
done

# Verify the signature
echo -e "$STRING_TO_SIGN" | openssl dgst -sha256 -verify public_key.pem -signature
<(echo "$SIGNATURE" | base64 -d)
```

Analisi dei formati di messaggi Amazon SNS

Quando Amazon SNS invia messaggi agli HTTP/HTTPS endpoint, questi contengono sia intestazioni HTTP che un corpo del messaggio JSON. Questi messaggi seguono un formato strutturato che

include metadati come il tipo di messaggio, l'ARN dell'argomento, i timestamp e le firme digitali. Analizzando correttamente i messaggi di Amazon SNS, puoi determinare se un messaggio è una conferma di iscrizione, una notifica o una conferma di annullamento dell'iscrizione, estrarre i dati pertinenti e verificare l'autenticità utilizzando la convalida della firma.

Intestazioni HTTP/HTTPS

Quando Amazon SNS invia un messaggio di conferma dell'abbonamento, notifica o annullamento dell'iscrizione agli HTTP/HTTPS endpoint, invia un messaggio POST con una serie di valori di intestazione specifici di Amazon SNS. Puoi utilizzare i valori d'intestazione per attività quali l'identificazione del tipo di messaggio senza dover analizzare il corpo del messaggio JSON per leggere il valore `Type`. Per impostazione predefinita, Amazon SNS invia tutte le notifiche agli endpoint HTTP/S con `Content-Type` impostato su `text/plain; charset=UTF-8`. Per scegliere un valore `Content-Type` diverso da `text/plain` (predefinito), consulta in `headerContentType` [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 unico universale (UUID), univoco per ogni messaggio pubblicato. Per una notifica che Amazon SNS reinvia durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

x-amz-sns-topic-arn

L'Amazon Resource Name (ARN) per l'argomento in cui questo messaggio è stato pubblicato.

x-amz-sns-subscription-arn

L'ARN per la sottoscrizione a questo endpoint.

L'intestazione HTTP POST seguente è un esempio di intestazione per un messaggio `Notification` inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
```

```
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

Conferma sottoscrizione HTTP/HTTPS in formato JSON

Dopo aver sottoscritto un HTTP/HTTPS endpoint, Amazon SNS invia un messaggio di conferma dell'iscrizione all' 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 invierà notifiche a questo endpoint fino alla conferma della sottoscrizione

Il messaggio di conferma della sottoscrizione è un messaggio POST con un corpo che contiene un documento JSON con le seguenti coppie nome/valore.

Type

Il tipo di messaggio. Per confermare la sottoscrizione, il tipo è `SubscriptionConfirmation`.

MessageId

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per un messaggio che Amazon SNS invia di nuovo durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

Token

Un valore che puoi utilizzare con l'operazione [ConfirmSubscription](#) per confermare la sottoscrizione. In alternativa, puoi selezionare `SubscribeURL`.

TopicArn

L'Amazon Resource Name (ARN) per l'argomento a cui questo endpoint è sottoscritto.

Message

Una stringa che descrive il messaggio. Per una conferma di sottoscrizione, la stringa deve avere il seguente aspetto:

You have chosen to subscribe to the topic arn:aws:sns:us-east-2:123456789012:MyTopic.\n\nTo confirm the subscription, visit the SubscribeURL included in this message.

SubscribeURL

L'URL da selezionare per confermare la sottoscrizione. In alternativa, puoi utilizzare il Token con l'operazione [ConfirmSubscription](#) per confermare la sottoscrizione.

Timestamp

L'ora (GMT) in cui è stata inviata la conferma della sottoscrizione.

SignatureVersion

La versione della firma Amazon SNS utilizzata.

- Se SignatureVersion è 1, Signature è una firma SHA1withRSA con codifica Base64 dei valori Message, MessageId, Type, Timestamp e TopicArn.
- Se SignatureVersion è 2, Signature è una firma SHA256withRSA con codifica Base64 dei valori Message, MessageId, Type, Timestamp e TopicArn.

Signature

Firma SHA1withRSA o SHA256withRSA con codifica Base64 dei valori Message, MessageIdType, Timestamp e TopicArn.

SigningCertURL

L'URL per il certificato utilizzato per firmare il messaggio.

Il seguente messaggio HTTP POST è un esempio di messaggio SubscriptionConfirmation inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
```

User-Agent: Amazon Simple Notification Service Agent

```
{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-west-2:123456789012:MyTopic.\n\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+DcEwjAPg809mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkk0xYDVrY0Ad8L",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

Notifica HTTP/HTTPS in formato JSON

Quando Amazon SNS invia una notifica a un endpoint HTTP o HTTPS sottoscritto, il corpo del messaggio POST inviato all'endpoint contiene un documento JSON con le seguenti coppie nome/valore.

Type

Il tipo di messaggio. Per una notifica, il tipo è `Notification`.

MessageId

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per una notifica che Amazon SNS reinvia durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

TopicArn

L'Amazon Resource Name (ARN) per l'argomento in cui questo messaggio è stato pubblicato.

Subject

Il parametro `Subject` specificato quando la notifica è stata pubblicata nell'argomento.

Note

Si tratta di un parametro facoltativo. Se non è stato specificato alcun Subject, questa coppia nome/valore non appare nel documento JSON.

Message

Il valore Message specificato nel momento in cui la notifica è stata pubblicata nell'argomento.

Timestamp

L'ora (GMT) di pubblicazione della notifica.

SignatureVersion

La versione della firma Amazon SNS utilizzata.

- Se SignatureVersion è 1, Signature è una firma SHA1withRSA con codifica Base64 dei valori Message, MessageId, Subject (se presente), Type, Timestamp e TopicArn.
- Se SignatureVersion è 2, Signature è una firma SHA256withRSA con codifica Base64 dei valori Message, MessageId, Subject (se presente), Type, Timestamp e TopicArn.

Signature

Firma SHA1withRSA o SHA256withRSA con codifica Base64 dei valori Message, MessageIdSubject (se presente), Type, Timestamp e TopicArn.

SigningCertURL

L'URL per il certificato utilizzato per firmare il messaggio.

UnsubscribeURL

Un URL che puoi utilizzare per annullare la sottoscrizione dell'endpoint da questo argomento. Se visiti questo URL, Amazon SNS annulla la sottoscrizione all'endpoint e interrompe l'invio di notifiche a esso.

Il seguente messaggio HTTP POST è un esempio di messaggio Notification inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
```

```
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

Conferma annullamento sottoscrizione HTTP/HTTPS in formato JSON

Dopo l'annullamento dell'iscrizione a un HTTP/HTTPS endpoint da un argomento, Amazon SNS invia un messaggio di conferma dell'annullamento dell'iscrizione all'endpoint.

Il messaggio di conferma dell'annullamento della sottoscrizione è un messaggio POST con un corpo che contiene un documento JSON con le seguenti coppie nome/valore.

Type

Il tipo di messaggio. Per confermare l'annullamento della sottoscrizione, il tipo è `UnsubscribeConfirmation`.

MessageId

Un identificatore unico universale (UUID), univoco per ogni messaggio pubblicato. Per un messaggio che Amazon SNS invia di nuovo durante un nuovo tentativo, viene utilizzato l'ID messaggio originale.

Token

Un valore che puoi utilizzare con l'operazione [ConfirmSubscription](#) per confermare nuovamente la sottoscrizione. In alternativa, puoi selezionare SubscribeURL.

TopicArn

L'Amazon Resource Name (ARN) per l'argomento da cui è stato effettuato l'annullamento della sottoscrizione di questo endpoint.

Message

Una stringa che descrive il messaggio. La stringa per la conferma di annullamento della sottoscrizione ha il seguente aspetto:

```
You have chosen to deactivate subscription arn:aws:sns:us-east-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this operation and restore the subscription, visit the SubscribeURL included in this message.
```

SubscribeURL

L'URL da selezionare per confermare nuovamente la sottoscrizione. In alternativa, puoi utilizzare il Token con l'operazione [ConfirmSubscription](#) per confermare nuovamente la sottoscrizione.

Timestamp

L'ora (GMT) in cui è stata inviata la conferma di annullamento della sottoscrizione.

SignatureVersion

La versione della firma Amazon SNS utilizzata.

- Se SignatureVersion è 1, Signature è una firma SHA1withRSA con codifica Base64 dei valori Message, MessageId, Type, Timestamp e TopicArn.
- Se SignatureVersion è 2, Signature è una firma SHA256withRSA con codifica Base64 dei valori Message, MessageId, Type, Timestamp e TopicArn.

Signature

Firma SHA1withRSA o SHA256withRSA con codifica Base64 dei valori Message, MessageIdType, Timestamp e TopicArn.

SigningCertURL

L'URL per il certificato utilizzato per firmare il messaggio.

Il seguente messaggio HTTP POST è un esempio di messaggio `UnsubscribeConfirmation` inviato a un endpoint HTTP.

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\n\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"
}
```

SetSubscriptionAttributes politica di consegna (formato JSON)

Se invii una richiesta all'operazione `SetSubscriptionAttributes` e imposti il parametro `AttributeName` su un valore di `DeliveryPolicy`, il valore del parametro `AttributeValue` deve essere un oggetto JSON valido. Per esempio, il caso seguente imposta la policy di consegna su 5 tentativi totali.

```
http://sns.us-east-2.amazonaws.com/
```

```
?Action=SetSubscriptionAttributes
&SubscriptionArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
%3A80289ba6-0fd4-4079-afb4-ce8c8260f0ca
&AttributeName=DeliveryPolicy
&AttributeValue={"healthyRetryPolicy":{"numRetries":5}}
...
```

Utilizza il seguente formato JSON per il valore del parametro `AttributeValue`.

```
{
  "healthyRetryPolicy" : {
    "minDelayTarget" : int,
    "maxDelayTarget" : int,
    "numRetries" : int,
    "numMaxDelayRetries" : int,
    "backoffFunction" : "linear|arithmetic|geometric|exponential"
  },
  "throttlePolicy" : {
    "maxReceivesPerSecond" : int
  },
  "requestPolicy" : {
    "headerContentType" : "text/plain | application/json | application/xml"
  }
}
```

Per ulteriori informazioni sull'operazione `SetSubscriptionAttributes`, consulta [Amazon Simple Notification Service API Reference](#). [SetSubscriptionAttributes](#) Per ulteriori informazioni sulle intestazioni dei tipi di contenuto HTTP supportate, consultare [Creazione di una politica di consegna HTTP/S](#).

SetTopicAttributes politica di consegna (formato JSON)

Se invii una richiesta all'operazione `SetTopicAttributes` e imposti il parametro `AttributeName` su un valore di `DeliveryPolicy`, il valore del parametro `AttributeValue` deve essere un oggetto JSON valido. Per esempio, il caso seguente imposta la policy di consegna su 5 tentativi totali.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetTopicAttributes
&TopicArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
&AttributeName=DeliveryPolicy
&AttributeValue={"http":{"defaultHealthyRetryPolicy":{"numRetries":5}}}
```

...

Utilizza il seguente formato JSON per il valore del parametro `AttributeValue`.

```
{
  "http" : {
    "defaultHealthyRetryPolicy" : {
      "minDelayTarget": int,
      "maxDelayTarget": int,
      "numRetries": int,
      "numMaxDelayRetries": int,
      "backoffFunction": "linear|arithmetic|geometric|exponential"
    },
    "disableSubscriptionOverrides" : Boolean,
    "defaultThrottlePolicy" : {
      "maxReceivesPerSecond" : int
    },
    "defaultRequestPolicy" : {
      "headerContentType" : "text/plain | application/json | application/xml"
    }
  }
}
```

Per ulteriori informazioni sull'`SetTopicAttribute`azione, consulta Amazon Simple Notification Service API Reference. [SetTopicAttributes](#) Per ulteriori informazioni sulle intestazioni dei tipi di contenuto HTTP supportate, consultare [Creazione di una politica di consegna HTTP/S](#).

Fanout degli eventi Amazon SNS su Event Fork AWS Pipelines

Per l'archiviazione e l'analisi degli eventi, Amazon SNS ora consiglia di utilizzare la sua integrazione nativa con Amazon Data Firehose. Puoi abbonare i flussi di distribuzione Firehose agli argomenti SNS, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift, Amazon Service (Service) e altro ancora. OpenSearch OpenSearch L'uso di Amazon SNS con i flussi di distribuzione Firehose è una soluzione completamente gestita e priva di codice che non richiede l'utilizzo di funzioni. AWS Lambda Per ulteriori informazioni, consulta [Flussi di distribuzione da Fanout a Firehose](#).

È possibile utilizzare Amazon SNS per creare applicazioni basate su eventi che utilizzano i servizi del sottoscrittore per eseguire automaticamente il lavoro in risposta a eventi attivati dai servizi del publisher. Questo modello di architettura può rendere i servizi più riutilizzabili, interoperabili e scalabili. Tuttavia, può essere impegnativo eseguire il forking dell'elaborazione di eventi in pipeline rivolte ai requisiti di gestione di eventi comuni, come l'archiviazione, il backup, la ricerca, l'analisi dei dati e la riproduzione.

Per accelerare lo sviluppo delle tue applicazioni basate sugli eventi, puoi sottoscrivere pipeline di gestione degli eventi, basate su AWS Event Fork Pipelines, agli argomenti di Amazon 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 Show apps that create IAM role o policy di risorse personalizzate) nel tuo account. AWS

Per un AWS caso d'uso di Event Fork Pipelines, vedi. [Implementazione e test dell'applicazione di esempio Amazon SNS Event Fork Pipeline](#)

Argomenti

- [Come funziona AWS Event Fork Pipelines](#)
- [Distribuzione di Event Fork AWS Pipelines](#)
- [Implementazione e test dell'applicazione di esempio Amazon SNS Event Fork Pipeline](#)
- [Sottoscrizione di AWS Event Fork Pipelines a un argomento di Amazon SNS](#)

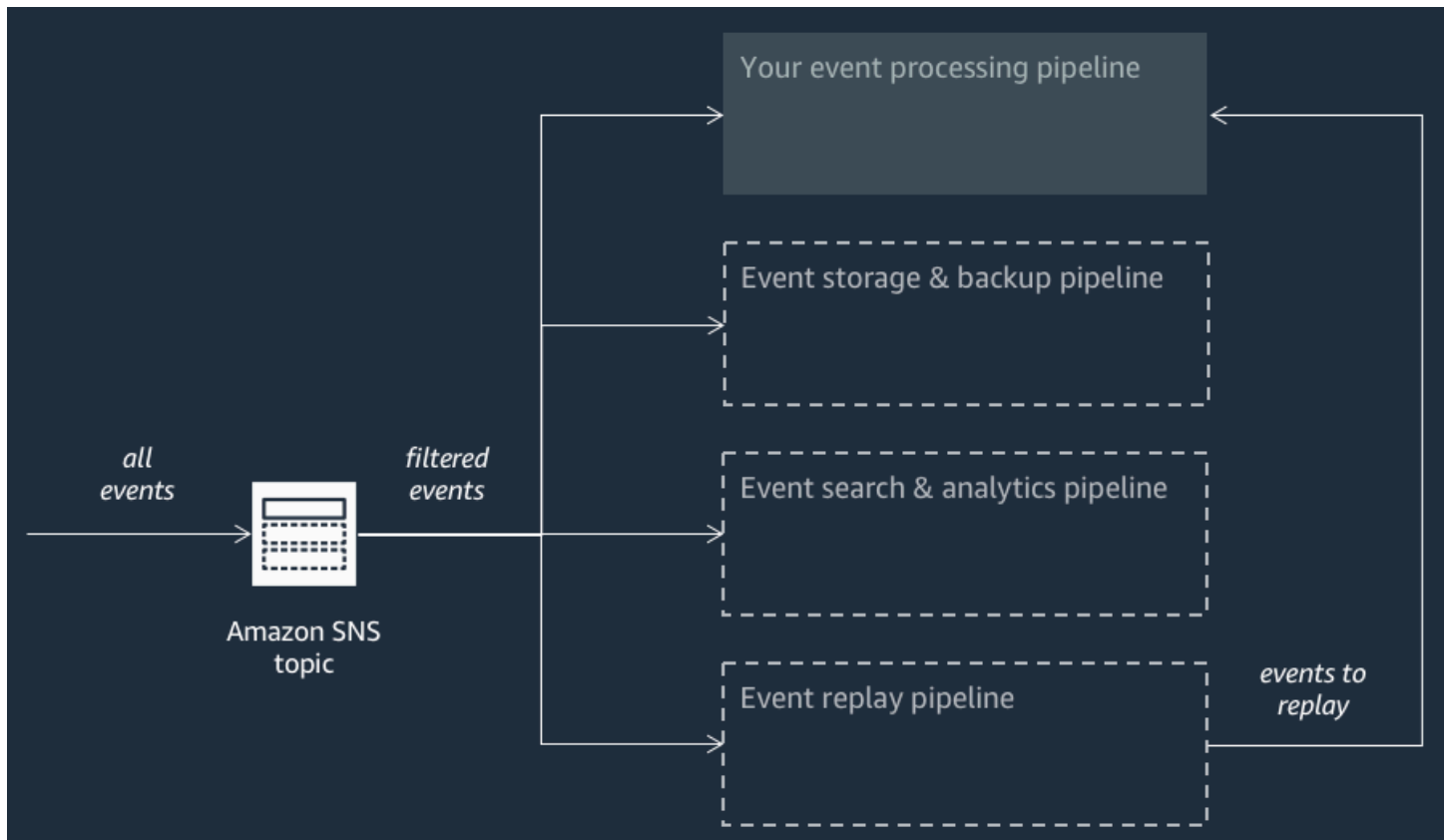
Come funziona AWS Event Fork Pipelines

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 implementare direttamente da (AWS SAR) alle vostre piattaforme per AWS Serverless Application Repository arricchire le vostre piattaforme basate sugli Account AWS eventi). 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 sul SAR in modo indipendente, come richiesto dall'architettura. AWS



Ogni pipeline è sottoscritta allo stesso argomento Amazon SNS, facendo così in modo di elaborare eventi non appena vengono pubblicati nell'argomento. Ogni pipeline è indipendente ed è in grado di impostare la propria [Policy di filtro per le sottoscrizioni](#). In questo modo una pipeline può elaborare solo un sottoinsieme di eventi a cui è interessata (invece che tutti gli eventi pubblicati nell'argomento).

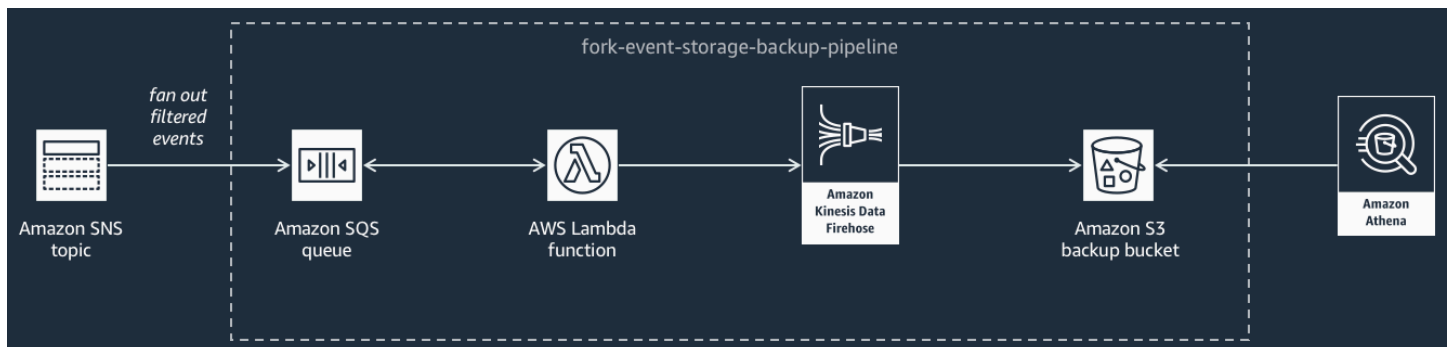
Note

Poiché colloca le tre AWS Event Fork Pipeline insieme alle normali pipeline di elaborazione degli eventi (probabilmente già iscritto al tuo argomento Amazon SNS), non è necessario modificare alcuna parte del tuo attuale editore 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 sottoscrivere questa pipeline al tuo argomento Amazon SNS per eseguire il backup automatico degli eventi che scorrono nel sistema.

Questa pipeline è composta da una coda Amazon SQS che memorizza nel buffer gli eventi forniti dall'argomento Amazon SNS, AWS Lambda una funzione che esegue automaticamente il polling di questi eventi nella coda e li inserisce in un flusso e un bucket Amazon S3 che esegue il backup durevole degli eventi caricati dallo stream.

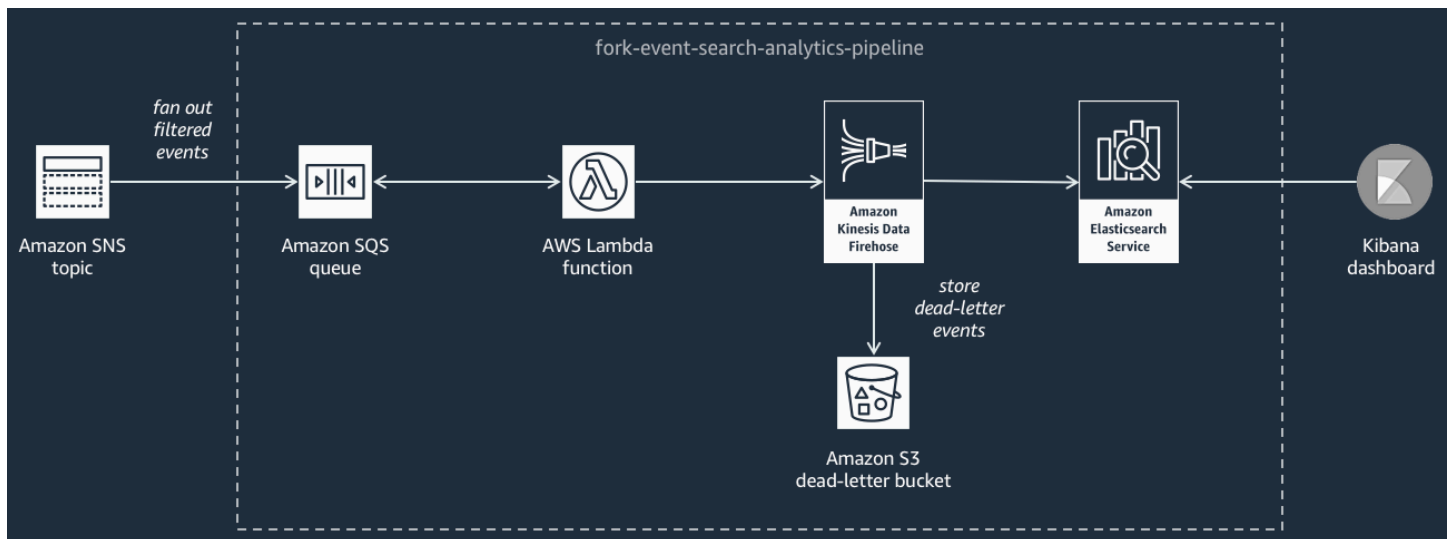


Per affinare il comportamento del flusso Firehose, puoi configurarlo in modo che trasformi, compri e esegua il buffering degli eventi prima di caricarli nel bucket. Man mano che gli eventi vengono caricati, puoi utilizzare Amazon Athena per eseguire query sul bucket utilizzando query SQL standard. Puoi anche configurare la pipeline in modo che riutilizzi un bucket Amazon S3 esistente o ne crei uno nuovo.

La pipeline di ricerca di eventi e di analisi dei dati

Il seguente diagramma mostra la [Pipeline di ricerca di eventi e di analisi dei dati](#). Puoi sottoscrivere questa pipeline al tuo argomento Amazon SNS per indicizzare gli eventi che scorrono nel sistema in un dominio di ricerca e per eseguirvi le analisi dei dati.

Questa pipeline è composta da una coda Amazon SQS che memorizza nel buffer gli eventi forniti dall'argomento Amazon SNS, AWS Lambda una funzione che analizza gli eventi dalla coda e li inserisce in un flusso, un dominio Amazon OpenSearch Service che indicizza gli eventi caricati dal flusso Firehose e un bucket Amazon S3 che memorizza gli eventi non indicizzabili in il dominio di ricerca.



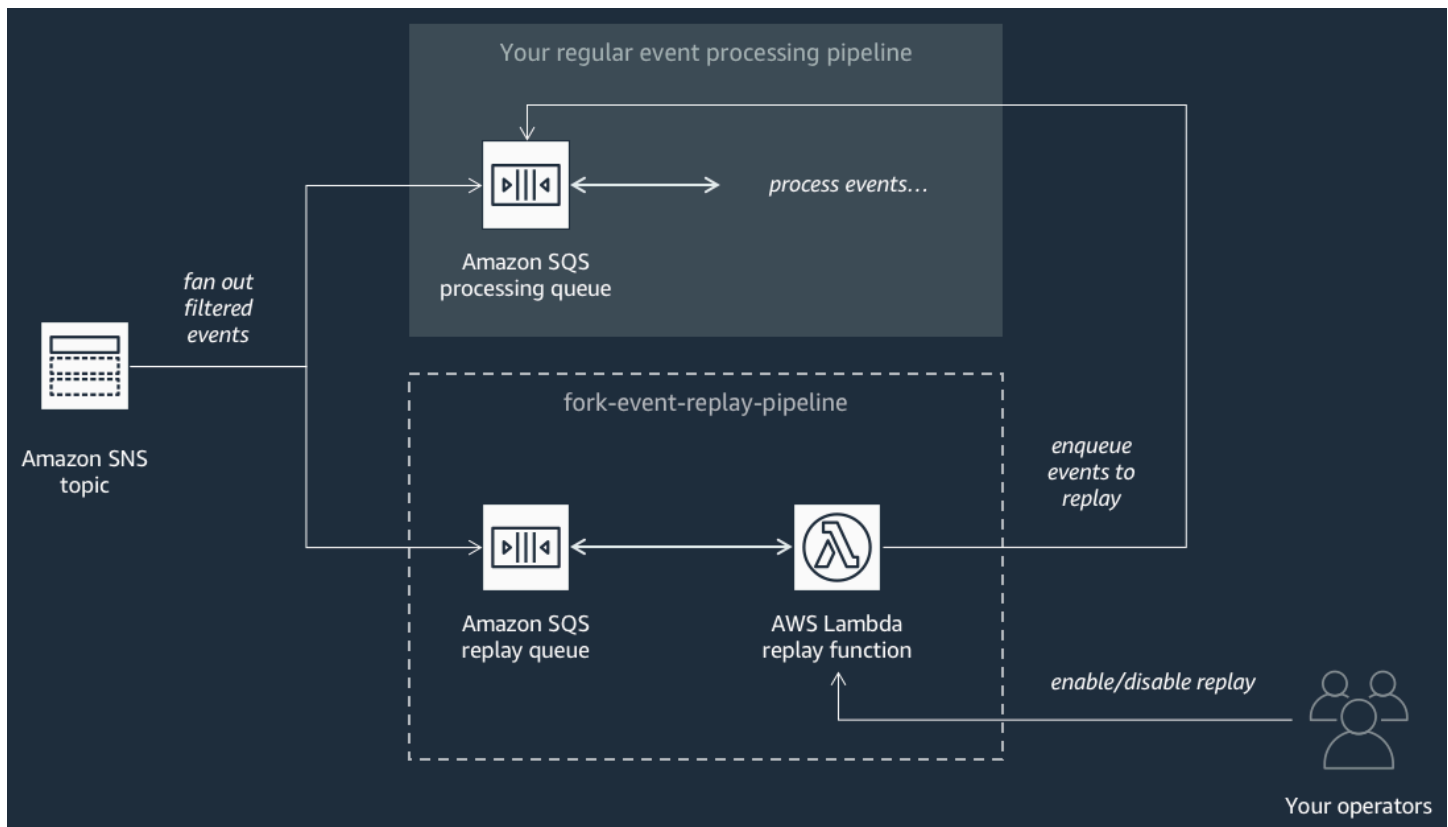
Per affinare il flusso Firehose in termini di buffering, trasformazione e compressione degli eventi, puoi configurare questa pipeline.

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 elaborati dal sistema negli ultimi 14 giorni (ad esempio quando la piattaforma deve essere ripristinata in seguito a un guasto), puoi sottoscrivere questa pipeline all'argomento Amazon SNS e rielaborare gli eventi.

Questa pipeline è composta da una coda Amazon SQS che memorizza nel buffer gli eventi forniti dall'argomento Amazon SNS e da AWS Lambda una funzione che analizza gli eventi dalla coda e li reindirizza nella normale pipeline di elaborazione degli eventi, anch'essa associata 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 riproduzione Amazon SQS come origine dell'evento per la funzione di riproduzione AWS Lambda.

Distribuzione di Event Fork AWS Pipelines

[La suite AWS Event Fork Pipelines \(scegli Show apps that create IAM roles or resource policy personalizzate\)](#) è disponibile come gruppo di applicazioni pubbliche in [AWS Serverless Application Repository](#), da cui è possibile distribuirle e testarle manualmente utilizzando la console. [AWS Lambda](#) Per informazioni sulla distribuzione delle pipeline tramite la console, consulta. [AWS Lambda Sottoscrizione di AWS Event Fork Pipelines a un argomento di Amazon SNS](#)

In uno scenario di produzione, consigliamo di incorporare AWS Event Fork Pipelines nel modello SAM generale dell'applicazione. AWS La funzionalità di applicazione annidata consente di eseguire questa operazione aggiungendo la risorsa [AWS::Serverless::Application](#) al modello

AWS SAM, facendo riferimento al AWS SAR ApplicationId e all'applicazione nidificata.

SemanticVersion

Ad esempio, è possibile utilizzare Event Storage and Backup Pipeline come applicazione annidata aggiungendo il seguente frammento YAML alla sezione del Resources modello SAM. AWS

```
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, nello snippet YAML riportato sopra, il TopicArn parametro fa riferimento alla [AWS::SNS::Topic](#) risorsaMySNSTopic, 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 dell'applicazione AWS SAR include il pulsante Copia come risorsa SAM, che copia il codice YAML necessario per annidare un'applicazione SAR negli appunti. AWS

Implementazione 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 AWS Event Fork Pipelines, agli argomenti di Amazon 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 Show apps that create IAM role o policy di risorse personalizzate) nel tuo account. AWS Per ulteriori informazioni, consulta [Come funziona AWS Event Fork Pipelines](#).

Questa pagina mostra come utilizzare l'applicazione di esempio Event Fork Pipelines Console di gestione AWS per distribuire e testare l'applicazione di esempio. AWS

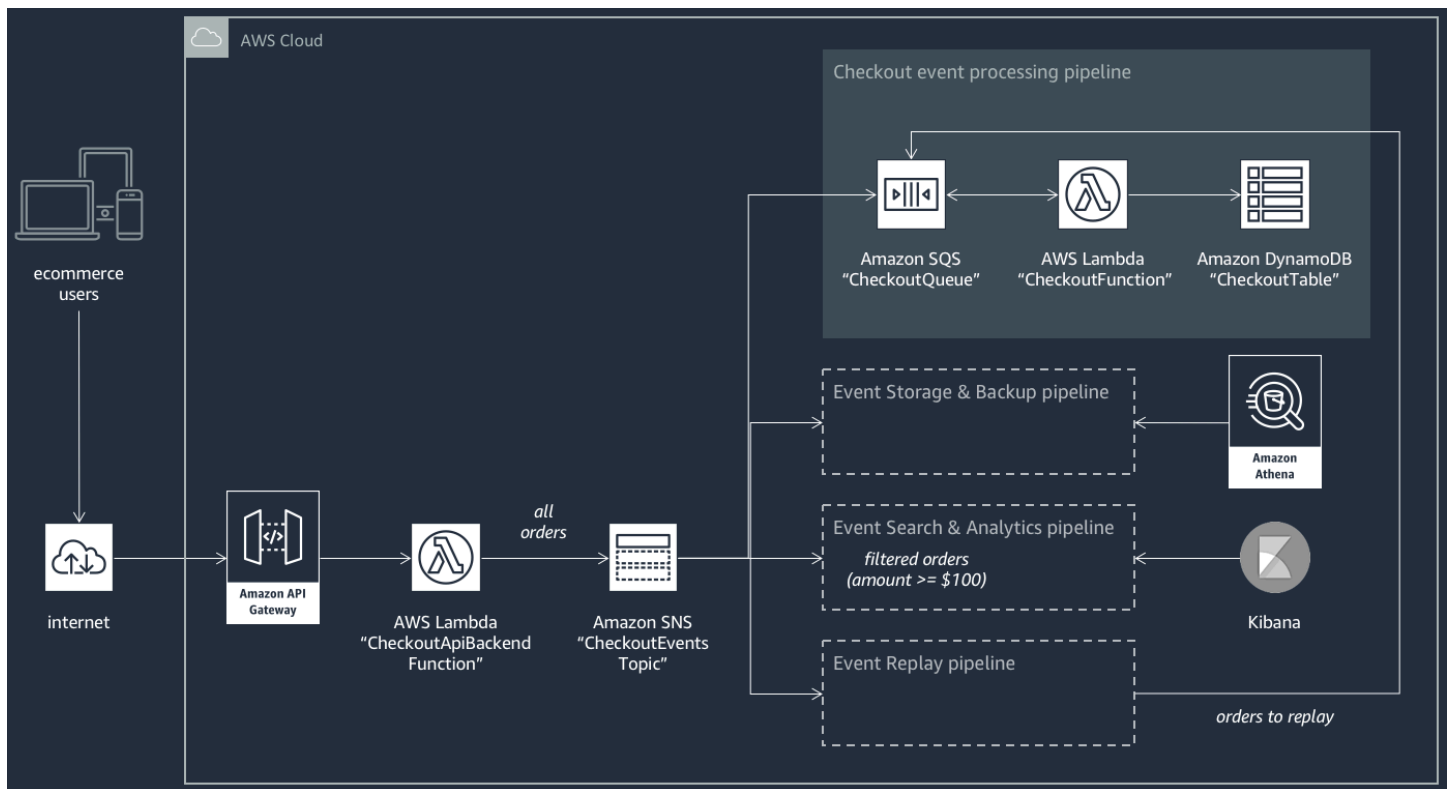
⚠ Important

Per evitare di incorrere in costi indesiderati dopo aver completato la distribuzione dell'applicazione di esempio AWS Event Fork Pipelines, eliminate il relativo stack.

CloudFormation Per ulteriori informazioni, vedi [Eliminazione di uno stack sulla console CloudFormation](#) nella Guida per l'utente di AWS CloudFormation .

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 distribuirlo 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' RESTful API ospitata da API Gateway e supportata dalla AWS Lambda funzione `CheckoutApiBackendFunction`. La funzione pubblica tutti gli ordini ricevuti in un argomento Amazon SNS denominato `CheckoutEventsTopic` che, a sua volta, li distribuisce a quattro diverse pipeline.

La prima è la normale pipeline di elaborazione del checkout progettata e implementata dal proprietario dell'applicazione di E-Commerce. Questa pipeline include la `CheckoutQueue` coda Amazon SQS che memorizza nel buffer tutti gli ordini ricevuti, AWS Lambda una funzione `CheckoutFunction` denominata 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 backup](#) è configurata per trasformare i dati in modo da rimuovere i dettagli delle carte di credito, eseguire il buffer dei dati per 60 secondi, comprimerli usando GZIP e crittografarli tramite la chiave predefinita gestita dal cliente per Amazon S3. 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 eventi](#) è configurata con la coda Amazon SQS che fa parte della normale pipeline di elaborazione degli ordini progettata e implementata dal proprietario dell'applicazione di E-Commerce.

Per ulteriori informazioni, consulta la sezione al [Nome della coda e URL](#) nella Guida per gli sviluppatori di Amazon Simple Queue Service.

Nella configurazione per la pipeline di ricerca di eventi e di analisi è impostata la seguente policy di filtro in formato JSON, che trova solo ordini in entrata il cui importo totale sia di 100 dollari o più. Per ulteriori informazioni, consulta [Filtraggio messaggi di Amazon SNS](#).

```
{
  "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

Utilizzando il 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. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
 - b. Cercare `fork-example-ecommerce-checkout-api` e scegliere l'applicazione.
4. Nella pagina `fork-example-ecommerce-checkout-api`, procedi come segue:
 - a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio `fork-example-ecommerce-my-app`).

 Note

- Per trovare con facilità le risorse in un secondo momento, mantenere il prefisso `fork-example-ecommerce`.
- Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se riutilizzi il nome di un'applicazione, la distribuzione aggiornerà solo lo CloudFormation stack precedentemente distribuito (anziché crearne uno nuovo).

b. (Facoltativo) Immettete una delle seguenti LogLevel impostazioni per l'esecuzione della funzione Lambda dell'applicazione:

- DEBUG
- ERROR
- INFO (predefinito)
- WARNING

5. Scegliere I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.) e quindi, in fondo alla pagina, selezionare Deploy (Distribuisci).

Nella *my-app* pagina Stato di distribuzione per fork-example-ecommerce -, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, CloudFormation inizia a creare lo stack e visualizza lo stato CREATE_IN_PROGRESS per ogni risorsa. Quando il processo è completo, visualizza lo stato CREATE_COMPLETE. CloudFormation

 Note

La distribuzione di tutte le risorse può richiedere 20-30 minuti.

Al termine della distribuzione, Lambda mostra lo stato Your application has been deployed (L'applicazione è stata distribuita).

Fase 2: Esecuzione dell'applicazione di esempio collegata a SNS

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. Copiare il link per Prod API endpoint (Endpoint API prod) e aggiungervi `/checkout`, ad esempio:

```
https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

4. Copiare il JSON seguente in un file denominato `test_event.json`.

```
{
  "id": 15311,
  "date": "2019-03-25T23:41:11-08:00",
  "status": "confirmed",
  "customer": {
    "id": 65144,
    "quantity": 2,
    "price": 25.00,
    "subtotal": 50.00
  }
}
```

5. Per inviare una richiesta HTTPS al proprio endpoint API, passare il payload degli eventi di esempio come input eseguendo un comando `curl`, ad esempio:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

L'API restituisce la seguente risposta vuota, a indicare che l'esecuzione è riuscita:

```
{ }
```

Fase 3: verifica delle prestazioni dell'applicazione e della pipeline Amazon SNS

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 payload JSON archiviato non contenga informazioni sulle carte di credito, decomprimere 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 viene configurata con una policy di filtro della sottoscrizione Amazon SNS che imposta una condizione di corrispondenza numerica.

Per verificare che l'evento sia indicizzato perché si riferisce a un ordine il cui valore è superiore a 100 USD, nella **abcdefghijkl** pagina `serverl-analyt`, scegli Indices, `checkout_events`.

Fase 4: Verifica dell'esecuzione della pipeline di riproduzione degli eventi

1. Accedere alla [console Amazon SQS](#).
2. Nell'elenco delle code, cercare `serverlessrepo-fork-example` e scegliere `ReplayQueue`.
3. Scegli `Invia e ricevi messaggi`.
4. Nella finestra di `123ABCD4E5F6` dialogo `Invia e ricevi messaggi` in `fork-example-ecommerce-my-app... ReplayP- ReplayQueue -`, 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

Passaggio 1: abilita il problema simulato e invia una seconda richiesta API

1. Accedi alla [console AWS Lambda](#).
2. Nel riquadro di navigazione, scegliere `Functions (Funzioni)`.
3. Cercare `serverlessrepo-fork-example` e selezionare `CheckoutFunction`.
4. Sul `fork-example-ecommerce-my-app - CheckoutFunction -ABCDEF...` pagina, nella sezione `Variabili d'ambiente`, imposta la variabile `BUG_ENABLED` su `true`, quindi scegli `Salva`.
5. Copiare il JSON seguente in un file denominato `test_event_2.json`.

```
{
  "id": 9917,
  "date": "2019-03-26T21:11:10-08:00",
  "status": "confirmed",
  "customer": {
    "id": 56999,
    "quantity": 1,
    "price": 75.00,
    "subtotal": 75.00
  }
}
```

6. Per inviare una richiesta HTTPS al proprio endpoint API, passare il payload degli eventi di esempio come input eseguendo un comando `curl`, ad esempio:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

L'API restituisce la seguente risposta vuota, a indicare che l'esecuzione è riuscita:

```
{ }
```

Fase 2: 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 archiviati, alcuni contrassegnati come CORRUPTED! (Danneggiati).

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 variabile `BUG_ENABLED` 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. Espandere la sezione Designer, scegliere il riquadro SQS e quindi, nella sezione SQS, scegliere Enabled (Abilitato).

 Note

L'abilitazione del trigger dell'origine evento Amazon SQS richiede circa un minuto.

4. Seleziona Salva.
5. Per visualizzare gli attributi ripristinati, tornare alla console Amazon DynamoDB.
6. Per disabilitare la riproduzione, torna alla AWS Lambda console e disabilita il trigger sorgente di eventi Amazon SQS per `ReplayFunction`

Sottoscrizione di AWS Event Fork Pipelines a un argomento di Amazon SNS

Per accelerare lo sviluppo delle tue applicazioni basate sugli eventi, puoi sottoscrivere pipeline di gestione degli eventi, basate su AWS Event Fork Pipelines, agli argomenti di Amazon 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 Show apps that create IAM role o policy di risorse personalizzate) nel tuo account. AWS Per ulteriori informazioni, consulta [Come funziona AWS Event Fork Pipelines](#).

Questa sezione mostra come utilizzare il Console di gestione AWS per distribuire una pipeline e quindi sottoscrivere AWS Event Fork Pipelines a un argomento di Amazon SNS. Prima di iniziare, [crea un argomento Amazon SNS](#).

Per eliminare le risorse che compongono una pipeline, trova la pipeline nella pagina Applicazioni della AWS Lambda console, espandi la sezione dei modelli SAM, scegli stack, quindi scegli Altre azioni, Elimina CloudFormationstack.

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 argomenti SNS, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift, Amazon Service (Service) e altro ancora. OpenSearch OpenSearch L'uso di Amazon SNS con i flussi di

distribuzione Firehose è una soluzione completamente gestita e priva di codice che non richiede l'utilizzo di funzioni. AWS Lambda Per ulteriori informazioni, consulta [Flussi di distribuzione da Fanout a Firehose](#).

Questo tutorial mostra come distribuire la [pipeline di storage di eventi e di backup](#) e iscriverla a un argomento Amazon SNS. Questo processo trasforma automaticamente il AWS SAM modello associato alla pipeline in uno 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:

- Coda Amazon SQS
- funzione Lambda
- Flussi di distribuzione Firehose
- S3 backup bucket Amazon S3

Per ulteriori informazioni sulla configurazione di uno stream con un bucket Amazon S3 come destinazione, consulta Amazon Data [S3DestinationConfiguration](#) Firehose API Reference.

Per ulteriori informazioni sulla trasformazione degli eventi e sulla configurazione del buffering degli eventi, della compressione degli eventi e della crittografia degli eventi, consulta [Creating an Delivery Stream](#) nella Amazon Data Firehose Developer Guide.

Per ulteriori informazioni su come filtrare gli eventi, consulta [Policy di filtro degli abbonamenti Amazon SNS](#) in questa guida.

1. Accedi alla [console AWS Lambda](#).
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:
 - a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
 - b. Cercare fork-event-storage-backup-pipeline e scegliere l'applicazione.
4. Nella pagina fork-event-storage-backup-pipeline, procedi come segue:

- a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio my-app-backup).

 Note

- Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se riutilizzi il nome di un'applicazione, la distribuzione aggiornerà solo lo CloudFormation stack precedentemente distribuito (anziché crearne uno nuovo).

- b. (Facoltativo) Per BucketArn, inserisci l'ARN del bucket Amazon S3 in cui vengono caricati gli eventi in entrata. Se non inserisci un valore, nel tuo account viene creato un nuovo bucket Amazon S3. AWS
- c. (Facoltativo) Per DataTransformationFunctionArn, inserisci l'ARN della funzione Lambda attraverso la quale vengono trasformati gli eventi in entrata. In assenza di valori immessi, la trasformazione dei dati è disabilitata.
- d. (Facoltativo) Immettete una delle seguenti LogLevelimpostazioni per l'esecuzione della funzione Lambda dell'applicazione:
 - DEBUG
 - ERROR
 - INFO (predefinito)
 - WARNING
- e. Per esempio TopicArn, inserisci l'ARN dell'argomento Amazon SNS a cui deve essere sottoscritta questa istanza della pipeline fork.
- f. (Facoltativo) Per StreamBufferingIntervallInSecondse StreamBufferingSizeInMBs, inserisci i valori per configurare il buffering degli eventi in arrivo. In assenza di valori immessi, vengono impostati 300 secondi e 5 MB.
- g. (Facoltativo) Immettete una delle seguenti StreamCompressionFormatimpostazioni per la compressione degli eventi in arrivo:
 - GZIP
 - SNAPPY
 - UNCOMPRESSED (predefinito)
 - ZIP

- h. (Facoltativo) Per `StreamPrefix`, inserisci il prefisso di stringa per denominare i file archiviati nel bucket di backup Amazon S3. In assenza di valori immessi, non viene utilizzato alcun prefisso.
- i. (Facoltativo) Per `SubscriptionFilterPolicy`, inserisci la politica di filtro degli abbonamenti Amazon SNS, in formato JSON, da utilizzare per filtrare gli eventi in arrivo. La politica di filtro decide quali eventi sono indicizzati nell'indice del servizio. OpenSearch In assenza di valori immessi, non viene applicato alcun filtro (tutti gli eventi vengono indicizzati).
- j. (Facoltativo) Per `SubscriptionFilterPolicyScope`, inserisci la stringa `MessageBody` o per `MessageAttributes` abilitare il filtraggio dei messaggi basato sul payload o sugli attributi.
- k. Scegliere `I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.` (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.), quindi selezionare `Deploy` (Distribuisci).

Nella *my-app* pagina Stato di distribuzione per, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, CloudFormation inizia a creare lo stack e visualizza lo stato `CREATE_IN_PROGRESS` per ogni risorsa. Quando il processo è completo, visualizza lo stato `CREATE_COMPLETE`. CloudFormation

Al termine della distribuzione, Lambda mostra lo stato `Your application has been deployed` (L'applicazione è stata distribuita).

I messaggi pubblicati sul tuo argomento Amazon SNS vengono archiviati automaticamente nel bucket di backup Amazon 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 integrazione nativa con Amazon Data Firehose. Puoi abbonare i flussi di distribuzione Firehose agli argomenti SNS, il che ti consente di inviare notifiche a endpoint di archiviazione e analisi come i bucket Amazon Simple Storage Service (Amazon S3), le tabelle Amazon Redshift, Amazon Service (Service) e altro ancora. OpenSearch OpenSearch L'uso di Amazon SNS con i flussi di distribuzione Firehose è una soluzione completamente gestita e priva di codice che non richiede

l'utilizzo di funzioni. AWS Lambda Per ulteriori informazioni, consulta [Flussi di distribuzione da Fanout a Firehose](#).

Questo tutorial mostra come distribuire la [pipeline di ricerca di eventi e di analisi](#) e iscrivere a un argomento Amazon SNS. Questo processo trasforma automaticamente il AWS SAM modello associato alla pipeline in uno 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:

- Coda Amazon SQS
- funzione Lambda
- Flussi di distribuzione Firehose
- Dominio Amazon OpenSearch Service
- Bucket di Amazon S3

Per ulteriori informazioni sulla configurazione di uno stream con un indice come destinazione, consulta [ElasticsearchDestinationConfiguration](#) Amazon Data Firehose API Reference.

Per ulteriori informazioni sulla trasformazione degli eventi e sulla configurazione del buffering degli eventi, della compressione degli eventi e della crittografia degli eventi, consulta [Creating an Delivery Stream](#) nella Amazon Data Firehose Developer Guide.

Per ulteriori informazioni su come filtrare gli eventi, consulta [Policy di filtro degli abbonamenti Amazon SNS](#) in questa guida.

1. Accedi alla [console AWS Lambda](#).
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:
 - a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
 - b. Cercare fork-event-search-analytics-pipeline e scegliere l'applicazione.
4. Nella pagina fork-event-search-analytics-pipeline, procedi come segue:

- a. Nella sezione Application settings (Impostazioni applicazione), immettere un Application name (Nome applicazione) (ad esempio my-app-search).

 Note

Il nome dell'applicazione deve essere univoco per ogni distribuzione. Se riutilizzi il nome di un'applicazione, la distribuzione aggiornerà solo lo CloudFormation stack precedentemente distribuito (anziché crearne uno nuovo).

- b. (Facoltativo) Per DataTransformationFunctionArn, immettete l'ARN della funzione Lambda utilizzata per trasformare gli eventi in entrata. In assenza di valori immessi, la trasformazione dei dati è disabilitata.
- c. (Facoltativo) Immettete una delle seguenti LogLevelimpostazioni per l'esecuzione della funzione Lambda dell'applicazione:
 - DEBUG
 - ERROR
 - INFO (predefinito)
 - WARNING
- d. (Facoltativo) Per SearchDomainArn, inserisci l'ARN del dominio del OpenSearch servizio, un cluster che configura le funzionalità di elaborazione e archiviazione necessarie. In assenza di valori immessi, viene creato un nuovo dominio con la configurazione predefinita.
- e. Per esempio TopicArn, inserisci l'ARN dell'argomento Amazon SNS a cui deve essere sottoscritta questa istanza della pipeline fork.
- f. Per SearchIndexName, inserisci il nome dell'indice dei OpenSearch servizi per la ricerca e l'analisi degli eventi.

 Note

I seguenti limiti si applicano ai nomi degli indici:

- Non possono includere lettere maiuscole
- Non possono includere i seguenti caratteri: \ / * ? " < > | ` , #
- Non possono iniziare con i seguenti caratteri: - + _
- Non possono corrispondere a: . . .

- Non possono essere più lunghi di 80 caratteri
- Non possono essere più lunghi di 255 byte
- Non può contenere due punti (da OpenSearch Service 7.0)

g. (Facoltativo) Immettete una delle seguenti `SearchIndexRotationPeriod` impostazioni per il periodo di rotazione dell'indice di OpenSearch servizio:

- `NoRotation` (predefinito)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

La rotazione dell'indice aggiunge un timestamp al nome dell'indice, facilitando così la scadenza dei dati meno recenti.

h. Per `SearchTypeName`, inserisci il nome del tipo di OpenSearch servizio per l'organizzazione degli eventi in un indice.

Note

- OpenSearch I nomi dei tipi di servizio possono contenere qualsiasi carattere (eccetto i byte nulli) ma non possono iniziare con. _
- Per OpenSearch Service 6.x, può esistere un solo tipo per indice. Se si specifica un nuovo tipo per un indice esistente che ha già un altro tipo, Firehose restituisce un errore di runtime.

i. (Facoltativo) Per `StreamBufferingIntervalInSeconds` `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 `StreamCompressionFormat` impostazioni per la compressione degli eventi in arrivo:

- `GZIP`
- `SNAPPY`

- UNCOMPRESSED (predefinito)
 - ZIP
- k. (Facoltativo) Per `StreamPrefix`, inserisci il prefisso di stringa per denominare i file archiviati nel bucket con lettera morta di Amazon S3. In assenza di valori immessi, non viene utilizzato alcun prefisso.
 - l. (Facoltativo) Per `StreamRetryDurationInSeconds`, inserire la durata dei nuovi tentativi nei casi in cui Firehose non è in grado di indicizzare gli eventi nell'indice OpenSearch dei servizi. In assenza di valori immessi, viene utilizzato un valore di 300 secondi.
 - m. (Facoltativo) Per `SubscriptionFilterPolicy`, inserisci la politica di filtro degli abbonamenti Amazon SNS, in formato JSON, da utilizzare per filtrare gli eventi in arrivo. La politica di filtro decide quali eventi sono indicizzati nell'indice del servizio. OpenSearch In assenza di valori immessi, non viene applicato alcun filtro (tutti gli eventi vengono indicizzati).
 - n. Scegliere `I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.` (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.), quindi selezionare `Deploy` (Distribuisci).

Nella *my-app-search* pagina Stato di distribuzione per, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, CloudFormation inizia a creare lo stack e visualizza lo stato `CREATE_IN_PROGRESS` per ogni risorsa. Quando il processo è completo, visualizza lo stato `CREATE_COMPLETE`. CloudFormation

Al termine della distribuzione, Lambda mostra lo stato `Your application has been deployed` (L'applicazione è stata distribuita).

I messaggi pubblicati sul tuo argomento Amazon SNS vengono indicizzati automaticamente nell'indice dei OpenSearch servizi fornito dalla pipeline Event Search and Analytics. Se la pipeline non è in grado di indicizzare un evento, lo archivia in un bucket con lettera morta di Amazon S3.

Implementazione della Event Replay Pipeline con l'integrazione di Amazon SNS

Questa pagina mostra come distribuire la [pipeline di riproduzione eventi](#) e iscrivere a un argomento Amazon SNS. Questo processo trasforma automaticamente il AWS SAM modello associato alla pipeline in uno 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 riproduzione eventi, tra cui una coda Amazon SQS e una funzione Lambda.

Per ulteriori informazioni su come filtrare gli eventi, consulta [Policy di filtro degli abbonamenti Amazon SNS](#) in questa guida.

1. Accedi alla [console AWS Lambda](#).
2. Nel riquadro di navigazione, selezionare Functions (Funzioni) e quindi Create function (Crea funzione).
3. Nella pagina Create function (Crea funzione), procedere come segue:
 - a. Scegliere Browse serverless app repository (Sfoglia repository app serverless), Public applications (Applicazioni pubbliche), Show apps that create custom IAM roles or resource policies (Mostra applicazioni in grado di creare ruoli IAM o policy di risorse personalizzati).
 - b. Cercare `fork-event-replay-pipeline` e scegliere l'applicazione.
4. Nella `fork-event-replay-pipeline` pagina, 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 CloudFormation stack precedentemente distribuito (anziché crearne uno nuovo).

- b. (Facoltativo) Immettete una delle seguenti LogLevel impostazioni per l'esecuzione della funzione Lambda dell'applicazione:
 - DEBUG
 - ERROR
 - INFO (predefinito)
 - WARNING
 - c. (Facoltativo) Per `ReplayQueueRetentionPeriodInSeconds`, inserisci il periodo di tempo, in secondi, per il quale la coda di replay di Amazon SQS conserva il messaggio. In assenza di valori immessi, viene utilizzato un valore di 1.209.600 secondi (14 giorni).
 - d. Per esempio `TopicArn`, inserisci l'ARN dell'argomento Amazon SNS a cui deve essere sottoscritta questa istanza della pipeline fork.

- e. Per `DestinationQueueName`, inserisci il nome della coda Amazon SQS a cui la funzione di replay Lambda inoltra i messaggi.
- f. (Facoltativo) Per `SubscriptionFilterPolicy`, inserisci la politica di filtro degli abbonamenti Amazon SNS, in formato JSON, da utilizzare per filtrare gli eventi in arrivo. Tale policy definisce quali eventi vengono aggiunti al buffer per la riproduzione. In assenza di valori immessi, non viene applicato alcun filtro (tutti gli eventi vengono aggiunti al buffer per la riproduzione).
- g. Scegliere `I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.` (Riconosco che questa app crea ruoli IAM e policy di risorse personalizzati e distribuisce applicazioni nidificate.), quindi selezionare `Deploy` (Distribuisci).

Nella *my-app-replay* pagina Stato di distribuzione per, Lambda visualizza lo stato L'applicazione è in fase di distribuzione.

Nella sezione Risorse, CloudFormation inizia a creare lo stack e visualizza lo stato `CREATE_IN_PROGRESS` per ogni risorsa. Quando il processo è completo, visualizza lo stato `CREATE_COMPLETE`. CloudFormation

Al termine della distribuzione, Lambda mostra lo stato `Your application has been deployed` (L'applicazione è stata distribuita).

I messaggi pubblicati nell'argomento Amazon SNS vengono aggiunti al buffer per la riproduzione nella coda Amazon SQS assegnata automaticamente dalla pipeline di riproduzione eventi.

Note

La riproduzione è disabilitata per impostazione predefinita. Per abilitarla, accedi alla pagina della funzione sulla console Lambda, espandi la sezione Designer, seleziona il riquadro SQS e quindi, nella sezione SQS, scegli `Enabled` (Abilitata).

Utilizzo 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. Puoi configurare finestre temporali flessibili per la consegna,

definire limiti per nuovi tentativi e impostare il tempo massimo di conservazione per invocazioni API non riuscite.

Questa pagina spiega come usare EventBridge Scheduler per pubblicare un messaggio da un argomento di Amazon SNS secondo una pianificazione.

Impostazione del ruolo di esecuzione

Quando crei una nuova EventBridge pianificazione, Scheduler deve avere l'autorizzazione a richiamare l'operazione API 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'API di destinazione che si desidera che EventBridge Scheduler richiami.

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 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 e richiama una destinazione con una frequenza specificata utilizzando un'espressione cron o un'espressione della frequenza.	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 Disattiva 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'operazione AWS API richiamata da Scheduler: EventBridge
 - a. Scegli Pubblicazione di Amazon SNS.
 - b. Nella sezione Pubblica, seleziona un argomento SNS o scegli Crea un nuovo argomento SNS.
 - c. (Facoltativo) Inserisci un payload JSON. Se non inserisci un payload, 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 policy di ripetizione per la tua pianificazione, in Policy di ripetizione e coda DLQ (Dead-Letter Queue) effettua le seguenti operazioni:
 - 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).	
Archivia l'evento nello stesso spazio in Account AWS cui stai creando il programma	a. Scegli Seleziona una coda Amazon SQS in my Account AWS as a DLQ. b. Scegli il nome della risorsa Amazon (ARN) della coda di Amazon SQS.	
Archivia l'evento in un luogo diverso Account AWS da quello in cui stai creando il programma	a. Scegli Specificare una coda Amazon SQS tra le altre Account AWS come DLQ. b. Inserisci il nome della risorsa Amazon (ARN) della coda di Amazon SQS.	

- d. Per utilizzare una chiave gestita dal cliente per crittografare l'input di destinazione, in Crittografia scegli Personalizza le impostazioni di crittografia (avanzate).

Se si sceglie questa opzione, inserire l'ARN di una chiave KMS esistente o scegliere Crea una AWS KMS key per accedere alla console AWS KMS . 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

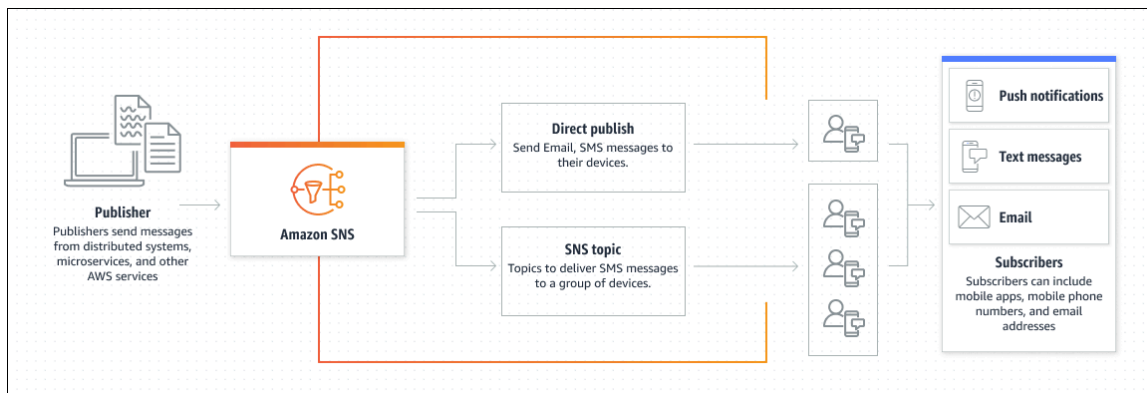
Per ulteriori informazioni su EventBridge Scheduler, consulta quanto segue:

- [EventBridge Guida per l'utente di Scheduler](#)
- [EventBridge Riferimento all'API Scheduler](#)
- [EventBridge Prezzi Scheduler](#)

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

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 funzionalità, puoi inviare notifiche push alle app mobili, messaggi di testo ai numeri di cellulare e messaggi di 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.

Questo argomento spiega come utilizzare Amazon SNS per le notifiche degli utenti con abbonati come applicazioni mobili, numeri di cellulare e indirizzi e-mail.



Messaggi di testo mobili con Amazon SNS

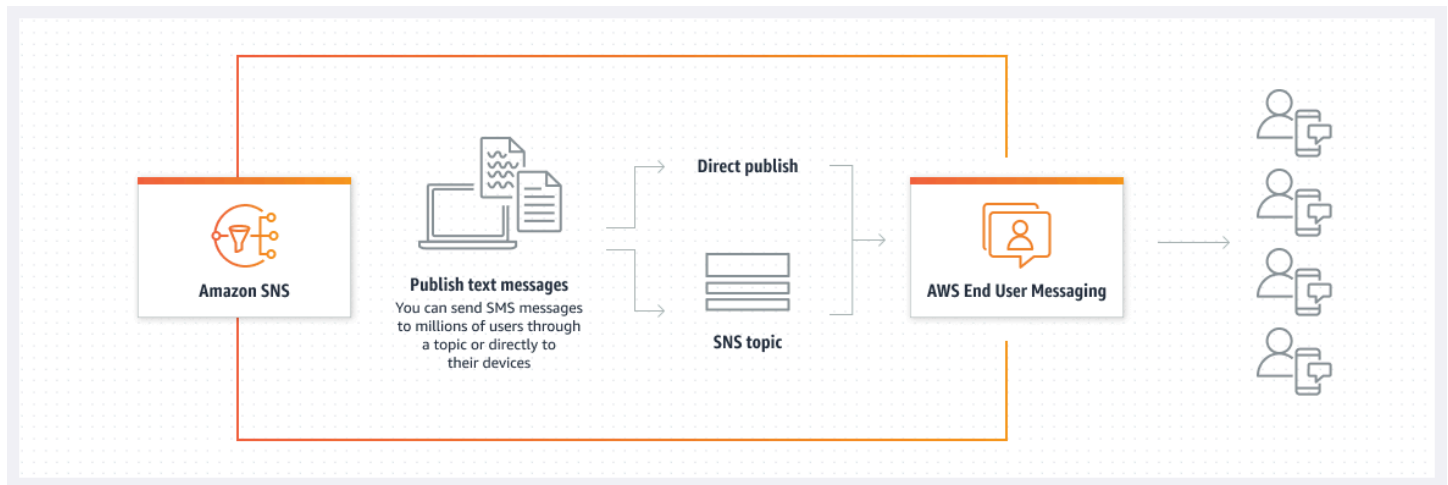
⚠ Important

La Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS è integrato [AWS End User Messaging SMS](#) per la consegna di messaggi SMS. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i messaggi SMS di Amazon SNS.

La messaggistica di testo mobile (SMS) di Amazon SNS è progettata per facilitare la consegna dei messaggi su varie piattaforme, come applicazioni Web, mobili e aziendali che supportano gli SMS. Gli utenti possono inviare messaggi a uno o più numeri di telefono abbonandoli a un argomento, semplificando il processo di distribuzione.

I messaggi Amazon SNS vengono recapitati da AWS End User Messaging SMS, il che garantisce una trasmissione affidabile dei messaggi. [All'interno di Amazon SNS APIs, puoi impostare varie proprietà come tipi di messaggi \(promozionali o transazionali\), limiti di spesa mensili, elenchi di opt-out e 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 globale di fornitura di SMS. Gestisce il routing, lo stato di consegna e qualsiasi richiesta di conformità alle normative regionali. [Per accedere a funzionalità SMS aggiuntive come autorizzazioni granulari, pool di telefoni, set di configurazioni, simulatore SMS e regole nazionali, consulta la Guida per l'utente.AWS End User Messaging SMS](#)



Le seguenti funzionalità chiave ti aiutano a inviare messaggi SMS Amazon SNS scalabili e facilmente estensibili:

[Personalizza le preferenze dei messaggi](#)

Personalizza le consegne di SMS per te Account AWS impostando le preferenze relative agli SMS in base al tuo budget e al tuo 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 consegne di SMS specificando le quote di spesa o per il recapito di singoli messaggi e le quote di spesa mensili per te. Account AWS Laddove richiesto dalle leggi e dai regolamenti locali (come Stati Uniti e Canada), i destinatari degli SMS possono [rinunciare](#), il che significa che scelgono di non ricevere più messaggi SMS dall'utente. Account AWS Dopo che un destinatario ha scelto di non ricevere messaggi, puoi, con alcune limitazioni, inserire nuovamente il numero di telefono in modo da poter riprendere a inviare messaggi.

[Invia messaggi SMS a livello globale](#)

Amazon SNS supporta la messaggistica SMS in più regioni, consentendoti di inviare messaggi in oltre 240 paesi e regioni.

In che modo Amazon SNS consegna i miei messaggi SMS?

Quando richiedi ad Amazon SNS di inviare SMS per tuo 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:

[Politiche IAM](#)

Puoi sfruttare IAM e le policy relative alle risorse per controllare e distribuire l'accesso alle tue risorse SMS tra altri AWS servizi e regioni.

Configurazioni [AWS End User Messaging SMS](#)

Tutte le configurazioni relative all'ID di origine (creazione, aggiornamento della configurazione, provisioning di una nuova origine IDs, modifica dei modelli di registrazione) utilizzano AWS End User Messaging SMS.

[AWS End User Messaging SMS fatturazione](#)

Tuttavia, tutta la fatturazione via SMS viene effettuata da AWS End User Messaging SMS. Puoi consolidare la AWS spesa per i carichi di lavoro relativi agli SMS, procurando e gestendo le risorse SMS in un'unica posizione centrale.

Guida introduttiva ad Amazon SNS SMS

Important

La Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS è integrato [AWS End User Messaging SMS](#) per la consegna di messaggi SMS. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i messaggi SMS di Amazon SNS.

Questo argomento ti guida nella gestione della sandbox SMS e nella configurazione di IAM e policy basate sulle risorse per concedere ad Amazon SNS le autorizzazioni necessarie per accedere e utilizzare. AWS End User Messaging SMS APIs

Prerequisiti

Amazon SNS consiglia di aggiornare la tua policy IAM per includere le seguenti azioni per garantire il controllo e la visibilità completi sulle tue risorse Amazon SNS:

- [AmazonSNSFullAccess](#)
- [AmazonSNSReadOnly](#)

Utilizzo della sandbox SMS di Amazon SNS

Gli account SMS Amazon SNS appena creati vengono inseriti automaticamente nella sandbox SMS 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 sandbox SMS, hai accesso a tutte le funzionalità di Amazon SNS ma sei soggetto a determinate limitazioni:

- Puoi inviare messaggi SMS solo a numeri di telefono di destinazione verificati.
- Puoi avere un massimo di 10 numeri di telefono di destinazione verificati.
- Puoi eliminare i numeri di telefono di destinazione solo dopo 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 messaggi SMS a qualsiasi destinatario.

Fase iniziale

I nuovi account Amazon SNS SMS vengono inseriti in una sandbox SMS. Utilizza i seguenti passaggi per creare e gestire i numeri di telefono nella sandbox, creare numeri di origine e mittente IDs e registrare la tua 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 dalla sandbox SMS di Amazon SNS, consulta. [Aggiungere e verificare numeri di telefono nella sandbox SMS di Amazon SNS](#)

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 SMS di Amazon SNS](#)
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 messaggistica SMS 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 di messaggi SMS tramite Amazon SNS](#).

Aggiungere e verificare numeri di telefono nella sandbox SMS di Amazon SNS

Prima di poter iniziare a inviare messaggi SMS Account AWS mentre ti trovi nella [sandbox SMS](#), devi completare i seguenti passaggi di configurazione. Ciò garantisce che il tuo account sia pronto per la messaggistica SMS e che i numeri di telefono di destinazione siano verificati correttamente.

1. Crea un [ID di origine](#). Analogamente agli account esterni alla sandbox SMS, è necessario un ID di origine prima di poter inviare messaggi SMS a destinatari in alcuni paesi o aree geografiche.
2. Aggiungi i numeri di telefono di destinazione a cui desideri inviare messaggi all'interno della sandbox SMS.
3. Verifica i numeri di telefono per assicurarti che i numeri di telefono di destinazione siano validi per l'uso nei tuoi messaggi SMS.

Aggiungi e verifica i numeri di telefono di destinazione

1. Accedi alla [console Amazon SNS](#).
2. Nel menu della console, scegli una [regione che supporti la messaggistica SMS](#).
3. Nel riquadro di navigazione, seleziona Text messaging (SMS) (SMS).
4. Nella sezione Numeri di telefono di destinazione Sandbox, seleziona Aggiungi numero di telefono.
5. In Dettagli della destinazione, fornisci le seguenti informazioni, quindi seleziona Aggiungi numero di telefono:
 - Codice del paese e numero di telefono della destinazione.

- La lingua in cui desideri che venga inviato il messaggio di verifica.
6. Dopo aver aggiunto il numero di telefono, Amazon SNS invierà una OTP al numero di telefono di destinazione fornito. Questa OTP è necessaria per la verifica.
 7. Riceverai l'OTP come messaggio SMS standard sul numero di telefono di destinazione che hai fornito.
 - Se non ricevi l'OTP entro 15 minuti, seleziona Reinvia il codice di verifica nella console Amazon SNS.
 - Puoi inviare nuovamente l'OTP fino a cinque volte in un periodo di 24 ore.
 8. Dopo aver ricevuto l'OTP, inseriscilo nella casella Codice di verifica e seleziona Verifica numero di telefono.
 9. Controlla lo stato della verifica.
 - Dopo aver verificato correttamente il numero di telefono, il numero di telefono e il relativo stato di verifica verranno visualizzati nella sezione dei numeri di telefono di destinazione di Sandbox.
 - Se lo stato è In sospeso, la verifica non ha avuto esito positivo. Ciò può accadere se, ad esempio, non hai inserito correttamente il prefisso internazionale.
 - Puoi eliminare i numeri di telefono in sospeso o verificati solo dopo 24 ore o più dall'ultimo tentativo di verifica.
 10. Se desideri utilizzare lo stesso numero di telefono di destinazione in altre aree geografiche, ripeti i passaggi precedenti per ogni regione in cui intendi utilizzarlo.

Risoluzione dei problemi relativi alla mancata ricezione di un testo OTP

Risolvi i problemi più comuni che possono impedire a un numero di telefono di ricevere messaggi OTP.

- Limite di spesa SMS di Amazon SNS: se hai Account AWS superato il limite di spesa per l'invio di messaggi SMS, è possibile che altri messaggi, inclusi gli SMS OTP, non vengano recapitati finché il limite non viene aumentato o il problema di fatturazione non viene risolto.
- Numero di telefono non utilizzato per le notifiche SMS: in alcuni paesi o regioni, i destinatari devono attivare la ricezione di messaggi SMS da codici brevi, che vengono comunemente utilizzati per i messaggi OTP. Se il numero di telefono del destinatario non è abilitato, non riceverà il testo OTP.
- Restrizioni o filtri dell'operatore: alcuni operatori di telefonia mobile potrebbero disporre di restrizioni o meccanismi di filtraggio che impediscono la consegna di determinati tipi di messaggi SMS, inclusi

i messaggi OTP. Ciò potrebbe essere dovuto alle politiche di sicurezza o alle misure antispam implementate dal gestore.

- Numero di telefono non valido o errato: se il numero di telefono fornito dal destinatario è errato o non valido, il testo OTP non verrà recapitato.
- Problemi di rete: problemi o interruzioni temporanee della rete potrebbero impedire la consegna di messaggi SMS, inclusi SMS OTP, al telefono del destinatario.
- Consegna ritardata: in alcuni casi, i messaggi SMS possono subire ritardi nella consegna a causa della congestione della rete o di altri fattori. Il testo OTP potrebbe alla fine essere recapitato, ma potrebbe subire ritardi oltre il periodo di tempo previsto.
- Sospensione o chiusura dell'account: in caso di problemi con il tuo account Account AWS, come il mancato pagamento o la violazione dei AWS termini di servizio, le funzionalità di messaggistica di Amazon SNS, inclusi gli SMS OTP, potrebbero essere sospese o interrotte.

Eliminazione di numeri di telefono dalla sandbox SMS di Amazon SNS

Puoi eliminare sia i numeri di telefono di destinazione in sospeso che quelli verificati dalla [sandbox SMS](#).

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 [console Amazon SNS](#).
2. Nel menu della console, scegli una [regione che supporti la messaggistica SMS](#) a 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 del 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 sandbox SMS di Amazon SNS

Account AWS Per uscire dalla [sandbox SMS](#) è necessario innanzitutto aggiungere, verificare e testare i numeri di telefono di destinazione. Dopo aver fatto ciò, crea una custodia con Supporto AWS.

Per richiedere che il tuo AWS account venga rimosso dalla sandbox SMS

1. Verifica i numeri di telefono
 - a. Mentre ti Account AWS trovi nella sandbox SMS, apri la console [Amazon SNS](#).
 - b. Nel pannello di navigazione, sotto Mobile, scegli Messaggi di testo (SMS).
 - c. Nella sezione Sandbox dedicata ai numeri di telefono di destinazione, [aggiungi e verifica](#) uno o più numeri di telefono di destinazione. Questa verifica garantisce che tu possa inviare e ricevere messaggi con successo.
2. Prova la pubblicazione di SMS
 - Conferma di essere in grado di inviare e ricevere messaggi ad almeno un numero di telefono verificato. Per istruzioni più dettagliate su come pubblicare messaggi SMS, consulta [Pubblicazione di messaggi SMS su un telefono cellulare tramite Amazon SNS](#).
3. Avvia la modifica della sandbox
 - Nella pagina Messaggi di testo per dispositivi mobili (SMS) della console Amazon SNS, sotto Informazioni sull'account, scegliere Exit SMS sandbox (Uscire dalla sandbox SMS). Questa azione ti reindirizza all'Amazon [Support Center](#) e crea automaticamente un caso di supporto con l'opzione di aumento della quota di servizio selezionata.
4. Compila il modulo
 - Nel modulo di assistenza sotto Aumento della quota di servizio, procedi come segue:
 - i. Scegli «SNS Text Messaging» come servizio.
 - ii. Fornisci l'URL del sito Web o il nome dell'app da cui intendi inviare messaggi SMS.

- iii. Specificate il tipo di messaggi che invierete: password monouso, promozionale o transazionale.
- iv. Scegli il tipo Regione AWS da cui inviare i messaggi SMS.
- v. Elenca i paesi o le regioni in cui intendi inviare messaggi SMS.
- vi. Descrivi in che modo i tuoi clienti scelgono di ricevere messaggi.
- vii. Includi tutti i modelli di messaggio che intendi utilizzare.

5. Specificare quota e regione

- In Requests (Richieste), procedere come segue:
 - i. Scegli Regione AWS dove vuoi spostare il tuo Account AWS.
 - ii. Scegli i limiti generali per il tipo di risorsa.
 - iii. Scegli Exit SMS Sandbox per Quota.
 - iv. (Facoltativo) Per richiedere ulteriori aumenti o altri aggiustamenti, scegli Aggiungi un'altra richiesta e specifica i dettagli necessari.
 - v. In Nuovo valore di quota, inserisci il limite in USD che stai richiedendo.

6. Dettagli aggiuntivi

- a. Nella descrizione del caso, fornisci eventuali dettagli aggiuntivi pertinenti alla tua richiesta.
- b. In Opzioni di contatto, scegli la lingua di contatto preferita.

7. Invia la richiesta

- Scegli Invia a cui inviare la richiesta Supporto.

Il Supporto 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.

Identità di origine per i messaggi SMS di Amazon SNS

Important

La Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS è integrato con [AWS End User Messaging SMS](#) per la consegna di messaggi SMS. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i messaggi SMS di Amazon SNS.

Le identità di origine per i messaggi SMS sono identificatori utilizzati per rappresentare il mittente di un messaggio SMS. Puoi identificarti 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 messaggio SMS. 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 (TFN) e codici brevi (numeri di telefono che contengono da quattro a sette cifre).

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. Quando si invia un messaggio SMS utilizzando un numero di origine, il dispositivo del destinatario mostra il numero di origine come numero di telefono del mittente. È possibile specificare diversi numeri di origine in base al caso d'uso.

Per ulteriori informazioni, consulta [Numeri di telefono nella Guida per l'AWS End User Messaging SMS utente](#).

Tip

Per visualizzare un elenco di tutti i numeri di origine esistenti nel tuo AWS account, nel riquadro di navigazione della console [Amazon SNS](#), scegli Origination numbers.

Mittente IDs

Un nome alfabetico che identifica il mittente di un messaggio SMS. Quando invii un messaggio SMS 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. Un ID mittente offre ai destinatari di SMS maggiori informazioni sul mittente rispetto a un numero di telefono o un codice lungo o breve.

IDs I mittenti sono supportati in diversi paesi e aree geografiche in tutto il mondo. In alcuni, un'azienda che invia messaggi SMS a singoli clienti deve utilizzare un ID mittente preregistrato presso un ente normativo o un gruppo di settore. Per un elenco completo dei Paesi e delle aree geografiche che supportano o richiedono il mittente IDs, consulta [Paesi e aree geografiche supportati per la messaggistica SMS AWS End User Messaging SMS nella Guida](#) per l'AWS End User Messaging SMS utente.

Non sono previsti costi aggiuntivi per l'utilizzo del mittente IDs. 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 d'America) non supportano l'utilizzo del IDs mittente. In alcuni, le aziende che inviano messaggi SMS a singoli clienti devono utilizzare un ID mittente preregistrato presso un ente normativo o un gruppo di settore.

Per ulteriori informazioni, consulta [Sender IDs nella Guida](#) per l'AWS End User Messaging SMS utente.

Configurazione della messaggistica SMS in Amazon SNS

Important

La Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS è integrato con [AWS End User Messaging SMS](#) per la consegna di messaggi SMS. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i messaggi SMS di Amazon SNS.

Puoi utilizzare le configurazioni di Amazon SNS SMS per impostare le preferenze SMS in base alle tue esigenze, ad esempio regolando le quote di spesa e impostando la registrazione dello stato di consegna. Questo argomento fornisce anche dettagli su come pubblicare messaggi SMS su argomenti utilizzando la console e l' AWS SDK di Amazon SNS, gestire in modo efficiente le quote e recuperare statistiche dettagliate sull'attività degli SMS.

Invio di messaggi SMS tramite Amazon SNS

Questa sezione descrive come inviare messaggi SMS utilizzando Amazon SNS, 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.

Pubblicazione di messaggi SMS su un argomento di Amazon SNS

Puoi inviare un singolo messaggio SMS a più numeri di telefono contemporaneamente sottoscrivendo tali numeri in un argomento Amazon SNS. Un argomento SNS è un canale di comunicazione dove è possibile aggiungere gli abbonati ai quali si vogliono indirizzare determinati messaggi. Un abbonato riceve tutti i messaggi pubblicati sull'argomento fino a quando non annulli l'abbonamento o finché l'abbonato non sceglie di non ricevere messaggi SMS dal tuo account. AWS

Invio di un messaggio a un argomento tramite la console AWS

Per creare un argomento

Se ancora non disponi di un argomento al quale inviare messaggi SMS, completa i seguenti passaggi.

1. Accedi alla [console Amazon SNS](#).
2. Nel menu della console, scegli una [regione che supporti la messaggistica SMS](#).
3. Nel pannello di navigazione, scegli Argomenti.
4. Nella pagina Topics (Argomenti), seleziona Create new topic (Crea nuovo argomento).
5. Nella pagina Create topic (Crea argomento), in Details (Dettagli), effettuare le seguenti operazioni:
 - a. Per Tipo, scegliere Standard.
 - b. In Name (Nome), immettere un nome.
 - c. (Facoltativo) In Display name (Nome visualizzato), inserisci un prefisso personalizzato per il messaggio SMS. Quando invii un messaggio all'argomento, Amazon SNS antepone il nome visualizzato a una parentesi angolare destra (>) e uno spazio. I nomi visualizzati non sono sensibili alle maiuscole/minuscole, in quanto sarà poi Amazon SNS a convertirli in caratteri maiuscoli. Ad esempio, se il nome visualizzato di un argomento è MyTopic e il messaggio è Hello World!, il messaggio appare in questo formato:

```
MYTOPIC> Hello World!
```


6. Scegliere Create topic (Crea argomento). Il nome dell'argomento e l'Amazon Resource Name (ARN) vengono visualizzati nella pagina Topics (Argomenti)

Per creare una sottoscrizione SMS

Con le sottoscrizioni puoi inviare un messaggio SMS a più destinatari pubblicandolo una volta sola nel tuo argomento.

 Note

Quando inizi a utilizzare Amazon SNS per inviare messaggi SMS, il tuo AWS account si trova nella sandbox SMS. La sandbox SMS offre un ambiente sicuro per provare le funzionalità Amazon SNS senza rischiare la reputazione del mittente SMS. Quando il tuo account è nella sandbox SMS, puoi utilizzare tutte le caratteristiche di Amazon SNS, ma puoi inviare SMS solo a numeri di telefono di destinazione verificati. Per ulteriori informazioni, consulta [Utilizzo della sandbox SMS di Amazon SNS](#).

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione scegli Subscriptions (Sottoscrizioni).
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Create subscription (Crea sottoscrizione), nella sezione Details (Dettagli), eseguire queste operazioni:
 - a. Per Topic ARN (ARN argomento), inserisci o seleziona l'Amazon Resource Name (ARN) dell'argomento a cui desideri inviare messaggi SMS.
 - b. In Protocol (Protocollo), scegli SMS.
 - c. Per Endpoint, inserisci il numero di telefono a cui desideri iscriverti all'argomento.
5. Scegli Create Subscription (Crea sottoscrizione). Le informazioni sull'abbonamento vengono visualizzate nella pagina Subscriptions (Abbonamenti).

Per aggiungere altri numeri di telefono, ripetere questi passaggi. Puoi anche aggiungere altri tipi di sottoscrizione, ad esempio le e-mail.

Come inviare un messaggio

Quando pubblichi un messaggio in un argomento, Amazon SNS prova a consegnare il messaggio a tutti i numeri di telefono che hanno effettuato la sottoscrizione a un dato argomento.

1. Nella [Console Amazon SNS](#), nella pagina Topics (Argomenti), scegli il nome dell'argomento a cui desideri inviare messaggi SMS.
2. Nella pagina dettagli, seleziona Publish message (Pubblica messaggio).
3. Nella pagina Publish message to topic (Pubblica messaggio nell'argomento), alla sezione Message details (Dettagli messaggio) procedi come indicato di seguito:
 - a. Per Subject (Oggetto), lascia il campo vuoto a meno che il tuo argomento non contenga anche sottoscrizioni e-mail e tu non voglia pubblicare il messaggio sulle sottoscrizioni e-mail ed SMS. Amazon SNS utilizza il Subject (Oggetto) che inserisci come oggetto dell'e-mail.
 - b. (Facoltativo) Per Time to Live (TTL) (Durata), inserisci un numero di secondi che Amazon SNS deve inviare il tuo messaggio SMS a tutti gli abbonati agli endpoint delle applicazioni mobili.
4. Su Message body (Corpo del messaggio), procedere come segue:
 - a. Per Message structure (Struttura dei messaggi), scegliere Identical payload for all delivery protocols (Payload identico per tutti i protocolli di consegna) per inviare lo stesso messaggio a tutti i tipi di protocollo sottoscritti all'argomento. In alternativa, scegliere Custom payload for each delivery protocol (Payload personalizzato per ogni protocollo di consegna) per personalizzare il messaggio per i sottoscrittori di diversi tipi di protocollo. Ad esempio, è possibile immettere un messaggio predefinito per gli abbonati al numero di telefono e un messaggio personalizzato per i sottoscrittori di posta elettronica.
 - b. Per Message body to send to the endpoint (Corpo del messaggio da inviare all'endpoint), immettere il messaggio o i messaggi personalizzati per protocollo di recapito.

Se il tuo argomento ha un nome visualizzato, Amazon SNS lo aggiunge al messaggio, aumentando così la lunghezza del messaggio. La lunghezza del nome visualizzato dipende dal numero di caratteri nel nome più due per la parentesi angolare destra (>) e lo spazio aggiunto da Amazon SNS.

Per informazioni sulle dimensioni massime degli SMS, consulta [Pubblicazione di messaggi SMS 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 messaggio SMS e ne visualizza quindi la conferma.

Invio di un messaggio a un argomento utilizzando il 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 and](#) Tools Reference AWS SDKs Guide.

Il codice di esempio seguente mostra come fare per:

- Creazione di un argomento Amazon SNS.
- Sottoscrivere un numero di telefono cellulare all'argomento.
- Pubblicazione di messaggi SMS nell'argomento in modo che tutti i numeri di telefono sottoscritti ricevano il messaggio in una sola volta.

Java

SDK per Java 2.x

 Note

C'è 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 un argomento e restituire il suo ARN.

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```

    * For more information, see the following documentation topic:
    *
    * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
    */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {

```

```

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

```

Sottoscrivere un endpoint a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

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

            Usage:    <topicArn> <phoneNumber>

            Where:
                topicArn - The ARN of the topic to subscribe.
                phoneNumber - A mobile phone number that receives
notifications (for example, +1XXX5550100).
            "";

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];

```

```

        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("sms")
                .endpoint(phoneNumber)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
"\n\n Status is "
                + result.sdkHttpResponse().statusCode());

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

```

Impostare gli attributi del messaggio, ad esempio l'ID del mittente, il prezzo massimo e il relativo tipo. Gli attributi del messaggio sono facoltativi.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**

```

```

* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSMSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSMSAttributes(SnsClient snsClient, HashMap<String,
String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result =
snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
Status was "
                + result.sdkHttpResponse().statusCode());

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

```

Publicare un messaggio in un argomento. Il messaggio viene inviato a ogni abbonato.

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

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

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is
sent (for example, +1XXX5550100).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
```



```
try {
    PublishRequest request = PublishRequest.builder()
        .message(message)
        .phoneNumber(phoneNumber)
        .build();

    PublishResponse result = snsClient.publish(request);
    System.out
        .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

Pubblicazione di messaggi SMS su un telefono cellulare tramite Amazon SNS

Puoi utilizzare Amazon SNS per inviare messaggi SMS direttamente a un telefono cellulare senza sottoscrivere il numero di telefono a un argomento Amazon SNS.

Note

La sottoscrizione di numeri di telefono a un argomento è utile se desideri inviare un messaggio a più numeri di telefono contemporaneamente. Per istruzioni sulla pubblicazione di un messaggio SMS in un argomento, consulta [Pubblicazione di messaggi SMS su un argomento di Amazon SNS](#).

Quando invii un messaggio, puoi controllare se è ottimizzato relativamente al costo e all'affidabilità della consegna. È anche possibile specificare un [sender ID or origination number \(ID mittente o numero di origine\)](#). Se invii il messaggio in modo programmatico utilizzando l'API Amazon SNS o AWS SDKs il, puoi specificare un prezzo massimo per la consegna del messaggio.

Ogni messaggio SMS può contenere fino a 140 byte, mentre la quota di caratteri dipende dallo schema di codifica. Ad esempio, un messaggio SMS può contenere:

- 160 caratteri GSM

- 140 caratteri ASCII
- 70 caratteri UCS-2

Se il messaggio supera la quota della dimensione, Amazon SNS lo invia sotto forma di più messaggi, ciascuno nel rispetto della quota della dimensione indicata. I messaggi non vengono troncati nel mezzo di una parola ma tra una parola e l'altra. La quota della dimensione totale massima di una singola pubblicazione SMS è di 1600 byte.

Quando invii un SMS, specifichi il numero di telefono utilizzando il formato E.164, una struttura di numerazione standard utilizzata per le telecomunicazioni internazionali. I numeri di telefono che seguono questo formato possono avere un massimo di 15 cifre oltre al prefisso con il segno più (+) e il prefisso del paese. Ad esempio, un numero di telefono statunitense in formato E.164 viene visualizzato come +1. XXX5550100

Invio di un messaggio (Console)

1. Accedi alla [console Amazon SNS](#).
2. Nel menu della console, scegli una [regione che supporti la messaggistica SMS](#).
3. Nel riquadro di navigazione, seleziona Text messaging (SMS) (SMS).
4. Nella pagina Messaggi di testo per dispositivi mobili (SMS) scegli Pubblicazione di testo.
5. Nella pagina Pubblicazione di messaggi SMS, per Tipo di messaggio scegliere una delle opzioni seguenti:
 - Promotional (Promozionali) - Messaggi non critici, come i messaggi di marketing.
 - Transactional (Transazionali) - Messaggi critici che supportano le transazioni dei clienti, come le password monouso per l'autenticazione a più fattori.

Note

Questa impostazione a livello di messaggio sostituisce il tipo di messaggio predefinito a livello di account. È possibile impostare un tipo di messaggio predefinito a livello di account dalla sezione Preferenze di messaggistica della pagina Messaggi di testo per dispositivi mobili (SMS).

Per informazioni sulle tariffe relative a messaggi promozionali e transazionali, consulta la pagina relativa alle [tariffe SMS globali](#).

6. In Number (Numero), inserisci il numero di telefono al quale vuoi inviare il messaggio.
7. In Message (Messaggio), inserisci il testo da inviare.
8. (Facoltativo) Su Identità di origine, specificare come identificarsi ai destinatari:
 - Per specificare l'ID mittente, digita un ID personalizzato contenente da 3 a 11 caratteri alfanumerici, tra cui almeno una lettera e nessuno spazio. L'ID mittente viene visualizzato come mittente del messaggio sul dispositivo ricevente. Ad esempio, puoi utilizzare il tuo marchio commerciale per rendere più facilmente riconoscibile l'origine del messaggio.

Il supporto per il mittente IDs varia in base al Paese e all' and/or 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 mittente IDs, consulta [Paesi e aree geografiche supportati per la messaggistica SMS 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.
- Nei paesi in cui è supportato solo il mittente, IDs viene visualizzato l'AVVISO.

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 numero da 10 DLC, un numero verde, un codice person-to-person lungo o codici brevi. Per ulteriori informazioni, consulta [Identità di origine per i messaggi SMS di Amazon SNS](#).

Se non specifichi un numero di origine, Amazon SNS seleziona un numero di origine da utilizzare per il messaggio di testo SMS, in base alla configurazione di Account AWS .

9. Se invii messaggi SMS ai destinatari in India, espandi Attributi specifici del paese e specifica gli attributi seguenti:

- ID entità — L'ID entità o l'ID entità principale (PE) per l'invio di messaggi SMS ai destinatari in India. Questo ID è una stringa univoca di 1-50 caratteri fornita dalla Telecom Regulatory Authority of India (TRAI) per identificare l'entità registrata presso il TRAI.
- ID modello — L'ID modello per l'invio di messaggi SMS ai destinatari in India. Questo ID è una stringa univoca fornita da TRAI di 1-50 caratteri che identificherà il modello registrato con TRAI. L'ID modello deve essere associato all'ID mittente specificato per il messaggio.

Per ulteriori informazioni sull'invio di messaggi SMS a destinatari in [India, consulta la procedura di registrazione dell'ID mittente indiano nella Guida](#) per l'AWS End User Messaging SMS utente.

10. Seleziona Publish message (Pubblica messaggio).

Tip

Per inviare messaggi SMS da un numero di origine, puoi anche scegliere Numeri di origine nel pannello di navigazione della console Amazon SNS. Scegliere un numero di origine che includa SMS nella colonna Capabilities (Funzionalità), quindi scegliere Pubblicazione di testo.

Invio di un messaggio (AWS SDKs)

Per inviare un messaggio SMS utilizzando uno di questi AWS SDKs, utilizza l'operazione API in quell'SDK che corrisponde alla `Publish` richiesta nell'API Amazon SNS. Con questa richiesta, puoi inviare un messaggio SMS direttamente a un numero di telefono. Puoi anche utilizzare il parametro `MessageAttributes` per impostare i valori per i seguenti nomi attributo:

AWS.SNS.SMS.SenderID

Un ID personalizzato che contiene da 3 a 11 caratteri alfanumerici o caratteri trattino (-), tra cui almeno una lettera e nessuno spazio. L'ID mittente viene visualizzato come mittente del messaggio sul dispositivo ricevente. Ad esempio, puoi utilizzare il tuo marchio commerciale per aiutare a rendere più facilmente riconoscibile l'origine del messaggio.

Il supporto per 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 mittente IDs, consulta [Paesi e aree geografiche supportati per la messaggistica SMS 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 regioni che richiedono un ID mittente in forma alfabetica, il messaggio visualizza NOTICE (PREAVVISO) come ID mittente.

Questo attributo a livello di messaggio sovrascrive l'attributo a livello di account `DefaultSenderId`, che può essere impostato tramite la richiesta `SetSMSAttributes`.

AWS.MM.SMS.OriginationNumber

Una stringa personalizzata di 5-14 numeri, che può includere un segno + iniziale opzionale (+). Questa stringa di numeri viene visualizzata come numero di telefono del mittente sul dispositivo ricevente. La stringa deve corrispondere a un numero di origine configurato nel tuo AWS account per il paese di destinazione. Il numero di origine può essere un numero da 10 DLC, un numero verde, un codice lungo person-to-person (P2P) o un codice breve. Per ulteriori informazioni, consulta Numeri di [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 account. AWS

AWS.SNS.SMS.MaxPrice

L'importo massimo in dollari che sei disposto a spendere per inviare il messaggio SMS. Amazon SNS non invierà il messaggio nel caso in cui il costo dovesse superare il prezzo massimo.

Questo attributo non ha effetto se i costi month-to-date degli SMS hanno già superato la quota impostata per l'attributo. `MonthlySpendLimit` È possibile impostare `MonthlySpendLimit` utilizzando l'attributo `SetSMSAttributes`.

Quando invii il messaggio a un argomento Amazon SNS, a ogni messaggio inviato viene applicato il prezzo massimo a ognuno dei numeri di telefono sottoscritti a un dato argomento.

AWS.SNS.SMS.SMSType

Tipo di messaggio che intendi inviare:

- **Promotional** (impostazione predefinita) - Messaggi non critici, come i messaggi di marketing.
- **Transactional** – Messaggi critici che supportano le transazioni dei clienti, come i passcode monouso per l'autenticazione a più fattori.

Questo attributo a livello di messaggio sovrascrive l'attributo a livello di account `DefaultSMSType`, che può essere impostato tramite la richiesta `SetSMSAttributes`.

AWS.MM.SMS.EntityId

Questo attributo è necessario solo per l'invio di messaggi SMS ai destinatari in India.

ID entità o ID entità principale (PE) per l'invio di messaggi SMS ai destinatari in India. Questo ID è una stringa univoca di 1-50 caratteri fornita dalla Telecom Regulatory Authority of India (TRAI) per identificare l'entità registrata presso il TRAI.

AWS.MM.SMS.TemplateId

Questo attributo è necessario solo per l'invio di messaggi SMS ai destinatari in India.

Questo è il modello per inviare messaggi SMS ai destinatari in India. Questo ID è una stringa univoca fornita da TRAI di 1-50 caratteri che identificherà il modello registrato con il TRAI. L'ID modello deve essere associato all'ID mittente specificato per il messaggio.

Invio di un messaggio

Negli esempi di codice seguenti viene illustrato come pubblicare messaggi SMS utilizzando Amazon SNS.

.NET

SDK per .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](#).

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;
```

```
    /// <summary>
    /// Initializes a new instance of the <see cref="SNSMessage"/> class.
    /// Constructs a new SNSMessage object initializing the Amazon Simple
    /// Notification Service (Amazon SNS) client using the supplied
    /// Region endpoint.
    /// </summary>
    /// <param name="regionEndpoint">The Amazon Region endpoint to use in
    /// sending test messages with this object.</param>
    public SNSMessage(RegionEndpoint regionEndpoint)
    {
        snsClient = new
AmazonSimpleNotificationServiceClient(regionEndpoint);
    }

    /// <summary>
    /// Sends the SMS message passed in the text parameter to the phone
number
    /// in phoneNum.
    /// </summary>
    /// <param name="phoneNum">The ten-digit phone number to which the text
    /// message will be sent.</param>
    /// <param name="text">The text of the message to send.</param>
    /// <returns>Async task.</returns>
    public async Task SendTextMessageAsync(string phoneNum, string text)
    {
        if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
        {
            return;
        }

        // Now actually send the message.
        var request = new PublishRequest
        {
            Message = text,
            PhoneNumber = phoneNum,
        };

        try
        {
            var response = await snsClient.PublishAsync(request);
        }
        catch (Exception ex)
        {

```

```
        Console.WriteLine($"Error sending message: {ex}");  
    }  
}  
}
```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**  
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an  
 * SMS text message to a phone number.  
 * Note: This requires additional AWS configuration prior to running example.  
 *  
 * NOTE: When you start using Amazon SNS to send SMS messages, your AWS account  
 * is in the SMS sandbox and you can only  
 * use verified destination phone numbers. See https://docs.aws.amazon.com/sns/  
 * latest/dg/sns-sms-sandbox.html.  
 * NOTE: If destination is in the US, you also have an additional restriction  
 * that you have use a dedicated  
 * origination ID (phone number). You can request an origination number using  
 * Amazon Pinpoint for a fee.  
 * See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-  
 * origination-numbers-with-amazon-sns/  
 * for more information.  
 *  
 * <phone_number_value> input parameter uses E.164 format.  
 * For example, in United States, this input value should be of the form:  
 * +12223334444
```



```

*/

//! Send an SMS text message to a phone number.
/*!
\param message: The message to publish.
\param phoneNumber: The phone number of the recipient in E.164 format.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

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

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

    return outcome.IsSuccess();
}

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is
sent (for example, +1XXX5550100).\s
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String message = args[0];
String phoneNumber = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();
pubTextSMS(snsClient, message, phoneNumber);
snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

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

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Kotlin.

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 * SNS.
 *
 * This code expects that you have AWS credentials set up per:
```

```
* https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
*/

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

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

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

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class SnsWrapper:
```

```

"""Encapsulates Amazon SNS topic and subscription functions."""

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

def publish_text_message(self, phone_number, message):
    """
    Publishes a text message directly to a phone number without need for a
    subscription.

    :param phone_number: The phone number that receives the message. This
    must be
                           in E.164 format. For example, a United States phone
                           number might be +12065550101.
    :param message: The message to send.
    :return: The ID of the message.
    """
    try:
        response = self.sns_resource.meta.client.publish(
            PhoneNumber=phone_number, Message=message
        )
        message_id = response["MessageId"]
        logger.info("Published message to %s.", phone_number)
    except ClientError:
        logger.exception("Couldn't publish message to %s.", phone_number)
        raise
    else:
        return message_id

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API SDK for Python (Boto3)AWS .

SAP ABAP

SDK per 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](#).

```
" iv_phone_number = '+12065550101' - Phone number in E.164 format
TRY.
    oo_result = lo_sns->publish(                " oo_result is returned for
testing purposes. "
    iv_phonenumber = iv_phone_number
    iv_message = iv_message ).
    MESSAGE 'Message published to phone number.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Phone number does not exist.' TYPE 'E'.
ENDTRY.
```

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

Impostazione delle preferenze di messaggistica SMS in Amazon SNS

Utilizza Amazon SNS per specificare le preferenze per i messaggi SMS. Ad esempio, puoi specificare se ottimizzare le consegne per costo o affidabilità, il limite di spesa mensile, la registrazione delle consegne e se eseguire la sottoscrizione a report di utilizzo di SMS giornalieri.

Queste preferenze diventano effettive per ogni SMS che invii dal tuo account, ma alcune possono essere sovrascritte all'invio di un singolo messaggio. Per ulteriori informazioni, consulta [Pubblicazione di messaggi SMS su un telefono cellulare tramite Amazon SNS](#).

Impostazione delle preferenze per i messaggi SMS utilizzando il Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Scegliere una [regione che supporti la messaggistica SMS](#).

3. Nel pannello di navigazione, scegli Mobile e poi Messaggi di testo (SMS).
4. Nella pagina Mobile text messaging (SMS) (Messaggi di testo mobili (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).
5. Nella pagina Modifica le preferenze per i messaggi di testo, nella sezione Dettagli, effettuare queste operazioni:
 - a. Per Default message type (Tipo di messaggio predefinito), scegliere una di queste opzioni:
 - Promozionale: messaggi non critici (ad esempio marketing). Amazon SNS ottimizza la consegna dei messaggi per generare il costo più basso.
 - Transazionale: messaggi critici che supportano le transazioni dei clienti, come le password monouso per l'autenticazione a più fattori. Amazon SNS ottimizza la consegna dei messaggi per ottenere la migliore affidabilità.

Per informazioni sulle tariffe relative a messaggi promozionali e transazionali, consulta la pagina relativa alle [tariffe SMS globali](#).

- b. In Account spend limit (Limite di spesa per account), digitare l'importo massimo in USD da spendere in messaggi SMS ogni mese di calendario.

 Important

- Per impostazione predefinita, la quota di spesa è impostata a 1,00 USD. Se si desidera aumentare la quota di servizio, [inviare una richiesta](#).
- Se l'importo impostato nella console supera la quota di servizio, Amazon SNS interrompe la pubblicazione di messaggi SMS.
- Poiché Amazon SNS è un sistema distribuito, blocca l'invio di SMS alcuni minuti dopo il superamento della quota di spesa. Se continui a inviare SMS durante tale intervallo, potresti incorrere in costi che superano la tua quota di spesa.

6. (Facoltativo) Per Default sender ID (ID mittente predefinito), immettere un ID personalizzato, ad esempio il marchio aziendale, visualizzato come mittente sul dispositivo di ricezione.

 Note

Il supporto per 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 Amazon S3 deve concedere l'accesso in scrittura ad Amazon SNS.

8. Scegli Save changes (Salva modifiche).

Impostazione delle preferenze (AWS SDKs)

Per impostare le tue preferenze SMS utilizzando uno di questi AWS SDKs, utilizza l'azione in quell'SDK che corrisponde alla `SetSMSAttributes` richiesta nell'API Amazon SNS. Con questa richiesta, assegna valori a differenti attributi SMS, come la quota di spesa mensile e il tipo di SMS predefinito (promozionale o transazionale). Per tutti gli attributi SMS, consulta [Set SMSAttributes](#) in the Amazon Simple Notification Service API Reference.

Gli esempi di codice seguenti mostrano come utilizzare `SetSMSAttributes`.

C++

SDK per 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'attributo predefinito `SMSType`.

```
#!/ 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 i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK per C++ API Reference.

CLI

AWS CLI

Impostazione degli attributi dei messaggi SMS

Nell'esempio `set-sms-attributes` seguente l'ID mittente predefinito per i messaggi SMS viene impostato su `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderId=MyName
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS CLI Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
String> attributes) {
        try {
```

```

        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result =
snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ".
Status was "
            + result.sdkHttpResponse().statusCode());

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

```

- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

```

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

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

```

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

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

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing
        // messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
$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 per PHP](#).
- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK per PHP API Reference.

Impostazione delle preferenze di messaggistica SMS per la consegna specifica del Paese

Puoi gestire e controllare il traffico SMS inviando messaggi solo a paesi di destinazione specifici. Ciò garantisce che i messaggi vengano inviati solo nei paesi approvati, evitando addebiti indesiderati per SMS. 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 Regole nazionali relative agli SMS, 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 Amazon SNS, [Amazon Cognito](#) e la chiamata API Amazon Pinpoint. [SendMessage](#)
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 [console Amazon SNS](#).
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 di telefono e degli abbonamenti Amazon SNS

Amazon SNS offre diverse opzioni per gestire chi riceve messaggi SMS dal tuo account. Con frequenza limitata, puoi inserire nell'elenco di invio numeri di telefono di destinatari che hanno scelto di non ricevere i messaggi SMS dal tuo account. Per interrompere l'invio di messaggi a sottoscrizioni SMS, puoi rimuovere le sottoscrizioni o gli argomenti in cui effettui la pubblicazioni destinate alle sottoscrizioni.

Richiesta di non ricevere i messaggi SMS

Laddove richiesto dalle leggi e dai regolamenti locali (come Stati Uniti e Canada), i destinatari degli SMS 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 allo stesso [numero di origine](#) utilizzato da Amazon SNS per recapitare il messaggio. Dopo la rinuncia, il destinatario non riceverà più i messaggi SMS recapitati dal destinatario, a Account AWS meno che tu non inserisca il numero di telefono.

Se il numero di telefono è abbonato a un argomento di Amazon SNS, la disattivazione non rimuove l'abbonamento, ma i messaggi SMS non verranno recapitati a quell'abbonamento a meno che tu non fornisca il numero di telefono.

Gestione dei numeri di telefono e degli abbonamenti tramite la console Amazon SNS

Per specificare quali numeri di telefono ricevono messaggi dal tuo account, puoi usare la console Amazon SNS.

Inserimento di un numero di telefono che è stato disattivato nella console Amazon SNS

Puoi visualizzare quali numeri di telefono sono stati disattivati dalla ricezione di messaggi SMS 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 [console Amazon SNS](#).
2. Nel menu della console, imposta lo strumento di selezione della regione su una [regione che supporti la messaggistica SMS](#).
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Nella pagina Messaggi di testo mobili (SMS), nella sezione Numeri di telefono annullati, 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 i messaggi SMS che gli invierai.

Eliminazione di un abbonamento SMS dalla console Amazon SNS

Elimina una sottoscrizione SMS per interrompere l'invio di messaggi SMS al numero specificato quando effettui pubblicazioni nei tuoi argomenti.

1. Nel riquadro di navigazione, scegli Sottoscrizioni.
2. Seleziona le caselle di controllo delle sottoscrizioni da eliminare. Scegli Actions (Operazioni), quindi Delete Subscriptions (Elimina sottoscrizioni).
3. Nella finestra Delete (Elimina) scegli Delete (Elimina). Amazon SNS elimina la sottoscrizione visualizzandone poi la conferma.

Eliminazione di un argomento 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 visualizzandone poi la conferma.

Gestione dei numeri di telefono e degli abbonamenti tramite l'SDK AWS

Puoi utilizzarlo AWS SDKs per effettuare richieste programmatiche ad Amazon SNS e gestire quali numeri di telefono possono ricevere messaggi SMS dal tuo account.

Per utilizzare un AWS SDK, devi configurarlo con le tue credenziali. Per ulteriori informazioni, consulta [File di configurazione e credenziali condivisi](#) nella Guida di riferimento agli strumenti AWS SDKs e agli strumenti.

Visualizzazione di tutti i numeri di telefono esclusi tramite l'SDK AWS

Per visualizzare tutti i numeri di telefono esclusi, invia una `ListPhoneNumbersOptedOut` richiesta con l'API Amazon SNS.

Gli esempi di codice seguenti mostrano come utilizzare `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Elencare le opzioni di opt-out dei messaggi SMS

Nell'esempio `list-phone-numbers-opted-out` seguente sono elencati i numeri di telefono per i quali sono abilitate le opzioni di opt-out per la ricezione di SMS.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
  "phoneNumbers": [
    "+15555550100"
  ]
}
```

- Per i dettagli sull'API, consulta Command [ListPhoneNumbersOptedOut](#)Reference AWS CLI .

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
                ListPhoneNumbersOptedOutRequest.builder().build();
```

```

        ListPhoneNumbersOptedOutResponse result =
snsClient.listPhoneNumbersOptedOut(request);
        System.out.println("Status is " +
result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
            + result.phoneNumbers());

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

```

- Per i dettagli sull'API, consulta la [ListPhoneNumbersOptedOut](#) sezione AWS SDK for Java 2.x API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

require 'vendor/autoload.php';

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

/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 * from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

```

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

try {
    $result = $SnSClient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per i dettagli sull'API, consulta la [ListPhoneNumbersOptedOut](#) sezione AWS SDK per PHP API Reference.

Verifica se un numero di telefono è stato disattivato utilizzando l'SDK AWS

Per verificare se un numero di telefono è stato disattivato, invia una `CheckIfPhoneNumberIsOptedOut` richiesta con l'API Amazon SNS.

Gli esempi di codice seguenti mostrano come utilizzare `CheckIfPhoneNumberIsOptedOut`.

.NET

SDK per .NET

 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](#).

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

```
using Amazon.SimpleNotificationService.Model;

/// <summary>
/// This example shows how to use the Amazon Simple Notification Service
/// (Amazon SNS) to check whether a phone number has been opted out.
/// </summary>
public class IsPhoneNumOptedOut
{
    public static async Task Main()
    {
        string phoneNumber = "+15551112222";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await CheckIfOptedOutAsync(client, phoneNumber);
    }

    /// <summary>
    /// Checks to see if the supplied phone number has been opted out.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS Client object used
    /// to check if the phone number has been opted out.</param>
    /// <param name="phoneNumber">A string representing the phone number
    /// to check.</param>
    public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
    {
        var request = new CheckIfPhoneNumberIsOptedOutRequest
        {
            PhoneNumber = phoneNumber,
        };

        try
        {
            var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

            if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
            {
                string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
            }
        }
    }
}
```

```
        Console.WriteLine($"The phone number: {phoneNumber} is  
{optOutStatus}");  
    }  
    }  
    catch (AuthorizationErrorException ex)  
    {  
        Console.WriteLine($"{ex.Message}");  
    }  
    }  
}
```

- Per i dettagli sull'API, consulta la [CheckIfPhoneNumberIsOptedOut](#) sezione AWS SDK per .NET API Reference.

CLI

AWS CLI

Controllo delle impostazioni di opt-out dei messaggi SMS per un numero di telefono

L'`check-if-phone-number-is-opted-out` esempio seguente verifica se al numero di telefono specificato è stata disattivata la ricezione di messaggi SMS dall' AWS account corrente.

```
aws sns check-if-phone-number-is-opted-out \  
--phone-number +1555550100
```

Output:

```
{  
  "isOptedOut": false  
}
```

- Per i dettagli sull'API, vedere [CheckIfPhoneNumberIsOptedOut](#) in AWS CLI Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String phoneNumber = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    checkPhone(snsClient, phoneNumber);
    snsClient.close();
}

public static void checkPhone(SnsClient snsClient, String phoneNumber) {
    try {
        CheckIfPhoneNumberIsOptedOutRequest request =
        CheckIfPhoneNumberIsOptedOutRequest.builder()
            .phoneNumber(phoneNumber)
            .build();

        CheckIfPhoneNumberIsOptedOutResponse result =
        snsClient.checkIfPhoneNumberIsOptedOut(request);
        System.out.println(
            result.isOptedOut() + "Phone Number " + phoneNumber + " has
Opted Out of receiving sns messages." +
            "\n\nStatus was " +
            result.sdkHttpResponse().statusCode());

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

- Per i dettagli sull'API, consulta la [CheckIfPhoneNumberIsOptedOut](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
//      totalRetryDelay: 0
//    },
//    isOptedOut: false
//  }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [CheckIfPhoneNumberIsOptedOut](#) sezione AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 * messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

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

```
]);

$phone = '+1XXX5550100';

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

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per i dettagli sull'API, consulta la [CheckIfPhoneNumbersOptedOut](#) sezione AWS SDK per PHP API Reference.

Inserimento di un numero di telefono che è stato disattivato utilizzando l'API Amazon SNS

Per attivare un numero di telefono, invia una `OptInPhoneNumber` richiesta con l'API Amazon SNS.

Puoi inserire un numero di telefono solo una volta ogni 30 giorni.

Eliminazione di un abbonamento SMS tramite l'SDK AWS

Per eliminare una sottoscrizione SMS da un argomento Amazon SNS, recupera l'ARN della sottoscrizione inviando una richiesta `ListSubscriptions` con l'API di `&SNS;`, quindi passa l'ARN a una richiesta `Unsubscribe`.

Gli esempi di codice seguenti mostrano come utilizzare `Unsubscribe`.

.NET

SDK per .NET

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](#).

Annulla l'iscrizione a un argomento tramite un ARN di sottoscrizione.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

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

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

CLI

AWS CLI

Annullamento della sottoscrizione a un argomento

Nell'esempio unsubscribe seguente viene eliminata la sottoscrizione specificata a un argomento.

```
aws sns unsubscribe \
  --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Questo comando non produce alcun output.

- Per informazioni dettagliate sull'API, consulta [Unsubscribe](#) nella documentazione di riferimento dei comandi della AWS CLI .

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:    <subscriptionArn>

        Where:
            subscriptionArn - The ARN of the subscription to delete.
            "";

    if (args.length < 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String subscriptionArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    unSub(snsClient, subscriptionArn);
    snsClient.close();
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " +
request.subscriptionArn());

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

```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
  xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
```

```
//    extendedRequestId: undefined,  
//    cfId: undefined,  
//    attempts: 1,  
//    totalRetryDelay: 0  
//  }  
// }  
return response;  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per JavaScript .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun unSub(subscriptionArnVal: String) {  
    val request =  
        UnsubscribeRequest {  
            subscriptionArn = subscriptionArnVal  
        }  
  
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->  
        snsClient.unsubscribe(request)  
        println("Subscription was removed for ${request.subscriptionArn}")  
    }  
}
```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Kotlin.

PHP

SDK per PHP

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSClient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
                             subscription.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True  # Already unsubscribed
            else:
```

```
logger.error(f"Error unsubscribing: {error_code} - {e}")
raise
```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API SDK for Python (Boto3)AWS .

SAP ABAP

SDK per 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 informazioni dettagliate sull'API, consulta [Unsubscribe](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

_ = try await snsClient.unsubscribe(
    input: UnsubscribeInput(
        subscriptionArn: arn
    )
)

print("Unsubscribed.")
```

- Per informazioni dettagliate sull'API, consulta [Unsubscribe](#) nella documentazione di riferimento dell'API AWS SDK per Swift.

Eliminazione di un argomento utilizzando l'SDK AWS

Per eliminare un argomento e tutte le relative sottoscrizioni, recupera l'ARN dell'argomento inviando una richiesta `ListTopics` con l'API di Amazon SNS, quindi passa l'ARN alla richiesta `DeleteTopic`.

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

.NET

SDK per .NET

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un argomento in base all'ARN dell'argomento.

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

- Per i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```

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

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

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

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

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Eliminazione di un argomento SNS

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

```

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

```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteTopic AWS CLI Command Reference](#).

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// 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 i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

            Usage:      <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            "";

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    System.out.println("Deleting a topic with name: " + topicArn);
    deleteSNSTopic(snsClient, topicArn);
    snsClient.close();
}

public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
            .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("\n\nStatus was " +
            result.sdkHttpResponse().statusCode());

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

- Per i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

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

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

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

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

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

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

- Per i dettagli sull'API, consulta la [DeleteTopic](#) sezione AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class SnsWrapper:
```

```

"""Encapsulates Amazon SNS topic and subscription functions."""

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

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

```

```

class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

```



```

def delete_topic(self, topic_arn: str) -> bool:
    """
    Delete an SNS topic.

    :param topic_arn: The ARN of the topic to delete.
    :return: True if successful.
    :raises ClientError: If the topic deletion fails.
    """
    try:
        self.sns_client.delete_topic(TopicArn=topic_arn)

        logger.info(f"Deleted topic: {topic_arn}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'NotFound':
            logger.warning(f"Topic not found: {topic_arn}")
            return True # Already deleted
        else:
            logger.error(f"Error deleting topic: {error_code} - {e}")
            raise

```

- Per i dettagli sull'API, consulta [DeleteTopic AWS](#) SDK for Python (Boto3) API Reference.

SAP ABAP

SDK per 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 i dettagli sulle API, [DeleteTopic](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS  
  
let config = try await SNSClient.SNSClientConfiguration(region: region)  
let snsClient = SNSClient(config: config)  
  
_ = try await snsClient.deleteTopic(  
    input: DeleteTopicInput(topicArn: arn)  
)
```

- Per i dettagli sull'API, consulta la [DeleteTopic](#) guida di riferimento all'API AWS SDK for Swift.

Monitoraggio delle attività SMS di Amazon SNS

Mediante il monitoraggio della tua attività SMS, puoi tenere traccia di numeri di telefono di destinazione, consegne riuscite o non riuscite, motivi della mancate consegne, costi e altre informazioni. Amazon SNS ti aiuta riepilogando le statistiche nella console, inviando informazioni ad Amazon CloudWatch e inviando report giornalieri sull'utilizzo di SMS a un bucket Amazon S3 da te specificato.

Visualizzazione delle statistiche di consegna degli SMS di Amazon SNS

Puoi utilizzare la console Amazon SNS per visualizzare le statistiche relative alla consegna dei tuoi SMS più recenti

1. Accedi alla [console Amazon SNS](#).
2. Nel menu della console, imposta lo strumento di selezione della regione su una [regione che supporti la messaggistica SMS](#).
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Sulla pagina Text messaging (SMS) (Preferenze messaggistica (SMS)) nella sezione Account stats (Statistiche account), visualizza i grafici delle consegne degli SMS promozionali e di vendita. Ogni tabella mostra i seguenti dati relativi ai 15 giorni precedenti:
 - Tasso di consegna (percentuale di successo)
 - Inviati (numero di tentativi di recapito)
 - Falliti (numero di mancate consegne)

Su questa pagina, puoi anche selezionare il pulsante Usage (Utilizzo) per andare al bucket Amazon S3 dove vengono salvati i report di utilizzo giornalieri. Per ulteriori informazioni, consulta [Iscrizione ai report giornalieri sull'utilizzo degli SMS di Amazon SNS](#).

Monitoraggio della consegna degli SMS di Amazon SNS con CloudWatch parametri e log di Amazon

Puoi utilizzare Amazon CloudWatch e Amazon CloudWatch Logs per monitorare le consegne dei tuoi messaggi SMS.

Visualizzazione dei CloudWatch parametri di Amazon

Amazon SNS raccoglie automaticamente i parametri relativi alle consegne dei messaggi SMS e li invia ad Amazon. CloudWatch Puoi utilizzarlo CloudWatch per monitorare queste metriche e creare allarmi per avvisarti quando una metrica supera una soglia. Ad esempio, puoi monitorare le CloudWatch metriche per conoscere la tariffa di consegna degli SMS e i costi degli SMS. month-to-date

Per informazioni sul monitoraggio delle CloudWatch metriche, sull'impostazione degli CloudWatch allarmi e sui tipi di metriche disponibili, consulta. [Monitoraggio degli argomenti di Amazon SNS tramite CloudWatch](#)

Visualizzazione dei registri CloudWatch

Puoi raccogliere informazioni sulle consegne di messaggi SMS riuscite e non riuscite abilitando Amazon SNS alla scrittura su Amazon Logs. CloudWatch Per ogni SMS inviato, Amazon SNS creerà un log con il costo del messaggio, lo stato di consegna e la motivazione in caso di fallimento, il tempo di sosta e altre informazioni.

Per abilitare e visualizzare i CloudWatch log per i tuoi messaggi SMS

1. Accedi alla [console Amazon SNS](#).
2. Nel menu della console, imposta lo strumento di selezione della regione su una [regione che supporti la messaggistica SMS](#).
3. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
4. Nella pagina Mobile text messaging (SMS) (Messaggi di testo mobili (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).
5. Nella pagina successiva, espandere la sezione Delivery status logging (Registrazione dello stato di consegna).
6. Per la frequenza di campionamento Success, specifica la percentuale di consegne SMS riuscite per le quali Amazon SNS scriverà i log in CloudWatch Logs. Esempio:
 - Per generare dei log unicamente per le consegne non riuscite, imposta questo valore su 0.
 - Per generare dei log per il 10% delle consegne riuscite, imposta 10.

Se non specifichi una percentuale, Amazon SNS genera dei log per tutte le consegne riuscite.

7. Per fornire le autorizzazioni necessarie, eseguire una delle seguenti operazioni:
 - Per creare un nuovo ruolo del servizio, scegli Creazione di un nuovo ruolo del servizio e poi Creazione di nuovi ruoli. Nella pagina successiva, scegli Abilita per consentire ad Amazon SNS di accedere in scrittura alle risorse del tuo account.
 - Per utilizzare un ruolo del servizio esistente, scegli Utilizzare il ruolo del servizio esistente e quindi incollare il nome ARN nella casella Ruolo IAM per consegne riuscite e non riuscite.

Il ruolo di servizio specificato deve consentire l'accesso in scrittura alle risorse dell'account. Per ulteriori informazioni sulla creazione di ruoli IAM, consulta [Creating a role for an AWS service](#) nella IAM User Guide.

8. Scegli Save changes (Salva modifiche).

9. Nella pagina Messaggi di testo per dispositivi mobili (SMS) vai alla pagina Registri dello stato di per visualizzare tutti i registri disponibili.

Note

A seconda del corriere del numero di telefono di destinazione, possono essere necessarie fino a 72 ore affinché i log delle consegne vengano visualizzati nella console Amazon SNS.

Log di esempio per gli SMS inviati con successo

Il log dello stato di consegna per un SMS inviato con successo sarà simile al seguente esempio:

```
{
  "notification": {
    "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
    "timestamp": "2016-06-28 00:40:34.558"
  },
  "delivery": {
    "phoneCarrier": "My Phone Carrier",
    "mnc": 270,
    "numberOfMessageParts": 1,
    "destination": "+1XXX5550100",
    "priceInUSD": 0.00645,
    "smsType": "Transactional",
    "mcc": 310,
    "providerResponse": "Message has been accepted by phone carrier",
    "dwellTimeMs": 599,
    "dwellTimeMsUntilDeviceAck": 1344
  },
  "status": "SUCCESS"
}
```

Log di esempio per gli SMS falliti

Il log dello stato di consegna per un SMS fallito sarà simile al seguente esempio:

```
{
  "notification": {
    "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
    "timestamp": "2016-06-28 00:40:34.559"
  },
  "status": "FAILURE"
}
```

```
{,
  "delivery": {
    "mnc": 0,
    "numberOfMessageParts": 1,
    "destination": "+1XXX5550100",
    "priceInUSD": 0.00645,
    "smsType": "Transactional",
    "mcc": 0,
    "providerResponse": "Unknown error attempting to reach phone",
    "dwellTimeMs": 1420,
    "dwellTimeMsUntilDeviceAck": 1692
  },
  "status": "FAILURE"
}
```

Ragioni per un SMS fallito

La ragione per un SMS fallito viene fornita dall'attributo `providerResponse`. L'invio di un messaggio SMS potrebbe fallire per i seguenti motivi:

- Bloccato come spam dall'operatore telefonico
- La destinazione è in un elenco bloccato
- Il numero di telefono non è valido
- Il testo del messaggio non è valido
- L'operatore telefonico ha bloccato il messaggio
- L'operatore telefonico non è raggiungibile/disponibile
- Il telefono ha bloccato l'SMS
- Il telefono è in un elenco bloccato
- Il telefono non è raggiungibile/disponibile
- Il numero di telefono è stato escluso
- La consegna supera il costo massimo
- Errore sconosciuto nel tentativo di raggiungere il telefono

Iscrizione ai report giornalieri sull'utilizzo degli SMS di Amazon SNS

Puoi monitorare le consegne di SMS effettuando la sottoscrizione a report di utilizzo di SMS giornalieri a partire da Amazon SNS. Per ogni giorno in cui invii almeno un messaggio SMS, Amazon

SNS fornisce un report di utilizzo in un file CSV per il bucket Amazon S3 specificato. Sono necessarie 24 ore prima che il rapporto sull'utilizzo degli SMS sia disponibile nel bucket Amazon S3.

Informazioni contenute nei report di utilizzo giornalieri

Il report di utilizzo include le seguenti informazioni per ogni SMS inviato dal tuo account.

Tenere presente che il report non include messaggi che vengono inviati ai destinatari che hanno scelto di non ricevere i messaggi.

- Tempo di pubblicazione per messaggio (in UTC)
- ID messaggio
- Numero di telefono di destinazione
- Tipo di messaggio
- Stato della consegna
- Costo del messaggio (in USD)
- Numero di parte (un messaggio viene suddiviso in più parti se è troppo lungo per un unico messaggio)
- Numero totale di parti

Note

Se Amazon SNS non ha ricevuto il numero di parte, impostiamo il suo valore su zero.

Sottoscrizione ai report di utilizzo giornalieri

Per eseguire la sottoscrizione ai report di utilizzo giornalieri, devi creare un bucket Amazon S3 con le autorizzazioni appropriate.

Creazione di un bucket Amazon S3 per i report di utilizzo giornalieri

1. Da chi Account AWS invia messaggi SMS, 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.

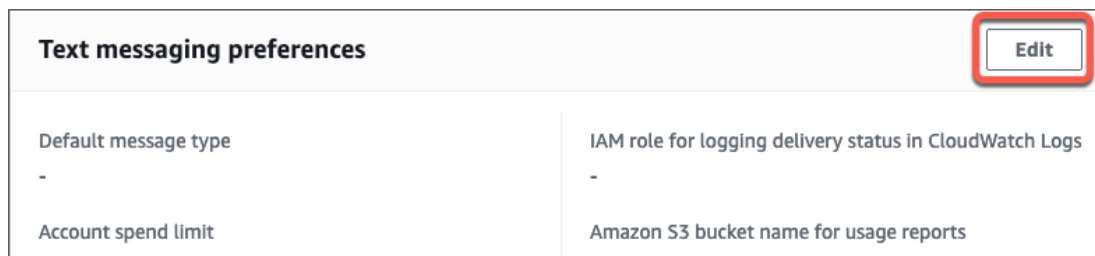
4. Scegli Create (Crea).
5. Nella tabella All Buckets (Tutti i bucket), seleziona il bucket.
6. Nella sezione Permissions (Autorizzazioni), scegliere Bucket policy (Policy bucket).
7. Nella finestra Bucket Policy Editor (Editor policy bucket), specifica una policy che autorizzi il principale del servizio Amazon SNS a 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 nello Step 3.

8. Scegli Save (Salva).

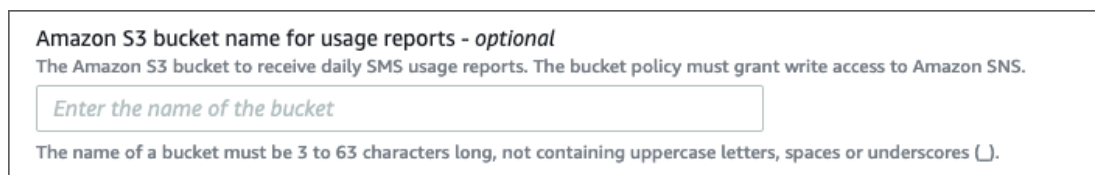
Sottoscrizione ai report di utilizzo giornalieri

1. Accedi alla console [Amazon SNS](#).
2. Nel riquadro di navigazione, selezionare Text messaging (SMS) (Messaggi di testo (SMS)).
3. Nella pagina Text messaging (SMS) (Messaggi di testo (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).



The screenshot shows the 'Text messaging preferences' section of the Amazon SNS console. It contains four settings: 'Default message type' (set to '-'), 'IAM role for logging delivery status in CloudWatch Logs' (set to '-'), 'Account spend limit', and 'Amazon S3 bucket name for usage reports'. An 'Edit' button is located in the top right corner and is highlighted with a red rectangular box.

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).



The screenshot shows the 'Amazon S3 bucket name for usage reports - optional' section. It includes a text input field with the placeholder text 'Enter the name of the bucket'. Below the input field, there is a note: 'The name of a bucket must be 3 to 63 characters long, not containing uppercase letters, spaces or underscores (_).'.

5. Scegli Save changes (Salva modifiche).

Esempio di policy di bucket

La policy seguente consente al principale del servizio Amazon SNS di eseguire le operazioni `s3:PutObject`, `s3:GetBucketLocation` e `s3:ListBucket`.

AWS fornisce strumenti per tutti i servizi con i responsabili dei servizi 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 si conosce l'ARN della risorsa, utilizzare `aws:SourceAccount`.

Utilizzare l'esempio seguente che include la protezione "confused deputy" quando crei un bucket Amazon S3 per ricevere report di utilizzo di SMS giornalieri da Amazon SNS.

JSON

```
{
  "Version": "2012-10-17",
  "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:us-west-1:123456789012:*"
        }
      }
    },
    {
      "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:us-west-1:123456789012:*"
        }
    }
},
{
    "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:us-west-1:123456789012:*"
        }
    }
}
]
}

```

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 di utilizzo in un bucket Amazon S3 di Account AWS proprietà di un altro, vedi [Come posso copiare oggetti Amazon S3](#) da un altro? Account AWS.

Esempio di report di utilizzo giornaliero

Dopo la sottoscrizione ai report di utilizzo giornalieri, Amazon SNS inserisce un file CSV con i dati sull'utilizzo nella seguente posizione:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Ogni file può contenere fino a 50.000 record. Se i record di un giorno superano questa quota, Amazon SNS aggiunge più file. Di seguito viene riportato un esempio di report:

```
PublishTimeUTC,MessageId,DestinationPhoneNumber,MessageType,DeliveryStatus,PriceInUSD,PartNumber
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 SMS di Amazon SNS

Important

La Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS è integrato [AWS End User Messaging SMS](#) per la consegna di messaggi SMS. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i messaggi SMS di Amazon SNS.

Alcune opzioni SMS con Amazon SNS non sono disponibili per il tuo AWS account finché non ti contatti. Supporto Apri un caso nel [Center Supporto AWS](#) per fare una delle richieste seguenti:

- Un aumento della soglia di spesa mensile per gli SMS

Per impostazione predefinita, la soglia di spesa mensile è di 1 USD. La soglia di spesa determina il volume di messaggi che è possibile inviare con Amazon SNS. Puoi richiedere una soglia che soddisfi il volume mensile previsto di messaggi SMS per il tuo caso d'uso.

- Una mossa dalla [sandbox SMS](#) in modo da poter inviare messaggi SMS senza restrizioni. Per ulteriori informazioni, consulta [Uscire dalla sandbox SMS di Amazon SNS](#).
- 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 messaggistica SMS AWS End User Messaging SMS](#) nella Guida per l'AWS End User Messaging SMS utente.

Richiesta di aumenti della quota di spesa mensile di Amazon SNS SMS

Amazon SNS fornisce quote di spesa che ti aiutano a gestire il costo massimo mensile sostenuto per l'invio di SMS utilizzando il tuo account. La quota di spesa limita i rischi in caso di attacchi dannosi e impedisce all'applicazione a monte di inviare più messaggi del previsto. Puoi configurare Amazon SNS in modo tale da interrompere la pubblicazione di messaggi SMS qualora il sistema determini che l'invio di un messaggio SMS comporta un costo che supera la quota di spesa per il mese corrente.

Per garantire che la tua attività rimanga operativa, ti consigliamo di richiedere una quota di spesa sufficientemente elevata per supportare i carichi di lavoro di produzione. Per ulteriori informazioni, consulta [Fase 1: apertura di un caso per SMS Amazon SNS](#). Una volta ricevuta la quota, puoi gestire il rischio applicando la quota completa o un valore inferiore, come descritto nella sezione [Fase 2: aggiornamento delle impostazioni relative agli SMS](#). Applicando un valore più basso, puoi controllare la spesa mensile con la possibilità di aumentarla in base alle esigenze.

Important

Poiché è un sistema distribuito, Amazon SNS blocca l'invio di SMS alcuni minuti dopo il superamento della quota di spesa. Se continui a inviare SMS durante tale periodo, potresti incorrere in costi che superano la tua quota di spesa.

La quota di spesa è impostata su 1 USD al mese per tutti i nuovi account. Questa quota è pensata per consentirti di testare le funzionalità di invio di messaggi di Amazon SNS. Per richiedere un aumento della quota di spesa via SMS per il tuo account, apri un caso di aumento della quota nel AWS Support Center.

Argomenti

- [Fase 1: apertura di un caso per SMS Amazon SNS](#)
- [Fase 2: aggiornamento delle impostazioni relative agli SMS nella console Amazon SNS](#)

Fase 1: apertura di un caso per SMS Amazon SNS

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, Supporto 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 Console di gestione AWS 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 Messaggi di testo SNS.
 - (Facoltativo) In Fornisci un link al sito o all'app che invierà messaggi SMS, fornisci informazioni relative al sito Web, all'applicazione o al servizio che invierà messaggi SMS.
 - (Facoltativo) In Tipo di messaggi di cui è previsto l'invio, scegli il tipo di messaggio che intendi inviare con codici lunghi:
 - One-Time Password (Password una tantum) - Messaggi che forniscono password che i clienti utilizzano per l'autenticazione a un sito o un'applicazione.
 - Promotional (Promozionale) - Messaggi non critici che promuovono l'azienda o un servizio, ad esempio offerte speciali o annunci.
 - Transactional (Transazionale) - Messaggi informativi importanti che supportano le transazioni con i clienti, come conferme d'ordine o avvisi dell'account. I messaggi transazionali non devono contenere contenuti promozionali o di marketing.
 - (Facoltativo) 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, immetti l'importo massimo (in USD) che puoi spendere in messaggi SMS per ogni mese di calendario.
7. In Case description (Descrizione caso), per Use case description (Descrizione del caso d'uso), specificare i dettagli seguenti:
- Il sito Web o l'app dell'azienda o del servizio che invia messaggi SMS
 - Servizio fornito dal sito Web o dall'app e contributo dei messaggi SMS a tale servizio
 - In che modo gli utenti si registrano per ricevere volontariamente i messaggi SMS sul sito Web, sull'app o altra posizione.

Se la quota di spesa richiesta, ovvero il valore specificato per New quota value (Nuovo valore quota), è superiore a 10.000 USD, fornire i seguenti dettagli aggiuntivi per ciascun paese a cui si inviano messaggi:

- Se si utilizza un ID mittente o un codice breve. Se si utilizza un ID mittente, fornire:
 - L'ID del mittente.

- Se l'ID del mittente è registrato con operatori wireless nel paese.
 - Il massimo previsto transactions-per-second (TPS) per la tua messaggistica.
 - La dimensione media dei messaggi.
 - Il modello dei messaggi che vengono inviati al paese specifico.
 - (Facoltativo) eventuali esigenze di codifica dei caratteri.
8. (Facoltativo) Se desideri inviare ulteriori richieste, scegli Aggiungi un'altra richiesta. Se si includono più richieste, specificare le informazioni necessarie per ciascuna. Per le informazioni richieste, consulta le altre sezioni all'interno di [Richiesta di supporto per la messaggistica SMS di Amazon SNS](#).
 9. In Opzioni di contatto, per Lingua di contatto preferita, scegli la lingua in cui desideri ricevere le comunicazioni per questo caso.
 10. Al termine, scegli Submit (Invia).

Il Supporto 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

Fase 2: aggiornamento delle impostazioni relative agli SMS nella console Amazon SNS

Dopo aver ricevuto la notifica relativa all'aumento della quota di spesa mensile, devi modificare la quota di spesa per l'account nella console Amazon SNS.

 Important

È necessario completare i seguenti passaggi o il limite di spesa per gli SMS non verrà aumentato.

Per modificare la quota di spesa nella console

1. Accedi alla [console Amazon SNS](#).

2. Apri il menu di navigazione a sinistra, espandi Dispositivi mobili, quindi scegli Messaggi di testo (SMS).
3. Nella pagina Mobile text messaging (SMS) (Messaggi di testo mobili (SMS)), nella sezione Text messaging preferences (Preferenze per i messaggi di testo), scegliere Edit (Modifica).
4. Nella pagina Modifica le preferenze per i messaggi di testo, nella sezione Dettagli, inserisci il nuovo limite di spesa SMS nel campo Limite di spesa per account.

 Note

Potrebbe essere visualizzato un avviso che indica che il valore immesso è maggiore del limite di spesa predefinito. È possibile ignorare questo avviso.

5. Scegli Save changes (Salva modifiche).

 Note

Se ricevi un errore di "Parametro non valido", controlla il contatto da Support AWS e conferma di aver inserito il nuovo limite di spesa SMS corretto. Se il problema persiste, apri una richiesta nel AWS Support Center.

Quando crei il tuo caso nel Supporto Centro, assicurati di includere tutte le informazioni richieste per il tipo di richiesta che stai inviando. Altrimenti, Supporto devi contattarti per ottenere queste informazioni prima di procedere. Inviando un caso dettagliato, aumenti le probabilità che venga soddisfatto senza ritardi. Per i dettagli che sono necessari per tipi specifici di richieste SMS, consulta i seguenti argomenti.

Per ulteriori informazioni sul mittente IDs, consulta la seguente documentazione nella Guida per l'AWS End User Messaging SMS utente:

AWS End User Messaging SMS Argomento	Descrizione
Richiesta di aumento della quota di spesa	La tua quota di spesa determina la quantità di denaro che puoi spendere per l'invio di messaggi SMS AWS End User Messaging SMS ogni mese.

AWS End User Messaging SMS Argomento	Descrizione
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 mittente IDs , puoi richiedere un ID mittente creando una nuova richiesta nel Centro. Supporto

Le migliori pratiche per la messaggistica SMS di Amazon SNS

Important

La Amazon SNS SMS Developer Guide è stata aggiornata. Amazon SNS è integrato [AWS End User Messaging SMS](#) per la consegna di messaggi SMS. Questa guida contiene le informazioni più recenti su come creare, configurare e gestire i messaggi SMS di Amazon SNS.

Gli utenti di telefonia mobile tendono ad avere scarsa tolleranza nei confronti dei messaggi SMS non sollecitati. Le campagne SMS non sollecitate otterranno quasi sempre tassi di risposta bassi e, di conseguenza, un ritorno sull'investimento ridotto.

Gli operatori di telefonia mobile, inoltre, controllano continuamente i mittenti di SMS in blocco e limitano o bloccano i messaggi provenienti da numeri che sono stati identificati come mittenti di messaggi non sollecitati.

L'invio di contenuti non sollecitati costituisce anche una violazione dell'[Acceptable Use Policy \(policy di utilizzo accettabile\) di AWS](#). Il team di Amazon SNS controlla regolarmente le campagne SMS e potrebbe limitare o bloccare la possibilità di inviare messaggi se risulta che vengono inviati messaggi indesiderati.

In molti paesi, regioni e giurisdizioni, infine, sono previste sanzioni severe per l'invio di messaggi SMS non sollecitati. Ad esempio, negli Stati Uniti, il Telephone Consumer Protection Act (TCPA) stabilisce che i consumatori hanno diritto a \$500-\$1.500 in danni (pagati dal mittente) per ogni messaggio non richiesto che ricevono.

Questa sezione illustra diverse best practice che potrebbero rivelarsi utili per migliorare il coinvolgimento dei clienti ed evitare costose sanzioni. Non contiene tuttavia consulenza legale. Consulta sempre un avvocato per ottenere adeguati pareri legali.

Conformità a leggi, normative e requisiti del gestore

La violazione di leggi e normative in vigore nei luoghi di residenza dei clienti può comportare multe e sanzioni elevate. Per questo motivo, è essenziale conoscere le leggi relative alla messaggistica SMS in vigore in ogni paese o regione in cui l'azienda opera.

L'elenco seguente contiene link alle leggi chiave applicate alle comunicazioni tramite SMS nei principali mercati di tutto il mondo.

- Stati Uniti: alcuni tipi di messaggi SMS sono soggetti al Telephone Consumer Protection Act (TCPA) del 1991. Per ulteriori informazioni, consulta le [regole e le normative](#) nel sito Web della Federal Communications Commission.
- Regno Unito: alcuni tipi di messaggi SMS sono soggetti alla direttiva CE relativa alla vita privata e alle comunicazioni elettroniche del 2003. Per ulteriori informazioni, consulta [What are PECR?](#) nel sito Web dell'Information Commissioner's Office del Regno Unito.
- Unione Europea: alcuni tipi di messaggi SMS sono soggetti alla direttiva relativa alla vita privata e alle comunicazioni elettroniche del 2002, detta anche "direttiva ePrivacy". Per ulteriori informazioni, consulta il [testo integrale della legge](#) nel sito Web europa.eu.
- Canada: alcuni tipi di messaggi SMS sono soggetti al Fighting Internet and Wireless Spam Act, comunemente noto come CASL (Canada's Anti-Spam Law). Per ulteriori informazioni, consulta il [testo integrale della legge](#) nel sito Web del parlamento canadese.
- Giappone: alcuni tipi di messaggi SMS sono soggetti alla legge sulla regolamentazione della trasmissione di posta elettronica specifica. Per ulteriori informazioni, consulta [Japan's Countermeasures Against Spam](#) nel sito Web del Ministero degli Affari interni e delle Comunicazioni del Giappone.

In qualità di mittente, queste leggi possono essere applicabili anche se l'azienda o l'organizzazione non ha sede in uno di questi Paesi. Alcune delle leggi dell'elenco sono state originariamente create per e-mail o chiamate telefoniche non sollecitate, ma sono state interpretate o estese in modo da coprire anche i messaggi SMS. Altri paesi e regioni potrebbero disporre di legislazione specifica relativa alla trasmissione di messaggi SMS. Consulta un avvocato in ogni paese o regione in cui si trovano i tuoi clienti per ottenere consulenza legale.

In molti Paesi, i gestori locali hanno sostanzialmente l'autorità di determinare il tipo di traffico gestito nelle rispettive reti. Ciò significa che i gestori potrebbero applicare limitazioni ai contenuti SMS che non risultano conformi ai requisiti minimi delle leggi locali.

Acquisizione dell'autorizzazione

Non inviare mai messaggi a destinatari che non hanno richiesto esplicitamente di ricevere i tipi specifici di messaggi che intendi inviare. Non condividere elenchi di consenso esplicito, nemmeno tra organizzazioni all'interno della stessa azienda.

Se i destinatari possono registrarsi per ricevere i messaggi mediante un modulo online, aggiungi sistemi che impediscono che script automatizzati possano eseguire la sottoscrizione all'insaputa dei destinatari. È inoltre necessario limitare il numero di volte in cui un utente può inviare un numero di telefono in una singola sessione.

Quando ricevi una richiesta con consenso esplicito per i messaggi SMS, invia al destinatario un messaggio per chiedere conferma del desiderio di ricevere tali messaggi. Non inviare altri messaggi al destinatario finché non conferma la propria sottoscrizione. Un messaggio per la conferma della sottoscrizione potrebbe essere simile all'esempio seguente:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may  
apply. Reply HELP for help, STOP to cancel.
```

Mantieni una documentazione della data, dell'ora e dell'origine di ogni richiesta con consenso esplicito e di ogni conferma. Potrebbe essere utile se un operatore o un ente normativo ne fa richiesta, nonché per eseguire i controlli di routine dell'elenco dei clienti.

Flusso di lavoro del consenso esplicito

In alcuni casi, ad esempio la registrazione mediante numeri verdi gratuiti o codici brevi negli Stati Uniti, 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.

The first screen, titled 'examplecorp', asks the user if they are ready to create an account. It provides fields for 'First name*', 'Last name*', and 'Email address*', each with a red asterisk indicating required information. A 'Next >' button is at the bottom.

The second screen, also titled 'examplecorp', explains Multi-Factor Authentication (MFA) and asks if the user wants to enable it. It has two radio button options: 'Enable MFA' (selected) and 'Disable MFA (less secure)'. A 'Next >' button is at the bottom.

The third screen, titled 'examplecorp', asks how the user wants to receive MFA messages. It has three radio button options: 'Email' (selected), 'Phone call', and 'Text message'. Below the 'Text message' option, there is a note about message and data rates, and links for 'Terms & Conditions' and 'Privacy Policy'. A 'Mobile number' field is present, with a note stating: 'When you press the **Next** button, we'll send you an MFA password to verify your phone number.' A blue bracket on the right side of the screen indicates that this section only appears when 'Text message' is selected. A 'Next >' button is at the bottom.

1. User provides basic account information.

2. User decides whether to enable MFA.

3. If MFA enabled, user chooses how to receive MFA token.

The fourth screen shows an incoming text message from '67876' with the text: 'Your ExampleCorp Multi-factor Authentication code is 918273. Text HELP for more info or STOP to opt out.' A 'Text Message' input field and a 'Send' button are at the bottom.

The fifth screen, titled 'examplecorp', shows the user entering the MFA token. It displays the message: 'We sent a text message to you at (425) 555-0142. Enter the six digit code in that message to confirm your phone number.' Below the message is a 'Resend code' link and a numeric keypad with digits 1-9, 0, and symbols for backspace and special characters. A 'Send' button is at the bottom.

4. If user chooses to receive MFA token by text, send a token.

5. User enters MFA token to verify phone number.

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 di opt-in, puoi fornire una schermata dello script di attivazione, del materiale di marketing o del database in cui vengono raccolti i numeri. I gestori di telefonia mobile potrebbero avere ulteriori domande su questi casi d'uso se l'opzione di consenso esplicito non è chiara o se il caso d'uso supera determinati volumi.

Non inviare messaggi a elenchi obsoleti

Le persone cambiano spesso numeri di telefono. Un numero di telefono per il quale è stato acquisito il consenso all'invio di messaggi due anni fa potrebbe ora appartenere a un altro utente. Non utilizzare un vecchio elenco di numeri di telefono per un nuovo programma di messaggistica; se lo fai, è possibile che alcuni messaggi non vengano recapitati perché il numero non è più operativo e alcuni destinatari potrebbero non ricordare di avere precedentemente fornito il loro consenso.

Controllo degli elenchi dei clienti

Se invii campagne SMS ricorrenti, controlla regolarmente gli elenchi dei clienti. Tale controllo garantisce che i messaggi vengano inviati solo ai clienti che sono interessati a riceverli.

Quando controlli l'elenco, invia a ogni cliente che ha acconsentito esplicitamente un messaggio di promemoria della sottoscrizione con le informazioni per annullarla. Un messaggio di promemoria potrebbe essere simile all'esempio seguente:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply  
HELP for help, STOP to unsubscribe.
```

Conservazione della documentazione

Conserva la documentazione che indica quando ogni cliente ha richiesto di ricevere SMS e quali messaggi hai inviato a ciascun cliente. Molti paesi e regioni di tutto il mondo richiedono ai mittenti di

SMS di conservare tale documentazione in modo che sia facilmente recuperabile. Tali informazioni potrebbero inoltre essere richieste dagli operatori di telefonia mobile in qualsiasi momento. Le informazioni esatte da fornire variano a seconda del paese o della regione. Per ulteriori informazioni sui requisiti di conservazione della documentazione, esamina le normative sui messaggi SMS commerciali di ogni paese o regione in cui si trovano i tuoi clienti.

Talvolta, un operatore o un ente normativo ci chiede di fornire una prova del fatto che un cliente ha acconsentito a ricevere messaggi da te. In queste situazioni, ti Supporto contatta fornendoti un elenco delle informazioni richieste dal corriere o dall'agenzia. Se non sei in grado di fornire le informazioni necessarie, potremmo sospendere la tua possibilità di inviare ulteriori messaggi SMS.

Rendi i tuoi messaggi chiari, attendibili e concisi

Gli SMS sono uno strumento unico. Il character-per-message limite di 160 significa che i tuoi messaggi devono essere concisi. Le tecniche che potresti utilizzare in altri canali di comunicazione, come la posta elettronica, potrebbero non essere applicabili al canale SMS e potrebbero persino sembrare non attendibili o ingannevoli se utilizzate con i messaggi SMS. Se il contenuto dei messaggi non è in linea con le best practice, i destinatari potrebbero ignorarli; nel peggiore dei casi, i gestori di telefonia mobile potrebbero identificare i messaggi come spam e bloccare i messaggi futuri ricevuti dal tuo numero di telefono.

Questa sezione fornisce alcuni suggerimenti e idee per creare un corpo efficace del messaggio SMS.

Identificati come mittente

I destinatari devono essere in grado di capire subito che un messaggio proviene da te. I mittenti che si attengono a questa best practice includono un identificativo ("nome del programma") all'inizio di ogni messaggio.

Non fare questo:

Your account has been accessed from a new device. Reply Y to confirm.

Prova invece questo:

ExampleCorp Financial Alerts: You have logged in to your account from a new device. Reply Y to confirm, or STOP to opt-out.

Non cercare di far sembrare il tuo messaggio un person-to-person messaggio

Alcuni esperti di marketing sono tentati di aggiungere un tocco personale ai loro messaggi SMS facendoli sembrare inviati da un individuo. Tuttavia, questa tecnica potrebbe essere fraintesa come un tentativo di phishing.

Non fare questo:

```
Hi, this is Jane. Did you know that you can save up to 50% at  
Example.com? Click here for more info: https://www.example.com.
```

Prova invece questo:

```
ExampleCorp Offers: Save 25-50% on sale items at Example.com. Click here  
to browse the sale: https://www.example.com. Text STOP to opt-out.
```

Fai attenzione quando parli di soldi

I truffatori spesso sfruttano il desiderio delle persone di risparmiare e ricevere denaro. Non far sembrare le offerte troppo belle per essere vere. Non usate il richiamo dei soldi per ingannare le persone. Non utilizzare simboli di valuta per fare riferimento al denaro.

Non fare questo:

```
Save big $$$ on your next car repair by going to https://  
www.example.com.
```

Prova invece questo:

```
ExampleCorp Offers: Your ExampleCorp insurance policy gets you discounts  
at 2300+ repair shops nationwide. More info at https://www.example.com.  
Text STOP to opt-out.
```

Usa solo i caratteri necessari

I marchi sono spesso propensi a proteggere la propria immagine includendo simboli come TM o [®] nei rispettivi messaggi. Tuttavia, questi simboli non fanno parte del set standard di caratteri (noto come alfabeto GSM) che può essere incluso in un messaggio SMS di 160 caratteri. Quando invii un messaggio contenente uno di questi caratteri, il messaggio viene inviato automaticamente utilizzando un sistema di codifica dei caratteri diverso, che supporta solo 70 caratteri per parte del messaggio. Di

conseguenza, il messaggio potrebbe venire suddiviso in più parti. Poiché ti viene addebitato un costo per ogni parte del messaggio inviata, l'invio dell'intero messaggio potrebbe costarti più del previsto. Inoltre, i destinatari potrebbero ricevere più messaggi sequenziali, anziché un unico messaggio. Per ulteriori informazioni sulla codifica dei caratteri degli SMS, consulta [Limiti relativi ai caratteri per gli SMS in Amazon SNS](#).

Non fare questo:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at  
example.com and use the promo code WIDGET.
```

Prova invece questo:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at  
example.com and use the promo code WIDGET.
```

 Note

I due esempi precedenti sono pressoché identici, ma il primo contiene il simbolo del marchio registrato (®), che non fa parte dell'alfabeto GSM. Di conseguenza, il primo esempio viene inviato come messaggio in due parti, mentre il secondo esempio viene inviato come messaggio unico.

Usa collegamenti validi e sicuri

Se il messaggio include link, ricontrolla i link per assicurarti che funzionino. Verifica i link su un dispositivo esterno alla rete aziendale per assicurarti che vengano risolti correttamente. A causa del limite di 160 caratteri dei messaggi SMS, 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

La limitazione di 160 caratteri del canale SMS induce alcuni mittenti a credere di dover utilizzare ampiamente le abbreviazioni nei loro messaggi. Tuttavia, l'uso eccessivo di abbreviazioni può sembrare poco professionale per molti lettori e potrebbe indurre alcuni utenti a segnalare il tuo messaggio come spam. È possibile scrivere un messaggio coerente senza utilizzare un numero eccessivo di abbreviazioni.

Non fare questo:

Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.

Prova invece questo:

ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets at example.com. Text STOP to opt-out.

Invio di risposte appropriate

Quando un destinatario risponde ai tuoi messaggi, assicurati di rispondere con informazioni utili. Quando un cliente risponde a uno dei tuoi messaggi con la parola chiave "HELP", ad esempio, invia informazioni sul programma a cui ha effettuato la sottoscrizione, sul numero di messaggi che invierai ogni mese e sui modi in cui potrà contattarti per ulteriori informazioni. Una risposta HELP potrebbe essere simile all'esempio seguente:

HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2 msgs/month. Msg & data rates may apply. Reply STOP to cancel.

Quando un cliente risponde con la parola chiave "STOP", comunica al cliente che non riceverà ulteriori messaggi. Una risposta STOP potrebbe essere simile all'esempio seguente:

You're unsubscribed from ExampleCorp alerts. No more messages will be sent. Reply HELP, email help@example.com, or call 425-555-0199 for more info.

Adattamento dell'invio al coinvolgimento

Le priorità dei clienti possono cambiare nel tempo. Se i clienti non ritengono più utili i tuoi messaggi, potrebbero cancellarsi completamente dalla ricezione o addirittura segnalare i tuoi messaggi come non sollecitati. Per questi motivi, è importante adattare le tue procedure di invio al coinvolgimento dei clienti.

Per i clienti che raramente interagiscono con i tuoi messaggi, dovresti adattare la frequenza dei messaggi. Se ai clienti coinvolti invii messaggi settimanali, ad esempio, potresti creare un riepilogo mensile separato per i clienti meno coinvolti.

Rimuovi infine dai tuoi elenchi i clienti che non sono affatto coinvolti. Questo evita che i tuoi messaggi generino frustrazione nei clienti e ti consente inoltre di risparmiare denaro e proteggere la tua reputazione come mittente.

Invio in orari appropriati

Invia messaggi solo durante il normale orario di ufficio diurno. Se invii messaggi all'ora di cena o nel cuore della notte, è probabile che i clienti annullino la sottoscrizione ai tuoi elenchi per evitare di essere disturbati. Non è inoltre opportuno inviare messaggi SMS quando i tuoi clienti non possono rispondere immediatamente.

Se invii campagne o percorsi a un pubblico molto vasto, ricontrolla la velocità di trasmissione effettiva dei numeri di origine. Dividi il numero di destinatari per la velocità di trasmissione effettiva per determinare il tempo necessario per inviare messaggi a tutti i destinatari.

Astensione dalla ripetizione in più canali

Se nelle tue campagne utilizzi più canali di comunicazione (ad esempio e-mail, SMS e notifiche push), non inviare lo stesso messaggio in ogni canale. Se invii lo stesso messaggio tramite più canali contemporaneamente, il tuo comportamento di invio verrà probabilmente percepito dai clienti come fastidioso anziché utile.

Utilizzo di codici brevi dedicati

Se utilizzi codici brevi, mantieni un codice breve separato per ogni marchio e per ciascun tipo di messaggio. Se l'azienda possiede due marchi, ad esempio, utilizza un codice breve separato per ciascuno. Analogamente, se invii messaggi sia transazionali che promozionali, utilizza un codice breve separato per ciascun tipo di messaggio. Per ulteriori informazioni sulla richiesta di codici brevi,

consulta [Richiesta di codici brevi per la messaggistica SMS AWS End User Messaging SMS nella Guida](#) per l'AWS End User Messaging SMS utente.

Verifica i numeri di telefono di destinazione

Quando invii messaggi SMS tramite Amazon SNS, ti viene addebitato un costo per ogni parte di messaggio inviata. Il prezzo da pagare per parte di messaggio varia in base al Paese o all'area geografica del destinatario. Per ulteriori informazioni sui prezzi degli SMS, consulta la pagina [Prezzi AWS mondiali degli SMS](#).

Quando Amazon SNS accetta una richiesta di invio di un messaggio SMS (come risultato di una chiamata all'[SendMessage](#) API o come risultato del lancio 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 messaggi SMS e tenta di recapitarli. Per questo motivo, è necessario verificare che i numeri di telefono a cui si inviano messaggi siano numeri di cellulare validi. 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 simulatore SMS](#) 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 una Regione AWS. Amazon SNS è disponibile in diverse versioni. Regioni AWS Per un elenco delle regioni in cui Amazon SNS è disponibile, consulta [Riferimenti generali di AWS](#).

I numeri di telefono utilizzati per i messaggi SMS, inclusi codici brevi, codici lunghi, numeri gratuiti e numeri 10DLC, non possono essere replicati nelle Regioni AWS. Di conseguenza, per utilizzare Amazon SNS in più regioni, devi richiedere numeri di telefono distinti in ogni regione in cui desideri utilizzare Amazon SNS. Ad esempio, se utilizzi un codice breve per inviare messaggi di testo a destinatari negli Stati Uniti d'America, devi richiedere codici brevi separati per ciascuno 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, numeri 10DLC e numeri gratuiti. Ciascuno di questi tipi di numeri di telefono segue un percorso diverso per raggiungere il destinatario. La

disponibilità di più tipi di numeri di telefono, uguali Regione AWS o distribuiti su più numeri, Regioni AWS offre un ulteriore livello di ridondanza, che può contribuire a migliorare la resilienza.

Limiti e restrizioni degli SMS

Per i limiti e le restrizioni relativi agli [SMS](#), consulta [Limiti e restrizioni relativi agli SMS e MMS nella Guida](#) per l'AWS End User Messaging SMS utente.

Gestione delle parole chiave di esclusione

I destinatari degli SMS possono utilizzare i propri dispositivi per disattivare i messaggi rispondendo con una parola chiave. Per ulteriori informazioni, consulta [Richiesta di non ricevere i messaggi SMS](#).

CreatePool

Utilizza l'azione API `CreatePool` per creare un nuovo pool e associare un'identità di origine specificata al pool. Per ulteriori informazioni, consulta [CreatePool AWS End User Messaging SMSAPI Reference](#).

PutKeyword

Utilizza l'azione `PutKeyword` API per creare o aggiornare una configurazione di parole chiave su un numero di telefono o un pool di origine. Per ulteriori informazioni, consulta la sezione [PutKeyword AWS End User Messaging SMSAPI Reference](#).

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 CLI](#) in AWS End User Messaging SMS.

Limiti relativi ai caratteri per gli SMS in Amazon SNS

Un singolo messaggio SMS può contenere fino a 140 byte di informazioni. Il numero di caratteri che può essere incluso in un singolo messaggio SMS dipende dal tipo di caratteri che il messaggio contiene.

Se utilizza solo [caratteri del set di caratteri GSM 03.38](#), noto anche come alfabeto GSM a 7 bit, il messaggio può contenere fino a 160 caratteri. Se il messaggio contiene caratteri non appartenenti al set di caratteri GSM 03.38, il messaggio può includere fino a 70 caratteri. Quando si invia un messaggio SMS, Amazon SNS determina automaticamente la codifica più efficiente da utilizzare.

Quando un messaggio contiene più caratteri del numero massimo, il messaggio viene suddiviso in più parti. Quando i messaggi vengono suddivisi in più parti, ogni parte contiene informazioni aggiuntive sulla parte del messaggio che lo precede. Quando il dispositivo del destinatario riceve parti del messaggio separate in questo modo, utilizza queste informazioni aggiuntive per garantire che tutte le parti del messaggio vengano visualizzate nell'ordine corretto. A seconda dell'operatore e del dispositivo mobile del destinatario, è possibile che più messaggi vengano visualizzati come un singolo messaggio o come una sequenza di messaggi separati. Di conseguenza, il numero di caratteri per ogni parte di un messaggio viene ridotto a 153 (per i messaggi contenenti solo caratteri GSM 03.38) o a 67 (per i messaggi contenenti altri caratteri). È possibile stimare quante parti di messaggio contiene il messaggio prima di inviarlo utilizzando gli strumenti di calcolo della lunghezza degli SMS, molti dei quali sono disponibili online. La dimensione massima supportata di qualsiasi messaggio è 1.600 caratteri GSM o 630 caratteri non GSM. Per ulteriori informazioni sul throughput (velocità di trasmissione effettiva) e sulla dimensione dei messaggi, consulta la pagina relativa ai [limiti di caratteri per gli SMS in Amazon Pinpoint](#) nella guida per l'utente di Amazon Pinpoint.

Per visualizzare il numero di parti per ogni messaggio inviato, è innanzitutto necessario abilitare le impostazioni relative allo [streaming di eventi](#). Quando si esegue questa operazione, Amazon SNS produce un evento `_SMS.SUCCESS` quando il messaggio viene recapitato al fornitore di servizi mobili del destinatario. Il record di evento `_SMS.SUCCESS` contiene un attributo denominato `attributes.number_of_message_parts`. Questo attributo specifica il numero di parti di messaggio contenute nel messaggio.

Important

Quando si invia un messaggio che contiene più di una parte, viene addebitato il numero di parti contenute nel messaggio.

Set di caratteri GSM 03.38

Nella tabella seguente sono elencati tutti i caratteri presenti nel set di caratteri GSM 03.38. Se invii un messaggio che include solo i caratteri riportati nella tabella seguente, il messaggio può contenere fino a 160 caratteri.

Caratteri standard GSM 03.38

A	B	C	D	E	F	G	H	I	J	K	L	M

Caratteri standard GSM 03.38												
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
\$	=	!	>	#	-	ı	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

In aggiunta ai caratteri riportati nella tabella precedente, il set di caratteri GSM 03.38 include diversi simboli. Ognuno di essi, tuttavia, viene conteggiato come due caratteri poiché include anche un carattere di escape non visibile:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Il set di caratteri GSM 03.38 include, infine, anche i caratteri non stampati seguenti:

- Carattere di spazio

- Controllo di avanzamento riga, che indica la fine di una riga di testo e l'inizio di un'altra
- Controllo di ritorno a capo, che passa all'inizio di una riga di testo (in genere dopo un carattere di avanzamento riga)
- Controllo di escape, che viene aggiunto automaticamente ai caratteri dell'elenco precedente

Messaggi di esempio

Questa sezione contiene vari messaggi SMS di esempio. Per ogni esempio, questa sezione mostra il numero totale di caratteri e il numero di parti del messaggio.

Esempio 1: messaggio lungo che contiene solo caratteri presenti nell'alfabeto GSM 03.38

Il seguente messaggio contiene solo caratteri presenti nell'alfabeto GSM 03.38.

Hello Carlos. Your Example Corp. bill of \$100 is now available. Autopay is scheduled for next Thursday, April 9. To view the details of your bill, go to <https://example.com/bill1>.

Il messaggio precedente contiene 180 caratteri, quindi deve essere diviso in più parti. Quando un messaggio è diviso in più parti, ogni parte può contenere 153 caratteri GSM 03.38. Di conseguenza, questo messaggio viene inviato come messaggio in due parti.

Esempio 2: un messaggio che contiene caratteri multibyte

Il seguente messaggio contiene diversi caratteri cinesi, tutti al di fuori dell'alfabeto GSM 03.38.

• #####1994#7#####

Il messaggio precedente contiene 71 caratteri. Tuttavia, poiché quasi tutti i caratteri nel messaggio sono al di fuori dell'alfabeto GSM 03.38, viene inviato come messaggio in due parti. Ciascuna di queste parti può contenere un massimo di 67 caratteri.

Esempio 3: messaggio che contiene un singolo carattere non GSM

Il seguente messaggio contiene un singolo carattere che non fa parte dell'alfabeto GSM 03.38. In questo esempio, il carattere è una virgoletta singola di chiusura ('), che è un carattere diverso da un normale apostrofo ('). Le applicazioni di elaborazione testi come Microsoft Word spesso sostituiscono automaticamente gli apostrofi con le virgolette singole di chiusura. Se si redigono messaggi SMS in Microsoft Word e si incollano in Amazon SNS, è necessario rimuovere questi caratteri speciali e sostituirli con apostrofi.

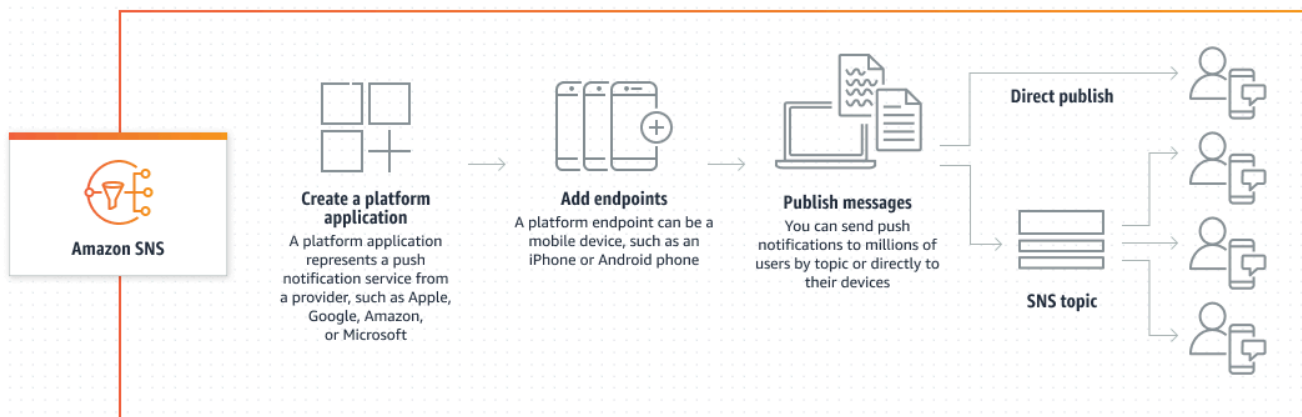
John: Your appointment with Dr. Salazar's office is scheduled for next Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.

Il messaggio precedente contiene 130 caratteri. Tuttavia, poiché contiene il carattere di virgoletta singola di chiusura, che non fa parte dell'alfabeto GSM 03.38, viene inviato come messaggio in due parti.

Se si sostituisce il carattere di virgoletta singola di chiusura in questo messaggio con un apostrofo (che fa parte dell'alfabeto GSM 03.38), il messaggio viene inviato come un messaggio singolo.

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 utente di Amazon SNS](#)
- [Configurazione delle notifiche push con Amazon SNS](#)
- [Configurazione di un'app mobile in Amazon SNS](#)
- [Utilizzo di Amazon SNS per notifiche push su dispositivi mobili](#)
- [Attributi dell'app mobile Amazon SNS](#)
- [Notifiche degli eventi delle applicazioni Amazon SNS per applicazioni mobili](#)
- [Operazioni API push per dispositivi mobili](#)
- [Errori comuni dell'API push mobile di Amazon SNS](#)

- [Utilizzo dell'attributo time to live message di Amazon SNS per le notifiche push mobili](#)
- [Regioni supportate dall'applicazione mobile Amazon SNS](#)
- [Best practice per la gestione delle notifiche push mobili di Amazon SNS](#)

Come funzionano le notifiche utente di Amazon SNS

L'invio di messaggi di notifica push a desktop e dispositivi mobili viene eseguito mediante uno dei seguenti servizi di notifica push supportati:

- Amazon Device Messaging (ADM)
- Servizio Apple Push Notification (APNs) per iOS e Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging (FCM)
- Servizio di notifica push Microsoft per Windows Phone (MPNS)
- Windows Push Notification Services (WNS)

I servizi di notifica push, come APNs FCM, mantengono una connessione con ogni app e dispositivo mobile associato registrato 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 di dispositivo per creare un endpoint mobile a cui può inviare direttamente messaggi di notifica push. Per consentire la comunicazione tra Amazon SNS e i differenti servizi di notifica push, devi inviare le tue credenziali del servizio di notifica push ad Amazon SNS affinché le utilizzi a tuo nome. Per ulteriori informazioni, consulta [Configurazione delle notifiche push con Amazon SNS](#).

Oltre a inviare direttamente messaggi di notifica push, puoi anche utilizzare Amazon SNS per inviare messaggi a endpoint mobili che dispongono di una sottoscrizione a un argomento. Il concetto è lo stesso della sottoscrizione di altri tipi di endpoint, ad esempio Amazon SQS, HTTP/S, e-mail e SMS, a un argomento, come descritto in [Che cos'è Amazon SNS?](#). La differenza è che Amazon SNS comunica mediante i servizi di notifica push affinché gli endpoint mobili con sottoscrizione ricevano i messaggi di notifica push inviati all'argomento.

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. Utilizzare le credenziali per creare un oggetto application platform (PlatformApplicationArn) utilizzando Amazon SNS. Per ulteriori informazioni, consulta [Creazione di un'applicazione della piattaforma Amazon SNS](#).
3. Utilizza le credenziali ottenute per richiedere un token per il dispositivo mobile e l'app dal servizio di notifiche push. Il token che ottieni rappresenta il tuo dispositivo e la tua app per dispositivi mobili.
4. Utilizzare il token dispositivo e PlatformApplicationArn per creare un oggetto endpoint piattaforma (EndpointArn) utilizzando Amazon SNS. Per ulteriori informazioni, consulta [Configurazione di un endpoint della piattaforma Amazon SNS per le notifiche mobili](#).
5. EndpointArn viene quindi utilizzato per [pubblicare un messaggio in una app in un dispositivo mobile](#). Per ulteriori informazioni, consulta [Messaggistica diretta per dispositivi mobili Amazon SNS](#) e API [Publish](#) (Pubblica) nella Guida API di Amazon Simple Notification Service.

Configurazione di un'app mobile in Amazon SNS

Questo argomento descrive come configurare le applicazioni mobili Console di gestione AWS utilizzando le informazioni descritte in [Prerequisiti per le notifiche utente Amazon SNS](#).

Prerequisiti per le notifiche utente Amazon SNS

Per iniziare a utilizzare le notifiche push mobili di Amazon SNS, avrai bisogno di quanto segue:

- Un set di credenziali per la connessione a uno dei servizi di notifica push supportati: ADM, Baidu, FCM APNs, MPNS o WNS.
- Un token di dispositivo o un ID di registrazione per l'app per dispositivi mobili e il dispositivo.
- Amazon SNS configurato per l'invio di messaggi di notifica push a endpoint mobili.
- Un app per dispositivi mobili registrata e configurata per l'utilizzo di uno dei servizi di notifica push supportati.

La registrazione dell'applicazione a un servizio di notifica mobile comporta varie fasi. Amazon SNS necessita di alcune delle informazioni fornite al servizio di notifica push per inviare direttamente messaggi di notifica push all'endpoint mobile. In genere, hai bisogno delle credenziali necessarie per la connessione al servizio di notifica push, il token di dispositivo o l'ID di registrazione (che rappresenta il dispositivo mobile e l'app) ricevuti dal servizio di notifica push e l'app per dispositivi mobili registrata al servizio di notifica push.

La forma esatta delle credenziali differisce a seconda della piattaforma mobile, ma in ogni caso, queste credenziali devono essere fornite durante una connessione alla piattaforma. Un set di credenziali viene generato per ogni app per dispositivi mobili e deve essere utilizzato per inviare un messaggio a qualsiasi istanza di quell'app.

I nomi specifici variano a seconda del servizio di notifica push utilizzato. Ad esempio, quando si utilizza APNs come servizio di notifica push, è necessario un token del dispositivo. Quando invece utilizzi FCM, il token di dispositivo equivalente è denominato ID di registrazione. Il token di dispositivo o l'ID di registrazione è una stringa inviata all'applicazione dal sistema operativo del dispositivo mobile. Identifica in modo univoco un'istanza di un'app per dispositivi mobili eseguita su un determinato dispositivo mobile e può essere considerata come l'identificatore univoco di questa coppia app-dispositivo.

Amazon SNS archivia le credenziali (e altre impostazioni) come risorsa di applicazione di piattaforma. I token 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 Console di gestione AWS o il push mobile di Amazon SNS. APIs Per ulteriori informazioni, consulta [Configurazione delle notifiche push con Amazon SNS](#).

Creazione di un'applicazione della piattaforma Amazon SNS

Per inviare notifiche da Amazon SNS 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 sia per l'app che per il dispositivo mobile. Questo endpoint consente ad Amazon SNS di inviare messaggi al dispositivo.

Per creare un'applicazione della piattaforma

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, seleziona Notifiche push.
3. Nella sezione Platform applications (Applicazioni di piattaforma), scegli Create platform application (Crea applicazione di piattaforma).
4. Scegli il tuo Regione AWS. Per un elenco delle regioni AWS in cui è possibile creare applicazioni per dispositivi mobili, consulta [Regioni supportate dall'applicazione mobile Amazon SNS](#).

5. Inserisci i seguenti dettagli dell'applicazione:

- Nome dell'applicazione: fornisci un nome per l'applicazione della piattaforma. Il nome deve contenere da 1 a 256 caratteri e può contenere lettere maiuscole e minuscole, numeri, caratteri di sottolineatura, trattini e punti.
- Piattaforma di notifica push: seleziona il servizio di notifica appropriato con cui è registrata l'app (ad esempio, Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM)).

6. A seconda della piattaforma selezionata, dovrai fornire credenziali specifiche:

- Per APNs(Apple Push Notification Service): scegli tra l'autenticazione basata su token o quella basata su certificati.
 - Per l'autenticazione basata su token, carica un file.p8 (generato tramite Keychain Access).
 - Per l'autenticazione basata su certificati, carica un file.p12 (anch'esso esportato da Keychain Access).
- Per FCM (Firebase Cloud Messaging): inserisci la chiave del server dalla console Firebase.
- Per altre piattaforme (come ADM o GCM): inserisci le rispettive chiavi o credenziali API.

7. Dopo aver inserito i dettagli necessari, scegli Crea applicazione della piattaforma. Questa azione registra l'app con Amazon SNS e crea l'oggetto applicativo della piattaforma corrispondente.

8. Al momento della creazione, Amazon SNS genera e restituisce un [PlatformApplicationArn](#) (Amazon Resource Name). Questo ARN identifica in modo univoco l'applicazione della piattaforma e viene utilizzato per la creazione di endpoint per dispositivi mobili.

Configurazione di un endpoint della piattaforma Amazon SNS per le notifiche mobili

Quando un'app e un dispositivo mobile si registrano con un servizio di notifica push (come APNs o Firebase Cloud Messaging), il servizio di notifica push restituisce un token del dispositivo. Amazon SNS utilizza questo token del dispositivo per creare un endpoint della piattaforma, che funge da destinazione per l'invio di messaggi di notifica push diretti all'app sul dispositivo. L'endpoint della piattaforma funge da ponte, instradando i messaggi inviati da Amazon SNS al servizio di notifica push per la consegna al dispositivo mobile corrispondente. Per ulteriori informazioni, consultare [Prerequisiti per le notifiche utente Amazon SNS](#) e [Configurazione delle notifiche push con Amazon SNS](#).

Comprendere i token dei dispositivi e gli endpoint della piattaforma

Un token del dispositivo identifica in modo univoco un dispositivo mobile registrato con un servizio di notifica push (ad esempio APNs, Firebase Cloud Messaging). Quando un'app si registra con il servizio di notifica push, genera un token di dispositivo specifico per quell'app e quel dispositivo. Amazon SNS utilizza questo token del dispositivo per creare un endpoint della piattaforma all'interno dell'applicazione della piattaforma corrispondente.

L'endpoint della piattaforma consente ad Amazon SNS di inviare messaggi di notifica push al dispositivo tramite il servizio di notifica push, mantenendo la connessione tra l'app e il dispositivo dell'utente.

Creazione di un endpoint di piattaforma

Per inviare notifiche push a un'app con Amazon SNS, è necessario prima registrare il token di dispositivo di quell'app in Amazon SNS richiamando l'operazione di creazione di endpoint di piattaforma. Questa azione utilizza l'Amazon Resource Name (ARN) dell'applicazione di piattaforma e del token di dispositivo come parametri e restituisce l'ARN dell'endpoint di piattaforma creato.

L'[CreatePlatformEndpoint](#) azione esegue le seguenti operazioni:

- Se l'endpoint della piattaforma esiste già, non crearlo di nuovo. Restituisce all'intermediario l'ARN dell'endpoint di 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 all'intermediario l'ARN dell'endpoint di piattaforma appena creato.

Non devi chiamare immediatamente l'operazione di creazione di endpoint di piattaforma a ogni avvio di un'applicazione poiché questo approccio non fornisce sempre un endpoint funzionante. Ciò può verificarsi, ad esempio, quando un'app viene disinstallata e reinstallata nello stesso dispositivo e il relativo endpoint esiste ma è disattivato. Una procedura di registrazione corretta deve garantire quanto segue:

1. L'endpoint di piattaforma esiste già per la combinazione app-dispositivo.
2. Il token di dispositivo nell'endpoint di piattaforma è il token di dispositivo valido più recente.
3. L'endpoint di piattaforma è attivato e pronto all'uso.

Pseudocodice

Lo pseudocodice seguente descrive una pratica consigliata per la creazione di un endpoint di piattaforma funzionante, corrente e attivato in un'ampia gamma di condizioni di avvio. Questo approccio funziona indipendentemente se si tratta della prima registrazione dell'app, se l'endpoint di piattaforma per l'app esiste già, se l'endpoint di piattaforma è attivato, ha il token di dispositivo corretto e così via. Non è un problema chiamarlo più volte in successione, in quanto non creerà endpoint di piattaforma duplicati o modificherà l'endpoint di piattaforma esistente se è già aggiornato e attivato.

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
    # this is a first-time registration
    call create platform endpoint
    store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
    # the platform endpoint was deleted
    call create platform endpoint with the latest device token
    store the returned platform endpoint ARN
else
    if (the device token in the endpoint does not match the latest one) or
        (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 e anche in caso di notifica della modifica di un token di dispositivo a Amazon SNS. In quest'ultimo caso, è sufficiente chiamare l'operazione con il valore di token di dispositivo più recente. Alcuni punti da considerare in relazione a questo approccio:

- Esistono due casi in cui può chiamare l'operazione di creazione di endpoint di piattaforma. All'inizio, quando l'app non conosce il relativo ARN di endpoint di piattaforma, come avviene durante una prima registrazione, Viene chiamato anche se la chiamata all'`GetEndpointAttributes` iniziale fallisce con un'eccezione non trovata, come accadrebbe se l'applicazione conoscesse l'ARN dell'endpoint ma fosse stata eliminata.

- L'GetEndpointAttributesazione 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 Esempio SDK

Il codice seguente mostra come implementare lo pseudo codice precedente utilizzando i client Amazon SNS 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 and](#) Tools Reference AWS SDKs 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

SDK per Java 2.x

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <token> <platformApplicationArn>

            Where:
```

token - The device token or registration ID of the mobile device. This is a unique identifier provided by the device platform (e.g., Apple Push Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM) for Android devices) when the mobile app is registered to receive push notifications.

platformApplicationArn - The ARN value of platform application. You can get this value from the AWS Management Console.\s

```

        """;

    if (args.length != 2) {
        System.out.println(usage);
        return;
    }

    String token = args[0];
    String platformApplicationArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    createEndpoint(snsClient, token, platformApplicationArn);
}

public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
    System.out.println("Creating platform endpoint with token " + token);
    try {
        CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
            .token(token)
            .platformApplicationArn(platformApplicationArn)
            .build();

        CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
        System.out.println("The ARN of the endpoint is " +
response.endpointArn());

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

```

```
}
```

Per ulteriori informazioni, consulta [Operazioni API push per dispositivi mobili](#).

Risoluzione dei problemi

Chiamata ripetuta dell'operazione di creazione di endpoint di piattaforma con un token di dispositivo obsoleto

Soprattutto per gli endpoint FCM, potreste pensare che sia meglio archiviare il primo token del dispositivo emesso dall'applicazione e quindi richiamare l'endpoint di creazione della piattaforma con quel token del dispositivo ogni volta all'avvio dell'applicazione. Ciò può sembrare corretto in quanto l'app non deve gestire lo stato del token di dispositivo e Amazon SNS aggiorna automaticamente il token di dispositivo al valore più recente. In verità, questa soluzione presenta alcuni seri inconvenienti:

- Amazon SNS si basa sul feedback di FCM per aggiornare i token di dispositivo obsoleti a quelli nuovi. FCM conserva le informazioni sui token di dispositivo obsoleti per qualche tempo, ma non indefinitamente. Una volta che FCM dimentica la connessione tra il vecchio token di dispositivo e il nuovo, Amazon SNS non sarà più in grado di aggiornare il token archiviato nell'endpoint di piattaforma al valore corretto, ma disattiverà l'endpoint di piattaforma.
- L'applicazione di piattaforma conterrà molteplici endpoint di piattaforma corrispondenti allo stesso token di dispositivo.
- Amazon SNS impone una quota al numero di endpoint di piattaforma che è possibile creare a partire dallo stesso token di dispositivo. Alla fine, la creazione di nuovi endpoint non riuscirà con un'eccezione di parametro non valido e il messaggio di errore seguente: "This endpoint is already registered with a different token." (Questo endpoint è già registrato con un altro token.).

Per ulteriori informazioni sulla gestione degli endpoint FCM, consulta [Gestione Amazon SNS 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 o FCM) 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 in quel token di dispositivo. Sebbene si possa ritenere che la soluzione migliore consista

semplicemente nel riattivare l'endpoint di piattaforma e continuare a pubblicare, nella maggior parte dei casi ciò non funzionerà: i messaggi pubblicati non verranno consegnati e l'endpoint di piattaforma sarà di nuovo disattivato poco tempo dopo.

Questo perché il token di dispositivo associato all'endpoint di piattaforma è effettivamente non valido. Le consegne a tale endpoint non possono riuscire in quanto non corrisponde più ad alcuna app installata. Al momento della pubblicazione successiva, la piattaforma mobile indicherà di nuovo ad Amazon SNS che il token di dispositivo non è valido e Amazon SNS disattiverà di nuovo l'endpoint di piattaforma.

Per riattivare un endpoint di piattaforma disattivato, è necessario associarlo a un token di dispositivo valido (con la chiamata dell'operazione di impostazione degli attributi di endpoint) e quindi attivarlo. Le consegne all'endpoint di piattaforma riusciranno solo dal momento dell'attivazione. L'unico caso in cui la riattivazione di un endpoint di piattaforma funziona senza l'aggiornamento del relativo token di dispositivo è quando un token di dispositivo associato a quell'endpoint che non era valido ridiventa valido. Ciò può verificarsi, ad esempio, quando un'app è stata disinstallata e quindi reinstallata nello stesso dispositivo mobile e riceve lo stesso token di dispositivo. L'approccio descritto qui sopra effettua questa operazione, assicurandosi di riattivare un endpoint di piattaforma solo dopo aver verificato che il token di dispositivo ad esso associato è quello più recente disponibile.

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. Questi token/IDs vengono aggiunti ad Amazon SNS per creare un endpoint per l'app e il dispositivo utilizzando l'API [PlatformApplicationArn](#). Una volta creato l'endpoint, [EndpointArn](#) viene restituito un messaggio che Amazon SNS utilizza per indirizzare le notifiche all'app/dispositivo corretto.

Puoi aggiungere i token del dispositivo o la registrazione IDs ad Amazon SNS nei seguenti modi:

- Aggiungi manualmente un singolo token tramite Console di gestione AWS
- Caricamento di diversi token usando l'API [CreatePlatformEndpoint](#)
- Registra i token per i dispositivi futuri

Per aggiungere manualmente un token del dispositivo o un ID di registrazione

1. Accedi alla [console Amazon SNS](#).

2. Nel riquadro di navigazione, seleziona Notifiche push.
3. Nella sezione Applicazioni della piattaforma, seleziona l'applicazione, quindi scegli Modifica. Se non hai ancora creato un'applicazione di piattaforma, segui subito la [Creazione di un'applicazione della piattaforma Amazon SNS](#) guida per farlo.
4. Scegliere Create Endpoint (Crea endpoint).
5. Nella casella Endpoint Token, inserisci il token o l'ID di registrazione, a seconda del servizio di notifica che stai utilizzando (ad esempio, l'ID di registrazione FCM).
6. (Facoltativo) Inserisci dati aggiuntivi nel campo Dati utente. Questi dati devono essere codificati in UTF-8 e avere dimensioni inferiori a 2 KB.
7. Scegliere Create Endpoint (Crea endpoint).

Una volta creato l'endpoint, puoi inviare messaggi direttamente al dispositivo mobile o ai dispositivi mobili abbonati a un argomento di Amazon SNS.

Per caricare diversi token utilizzando l'API **CreatePlatformEndpoint**

I passaggi seguenti mostrano come utilizzare l'app Java di esempio (bulkuploadpacchetto) fornita da AWS per caricare diversi token (token del dispositivo o registrazione IDs) su Amazon SNS. Puoi usare questa app di esempio per iniziare a caricare i token esistenti.

Note

Nei passaggi seguenti viene utilizzata l'IDE Java Eclipse. I passaggi presuppongono che tu abbia installato AWS SDK per Java e che tu disponga delle credenziali AWS di sicurezza per il tuo Account AWS. Per ulteriori informazioni, consulta [AWS SDK per Java](#). Per ulteriori informazioni sulle credenziali, consulta le [credenziali AWS di sicurezza](#) nella Guida per l'utente IAM.

1. Download e decomprimi il file [snsmobilepush.zip](#).
2. Crea un nuovo progetto Java in Eclipse e importa la SNSSamples cartella nel progetto.
3. Scarica la libreria [OpenCSV](#) e aggiungila al percorso di compilazione.
4. Nel BulkUpload.properties file, specifica quanto segue:
 - Il tuo ApplicationArn (ARN dell'applicazione della piattaforma).
 - Il percorso assoluto del file CSV contenente i token.

- Registrazione dei nomi di file per i token riusciti e non riusciti. Ad esempio `goodTokens.csv` e `badTokens.csv`.
- (Facoltativo) Una configurazione per il delimitatore, il carattere di virgoletta e il numero di thread da utilizzare.

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
```

5. Esegui l'`BatchCreatePlatformEndpointSampleapplicazione.java` per caricare i token su Amazon SNS. I token caricati correttamente verranno registrati in `goodTokens.csv`, mentre i token non validi verranno registrati in `badTokens.csv`.

Per registrare token dai dispositivi per future installazioni di app

Sono disponibili due opzioni per questo processo:

Usa il servizio Amazon Cognito

La tua app mobile può utilizzare credenziali di sicurezza temporanee per creare endpoint. Amazon Cognito è consigliato per generare credenziali temporanee. Per ulteriori informazioni, consulta la [Amazon Cognito Developer Guide](#)

Per tenere traccia [delle registrazioni delle](#) app, utilizza gli eventi Amazon SNS per ricevere notifiche quando vengono creati nuovi ARNs endpoint.

In alternativa, puoi utilizzare l'[ListEndpointByPlatformApplication](#) API per recuperare l'elenco degli endpoint registrati.

Utilizza un server proxy

Se l'infrastruttura dell'app supporta già la registrazione dei dispositivi al momento dell'installazione, puoi utilizzare il server come proxy. Inoltrerà i token del dispositivo ad Amazon SNS tramite [CreatePlatformEndpoint](#) l'API.

L'ARN dell'endpoint creato da Amazon SNS verrà restituito e potrà essere archiviato dal server per future pubblicazioni di messaggi.

Amazon SNS: metodi di autenticazione delle notifiche push di Apple

Puoi autorizzare Amazon SNS a inviare notifiche push all'app iOS o macOS fornendo informazioni che identificano l'utente come sviluppatore dell'app. Per autenticarsi, fornire una chiave o un certificato [durante la creazione di un'applicazione di piattaforma](#), entrambi possono essere ottenuti dal tuo account Apple Developer.

Chiave di firma dei token

Una chiave di firma privata 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 con cui autenticarsi APNs per ogni notifica push inviata. 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 certificato TLS che Amazon SNS utilizza per l'autenticazione quando APNs invia notifiche push. Si può ottenere il certificato dal proprio account sviluppatore Apple.

I certificati scadono dopo un anno. Alla scadenza, è necessario creare un nuovo certificato da fornire ad Amazon SNS. Per ulteriori informazioni, consulta [Stabilire una connessione basata su certificati](#) sul sito Web Apple Developer. APNs

Per gestire le APNs impostazioni utilizzando la console di gestione AWS

1. Accedi alla [console Amazon SNS](#).

2. Nel riquadro di navigazione, seleziona Notifiche push.
3. Nella sezione Applicazioni della piattaforma, seleziona l'applicazione di cui desideri modificare APNs le impostazioni, quindi scegli Modifica. Se non hai ancora creato un'applicazione di piattaforma, segui subito la [Creazione di un'applicazione della piattaforma Amazon SNS](#) guida per farlo.
4. Scegliete Modifica per modificare le impostazioni dell'applicazione della piattaforma.
5. Nella sezione Tipo di autenticazione, scegli una delle seguenti opzioni:
 - Autenticazione basata su token (consigliata per le integrazioni moderne APNs)
 - Autenticazione basata su certificati (metodo precedente)
6. Configura le tue credenziali in base al tipo di autenticazione:
 - Per l'autenticazione basata su token:
 - Carica il file.p8, che è la chiave di firma del token di autenticazione che hai scaricato dal tuo account Apple Developer.
 - Inserisci l'ID della chiave di firma che trovi nel tuo account Apple Developer. Vai su Certificati, IDsProfili, Chiavi e seleziona la chiave che desideri utilizzare.
 - Fornisci il Team Identifier dal tuo account Apple Developer. Puoi trovarlo nella pagina di iscrizione.
 - Inserisci il Bundle Identifier assegnato alla tua app. Puoi trovarlo in Certificati IDs e Profili, App. IDs
 - Per l'autenticazione basata su certificati:
 - Carica il file.p12 per il tuo certificato TLS. Questo file può essere esportato da Keychain Access su macOS dopo aver scaricato il certificato dal tuo account Apple Developer.
 - Se hai assegnato una password al tuo certificato .p12, inseriscila qui.
7. Dopo aver inserito le credenziali necessarie, scegli Salva modifiche per aggiornare le impostazioni.

Integrazione di Amazon SNS con la configurazione dell'autenticazione di Firebase Cloud Messaging

Questo argomento descrive come ottenere le credenziali API FCM (HTTP v1) richieste da Google da utilizzare con l' AWS API e il. AWS CLI Console di gestione AWS

Important

26 marzo 2024 — Amazon SNS supporta l'API HTTP v1 FCM per dispositivi Apple e destinazioni Webpush. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima API HTTP v1 di FCM entro il 1° giugno 2024 per evitare interruzioni delle applicazioni.

18 gennaio 2024 — Amazon SNS ha introdotto il supporto per l'API HTTP v1 di FCM per la consegna di notifiche push mobili ai dispositivi Android.

20 giugno 2023 — Google ha reso obsoleta la propria API HTTP legacy di Firebase Cloud Messaging (FCM). Amazon SNS ora supporta la distribuzione a tutti i tipi di dispositivi utilizzando l'API HTTP v1 di FCM. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima API HTTP v1 di FCM entro il 1° giugno 2024 per evitare interruzioni.

Puoi autorizzare Amazon SNS a inviare notifiche push alle applicazioni fornendo informazioni che ti identificano come sviluppatore dell'app. Per l'autenticazione, fornisci una chiave API o un token [durante la creazione di un'applicazione di piattaforma](#). [Puoi ottenere le seguenti informazioni dalla console dell'applicazione Firebase](#):

Chiave API

La chiave API rappresenta le credenziali utilizzate per chiamare l'API Legacy di Firebase. FCM Legacy APIs verrà rimosso da Google il 20 giugno 2024. Se attualmente utilizzi una chiave API come credenziali della piattaforma, puoi aggiornare le credenziali della piattaforma selezionando Token come opzione e caricando il file JSON associato all'applicazione Firebase.

Token

Quando si chiama l'API HTTP v1, viene utilizzato un token di accesso di breve durata. Questa è l'API consigliata da Firebase per l'invio di notifiche push. Per generare token di accesso, Firebase fornisce agli sviluppatori un set di credenziali sotto forma di file di chiave privata (noto anche come file service.json).

Prerequisito

Prima di poter iniziare a gestire le impostazioni FCM in Amazon SNS, devi ottenere le credenziali FCM service.json. Per ottenere le tue credenziali service.json, consulta [Migrare dalla versione precedente di FCM APIs](#) a HTTP v1 nella documentazione di Google Firebase.

Gestione delle impostazioni FCM tramite la CLI

Puoi creare notifiche push FCM utilizzando l' AWS API. Il numero e le dimensioni delle risorse Amazon SNS in un AWS account sono limitati. Per ulteriori informazioni, consulta gli [endpoint e le quote di Amazon Simple Notification Service](#) nella Riferimenti generali di AWS Guida.

Per creare una notifica push FCM insieme a un argomento AWS (API) di Amazon SNS

Quando utilizzi le credenziali chiave, PlatformCredential è API key. Quando utilizzi le credenziali token, PlatformCredential è un file di chiavi private in formato JSON:

- [CreatePlatformApplication](#)

Per recuperare un tipo di credenziale FCM per un argomento (API) di Amazon SNS esistente AWS

Recupera il tipo di credenziali, "AuthenticationMethod": "Token" o
"AuthenticationMethod": "Key":

- [GetPlatformApplicationAttributes](#)

Per impostare un attributo FCM per un argomento AWS (API) di Amazon SNS esistente

Configura l'attributo FCM:

- [SetPlatformApplicationAttributes](#)

Gestione delle impostazioni FCM tramite la console

È possibile creare notifiche push FCM utilizzando AWS Command Line Interface (CLI). Il numero e le dimensioni delle risorse Amazon SNS in un AWS account sono limitati. Per ulteriori informazioni, consulta [Endpoint e quote di Amazon Simple Notification Service](#).

Creazione di una notifica push FCM insieme a un argomento SNS (AWS CLI)

Quando utilizzi le credenziali chiave, PlatformCredential è API key. Quando utilizzi le credenziali token, PlatformCredential è un file di chiavi private in formato JSON. Quando si utilizza la AWS CLI, il file deve essere in formato stringa e i caratteri speciali devono essere ignorati. Per formattare correttamente il file, Amazon SNS consiglia di utilizzare il seguente comando:
SERVICE_JSON=`jq @json <<< cat service.json`

- [create-platform-application](#)

Recupero di un tipo di credenziali FCM per un argomento Amazon SNS esistente (AWS CLI)

Recupera il tipo di credenziali, "AuthenticationMethod": "Token" o "AuthenticationMethod": "Key":

- [get-platform-application-attributes](#)

Configurazione di un attributo FCM per un argomento Amazon SNS esistente (AWS CLI)

Configura l'attributo FCM:

- [set-platform-application-attributes](#)

Gestione delle impostazioni FCM (console)

Utilizza i seguenti passaggi per inserire e gestire le tue credenziali Firebase Cloud Messaging (FCM) in Amazon SNS.

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, seleziona Notifiche push.
3. Nella sezione Applicazioni della piattaforma, seleziona l'applicazione della piattaforma FCM di cui desideri modificare le credenziali, quindi scegli Modifica.
4. Nella sezione Firebase Cloud Messaging Credentials, scegli una delle seguenti opzioni:
 - Autenticazione basata su token (metodo consigliato): carica il file della chiave privata (JSON) che hai scaricato dalla console Firebase. Questo file contiene le credenziali necessarie per generare token di accesso di breve durata per le notifiche FCM. Per ottenere questo file:
 1. Vai alla console dell'[applicazione Firebase](#).
 2. Nelle Impostazioni del progetto, seleziona Cloud Messaging.
 3. Scarica il file JSON con chiave privata (da utilizzare nel metodo di autenticazione basato su token).
 - Autenticazione con chiave API: se preferisci utilizzare il vecchio metodo di autenticazione con chiave API, inserisci la chiave API di Google nel campo fornito. Per ottenere questo file:
 1. Vai alla console dell'[applicazione Firebase](#).

2. In Impostazioni del progetto, seleziona Cloud Messaging.
3. Copia la chiave del server (chiave API) da utilizzare per l'invio di notifiche.
5. Al termine, scegliere Save changes (Salva modifiche).

Argomenti correlati

- [Utilizzo dei payload di Google Firebase Cloud Messaging v1 in Amazon SNS](#)

Gestione Amazon SNS degli endpoint Firebase Cloud Messaging

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'ARN endpoint Amazon SNS 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 con Amazon SNS come endpoint della piattaforma. Ti consigliamo di archiviare il token del dispositivo, l'ARN dell'endpoint della piattaforma Amazon SNS e il timestamp salvandoli sul tuo server delle applicazioni o su un altro archivio persistente. Per configurare l'applicazione FCM per recuperare e archiviare i token del dispositivo, consulta [Recuperare e archiviare i token di registrazione nella documentazione di Google su Firebase](#).

È importante mantenere i token. up-to-date I token del dispositivo dell'utente possono cambiare nelle seguenti condizioni:

1. L'applicazione mobile viene ripristinata su un nuovo dispositivo.
2. L'utente disinstalla o aggiorna l'applicazione.
3. L'utente cancella i dati dell'applicazione.

Quando il token del dispositivo cambia, ti consigliamo di aggiornare l'endpoint Amazon SNS corrispondente con il nuovo token. Ciò consente ad Amazon SNS di continuare la comunicazione con il dispositivo registrato. Puoi farlo implementando il seguente pseudo codice all'interno della tua applicazione mobile. Descrive una pratica consigliata per la creazione e la manutenzione degli endpoint della piattaforma abilitati. Questo approccio può essere eseguito ogni volta che le applicazioni mobili vengono avviate o come processo pianificato in background.

Pseudocodice

Utilizza il seguente pseudo codice FCM per gestire e mantenere i token dei dispositivi.

```
retrieve the latest token from the mobile OS
if (endpoint arn not stored)
    # first time registration
    call CreatePlatformEndpoint
    store returned endpoint arn
endif

call GetEndpointAttributes on the endpoint arn

if (getting attributes encountered NotFound exception)
    #endpoint was deleted
    call CreatePlatformEndpoint
    store returned endpoint arn
else
    if (token in endpoint does not match latest) or
        (GetEndpointAttributes shows endpoint as disabled)
        call SetEndpointAttributes to set the
            latest token and enable the endpoint
    endif
endif
```

Per ulteriori informazioni sui requisiti di aggiornamento dei token, consulta [Aggiornare i token regolarmente nella documentazione di Google su Firebase](#).

Rilevamento di token non validi

Quando un messaggio viene inviato a un endpoint FCM v1 con un token di dispositivo non valido, Amazon SNS riceverà una delle seguenti eccezioni:

- UNREGISTERED(HTTP 404) — Quando Amazon SNS riceve questa eccezione, riceverai un evento di errore `FailureType` di `InvalidPlatformToken` consegna con un token di piattaforma

FailureMessage of e uno of associato all'endpoint non valido. Amazon SNS disabiliterà l'endpoint della piattaforma quando una consegna fallisce, con questa eccezione.

- **INVALID_ARGUMENT(HTTP 400)** — Quando Amazon SNS riceve questa eccezione, significa che il token del dispositivo o il payload del messaggio non sono validi. Per ulteriori informazioni, consulta la documentazione di Google su [ErrorCode](#) Firebase.

Poiché **INVALID_ARGUMENT** può essere restituito in entrambi i casi, Amazon SNS restituirà un corpo *FailureType* di *InvalidNotification* notifica e uno *FailureMessage* di non è valido. Quando ricevi questo errore, verifica che il payload sia corretto. Se è corretto, verifica che il token del dispositivo lo sia up-to-date. Amazon SNS non disattiverà l'endpoint della piattaforma quando una consegna fallisce, con questa eccezione.

Un altro caso in cui si verificherà un errore di *InvalidPlatformToken* consegna è quando il token del dispositivo registrato non appartiene all'applicazione che tenta di inviare il messaggio. In questo caso, Google restituirà un errore **SENDER_ID_MISMATCH**. Amazon SNS disabiliterà l'endpoint della piattaforma quando una consegna fallisce, con questa eccezione.

Tutti i codici di errore rilevati ricevuti dall'API FCM v1 sono disponibili CloudWatch quando configuri la [registrazione dello stato di consegna](#) per la tua applicazione.

Per ricevere gli eventi di consegna della tua applicazione, consulta. [Eventi applicazione disponibili](#)

Rimuovere i token obsoleti

I token vengono considerati obsoleti una volta che il recapito dei messaggi al dispositivo endpoint inizia a fallire. Amazon SNS imposta questi token obsoleti come endpoint disabilitati per l'applicazione della tua piattaforma. Quando pubblichi su un endpoint disabilitato, Amazon SNS restituirà *EventDeliveryFailure* un evento con *FailureType* *EndpointDisabled* of e *FailureMessage* un endpoint disabilitato. Per ricevere eventi di consegna per la tua applicazione, consulta. [Eventi applicazione disponibili](#)

Quando ricevi questo errore da Amazon SNS, devi rimuovere o aggiornare il token obsoleto nell'applicazione della tua piattaforma.

Utilizzo di Amazon SNS per notifiche push su dispositivi mobili

Questa sezione descrive come inviare un messaggio di notifica push.

Pubblicazione in un argomento

Puoi anche usare Amazon SNS per inviare i messaggi agli endpoint mobili che hanno effettuato la sottoscrizione a un argomento. Il concetto è lo stesso della sottoscrizione di altri tipi di endpoint, ad esempio Amazon SQS, HTTP/S, e-mail e SMS, a un argomento, come descritto in [Che cos'è Amazon SNS?](#). La differenza è che Amazon SNS comunica attraverso servizi di notifica come Apple Push Notification Service (APNS) e Google Firebase Cloud Messaging (FCM). Attraverso i servizi di notifica, gli endpoint mobili sottoscritti ricevono le notifiche inviate all'argomento.

Messaggistica diretta per dispositivi mobili Amazon SNS

Invia i messaggi di notifica push Amazon SNS direttamente a un endpoint che rappresenta un'applicazione su un dispositivo mobile.

Per inviare un messaggio diretto

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, scegli Push notifications (Notifiche push).
3. Nella pagina delle notifiche push per dispositivi mobili, nella sezione Applicazioni della piattaforma, scegli, ad esempio **MyApp**, il nome dell'applicazione.
4. Nella **MyApp** pagina, nella sezione Endpoints, scegli un endpoint, quindi scegli Pubblica messaggio.
5. Nella pagina Publish message to endpoint (Pubblica messaggio a un endpoint), inserisci il messaggio da visualizzare nell'applicazione sul dispositivo mobile e quindi scegli Publish message (Pubblica messaggio).

Amazon SNS invia il messaggio di notifica al servizio di notifica della piattaforma che, a sua volta, invia il messaggio all'applicazione.

Pubblicazione di notifiche Amazon SNS con payload specifici della piattaforma

Puoi utilizzare Amazon SNS APIs per inviare messaggi personalizzati con payload specifici della piattaforma ai dispositivi mobili. Console di gestione AWS Per informazioni sull'uso di Amazon SNS APIs, consulta [Operazioni API push per dispositivi mobili](#) e inserisci il SNSMobilePush.java file. [snsmobilepush.zip](#)

Invio di messaggi in formato JSON

Quando si inviano payload specifici della piattaforma, i dati devono essere stringhe coppia chiave-valore in formato JSON, con virgolette doppie con carattere di escape.

I seguenti esempi mostrano un messaggio personalizzato per la piattaforma FCM .

```
{
  "GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
    \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
```

Invio di messaggi specifici della piattaforma

Oltre a inviare dati personalizzati come coppie chiave-valore, è possibile inviare coppie chiave-valore specifiche per la piattaforma.

L'esempio seguente mostra l'inclusione dei parametri FCM `time_to_live` e `collapse_key` dopo le coppie chiave-valore dei dati personalizzate nel parametro FCM `data`.

```
{
  "GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
    \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\": {\"time_to_live
    \": 3600, \"collapse_key\": \"deals\"}}}}"
```

Per l'elenco delle coppie chiave-valore supportate in ciascuno dei servizi di notifica push supportati in Amazon SNS, consulta i collegamenti seguenti:

Important

Amazon SNS ora supporta l'API HTTP v1 Firebase Cloud Messaging (FCM) per l'invio di notifiche push mobili ai dispositivi Android.

26 marzo 2024 — Amazon SNS supporta l'API HTTP v1 FCM per dispositivi Apple e destinazioni Webpush. Ti consigliamo di migrare le tue applicazioni push mobili esistenti all'ultima API HTTP v1 di FCM entro il 1° giugno 2024 per evitare interruzioni delle applicazioni.

- [Riferimento chiave di Payload](#) nella documentazione APNs

- [Firebase Cloud Messaging HTTP Protocol \(Protocollo HTTP di Firebase Cloud Messaging\)](#) nella documentazione .
- [Invio di un messaggio](#) nella documentazione

Invio di messaggi a un'applicazione su più piattaforme

Per inviare un messaggio a un'applicazione installata su dispositivi per più piattaforme, come FCM e APNs, devi prima sottoscrivere gli endpoint mobili a un argomento in Amazon SNS e quindi pubblicare il messaggio sull'argomento.

L'esempio seguente mostra un messaggio da inviare agli endpoint mobili sottoscritti su APNs 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 di avviso o in APNs background

Amazon SNS può inviare messaggi ad APNs 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.

Specificazione di valori di intestazione personalizzati APNs

Ti consigliamo di specificare valori personalizzati per l'[attributo del messaggio AWS.SNS.MOBILE.APNS.PUSH_TYPE riservato](#) utilizzando l'azione dell'API Amazon Publish SNS

o AWS SDKs il. AWS CLI Il seguente esempio di CLI imposta `content-available` su 1 e `apns-push-type` su `background` per l'argomento specificato.

```
aws sns publish \
--endpoint-url https://sns.us-east-1.amazonaws.com \
--target-arn arn:aws:sns:us-east-1:123456789012:endpoint/APNS_PLATFORM/MYAPP/1234a567-
bc89-012d-3e45-6fg7h890123i \
--message '{"APNS_PLATFORM":{"aps":{"content-available":1}}}' \
--message-attributes '{ \
  "AWS.SNS.MOBILE.APNS.TOPIC":
{"DataType":"String","StringValue":"com.amazon.mobile.messaging.myapp"}, \
  "AWS.SNS.MOBILE.APNS.PUSH_TYPE":{"DataType":"String","StringValue":"background"}, \
  "AWS.SNS.MOBILE.APNS.PRIORITY":{"DataType":"String","StringValue":"5"}}' \
--message-structure json
```

Note

Assicurati che la struttura JSON 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` APNs intestazione, Amazon SNS imposta l'intestazione su `background` o in base `alert` alla chiave nel dizionario della configurazione `content-available` del payload in `aps` formato JSON. APNs

Note

Amazon SNS è in grado di dedurre solo le intestazioni `alert` o `background`, anche se l'intestazione `apns-push-type` può essere impostata su altri valori.

- `apns-push-type` è impostato su `alert`.
 - Se il dizionario `aps` contiene `content-available` impostato su 1 e una o più chiavi che attivano le interazioni dell'utente.
 - Se il dizionario `aps` contiene `content-available` impostato su 0 o se la chiave `content-available` è assente.
 - Se il valore della chiave `content-available` non è un numero intero o un valore booleano.

- `apns-push-type` è impostato su `background`.
- Se il dizionario `aps` contiene solo `content-available` impostato su 1 e nessun'altra chiave che attiva le interazioni dell'utente.

Important

Se Amazon SNS invia un oggetto di configurazione non elaborato APNs come notifica solo in background, devi includere `content-available` set to nel dizionario. 1 `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` APNs intestazione del messaggio su `background`. Quando Amazon SNS rileva che il dizionario `apn` contiene la chiave `content-available` impostata su 1 e non contiene altre chiavi che possono attivare interazioni utente imposta l'intestazione su `background`.

Utilizzo dei payload di Google Firebase Cloud Messaging v1 in Amazon SNS

Amazon SNS supporta l'utilizzo dell'API HTTP v1 di FCM per inviare notifiche a destinazioni Android, iOS e Webpush. Questo argomento fornisce esempi della struttura del payload durante la pubblicazione di notifiche push per dispositivi mobili utilizzando la CLI o l'API Amazon SNS.

Puoi includere i seguenti tipi di messaggi nel tuo payload quando invii una notifica FCM:

- **Messaggio di dati:** un messaggio di dati viene gestito dall'app client e contiene coppie chiave-valore personalizzate. Quando si crea un messaggio di dati, è necessario includere la data a chiave con un oggetto JSON come valore, quindi inserire le coppie chiave-valore personalizzate.
- **Messaggio di notifica o messaggio visualizzato:** un messaggio di notifica contiene un set predefinito di chiavi gestite da FCM SDK. Queste chiavi variano a seconda del tipo di dispositivo a cui vengono consegnate. Per ulteriori informazioni sui tasti di notifica specifici della piattaforma, consulta quanto segue:

- [Tasti di notifica Android](#)
- [Chiavi di notifica APNS](#)
- [Chiavi di notifica Webpush](#)

Per ulteriori informazioni sui tipi di messaggi FCM, consulta [Tipi di messaggio nella documentazione di Firebase di Google](#).

Utilizzo della struttura di payload FCM v1 per inviare messaggi

Se state creando un'applicazione FCM per la prima volta o desiderate sfruttare le funzionalità di FCM v1, 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 da FCM legacy APIs a HTTP v1](#) e [Personalizzazione di un messaggio tra piattaforme nella documentazione Firebase di Google](#).

Payload di esempio FCM v1 inviato ad Amazon SNS:

Note

Il valore della GCM chiave utilizzato nell'esempio seguente deve essere codificato come stringa quando si pubblica una notifica tramite Amazon SNS.

```
{
  "GCM": "{
    \"fcmV1Message\": {
      \"validate_only\": false,
      \"message\": {
        \"notification\": {
          \"title\": \"string\",
          \"body\": \"string\"
        },
        \"data\": {
          \"dataGen\": \"priority message\"
        },
        \"android\": {
          \"priority\": \"high\",
          \"notification\": {
            \"body_loc_args\": [\"string\"],
            \"title_loc_args\": [\"string\"],
```

```
\sound\": \"string\",
\"title_loc_key\": \"string\",
\"title\": \"string\",
\"body\": \"string\",
\"click_action\": \"clicky_clacky\",
\"body_loc_key\": \"string\"
},
\"data\": {
  \"dataAndroid\": \"priority message\"
},
\"ttl\": \"10023.32s\"
},
\"apns\": {
  \"payload\": {
    \"aps\": {
      \"alert\": {
        \"subtitle\": \"string\",
        \"title-loc-args\": [\"string\"],
        \"title-loc-key\": \"string\",
        \"loc-args\": [\"string\"],
        \"loc-key\": \"string\",
        \"title\": \"string\",
        \"body\": \"string\"
      },
      \"category\": \"Click\",
      \"content-available\": 0,
      \"sound\": \"string\",
      \"badge\": 5
    }
  }
},
\"webpush\": {
  \"notification\": {
    \"badge\": \"5\",
    \"title\": \"string\",
    \"body\": \"string\"
  },
  \"data\": {
    \"dataWeb\": \"priority message\"
  }
}
}
```

```
}
```

Quando invii un payload JSON, assicurati di includere l'`message-structure` attributo nella richiesta e di impostarlo su `json`.

Esempio di CLI:

```
aws sns publish --topic $TOPIC_ARN --message '{"GCM": {"fcmV1Message": {"message": {"notification": {"title": "string", "body": "string"}, "android": {"priority": "high", "notification": {"title": "string", "body": "string"}, "data": {"customAndroidDataKey": "custom key value", "ttl": "0s"}, "apns": {"payload": {"aps": {"alert": {"title": "string", "body": "string"}, "content-available": 1, "badge": 5}}}, "webpush": {"notification": {"badge": "URL", "body": "Test"}, "data": {"customWebpushDataKey": "priority message"}}, "data": {"customGeneralDataKey": "priority message"}}}}, "default": {"notification": {"title": "test"}}}}' --region $REGION --message-structure json
```

Per ulteriori informazioni sull'invio di payload in formato FCM v1, consulta quanto segue nella documentazione di Firebase di Google:

- [Esegui la migrazione dalla versione precedente di FCM a HTTP v1 APIs](#)
- [Informazioni sui messaggi FCM](#)
- [Risorsa REST: `projects.messages`](#)

Utilizzo della struttura di payload legacy per inviare messaggi all'API FCM v1

Durante la migrazione a FCM v1, non è necessario modificare la struttura del payload utilizzata per le credenziali 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 le chiavi. `fcm_options` Se desideri utilizzare questi campi, invia un payload FCM v1.
- In alcuni casi, le intestazioni dei messaggi sono necessarie da FCM v1 per inviare notifiche silenziose ai 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 un elenco delle APNs intestazioni e per cosa vengono utilizzate, consulta [Communicating with APNs](#) nella Apple Developer Guide.
- Se utilizzi l'attributo TTL Amazon SNS per inviare la notifica, questo verrà aggiornato solo sul `android` campo. Se desideri impostare l'attributo TTL APNS, utilizza il payload FCM v1.

- Le webpush chiavi androidapns, e verranno mappate e compilate con tutte le chiavi pertinenti fornite. Ad esempio, se fornisci title una chiave condivisa tra tutte e tre le piattaforme, la mappatura FCM v1 popolerà tutte e tre le piattaforme con il titolo che hai fornito.
- Alcune chiavi condivise tra piattaforme prevedono tipi di valori diversi. Ad esempio, la badge chiave passata a apns prevede un valore intero, mentre la badge chiave passata a webpush prevede un valore String. Nei casi in cui si fornisce la badge chiave, la mappatura FCM v1 popolerà solo la chiave per la quale è stato fornito un valore valido.

Eventi di mancata consegna da parte di FCM

La tabella seguente fornisce il tipo di errore di Amazon SNS che corrisponde ai codici di errore/stato ricevuti da Google per le richieste di notifica FCM v1. [Tutti i codici di errore osservati ricevuti dall'API FCM v1 sono disponibili CloudWatch quando configuri la registrazione dello stato di consegna per la tua applicazione.](#)

error/status Codice FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
UNREGISTERED	InvalidPlatformToken	Il token di piattaforma associato all'endpoint non è valido.	Il token del dispositivo collegato all'endpoint è obsoleto o non valido. Amazon SNS ha disabilitato il tuo endpoint. Aggiorna l'endpoint Amazon SNS al token del dispositivo più recente.
INVALID_ARGUMENT	InvalidNotification	Il corpo della notifica non è valido.	Il token del dispositivo o il payload del messaggio potrebbero non essere validi. Verifica che il payload del messaggio sia valido. Se il payload del messaggio è

error/status Codice FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
			valido, aggiorna l'endpoint Amazon SNS al token del dispositivo più recente.
SENDER_ID_MISMATCH	InvalidPlatformToken	Il token della piattaforma associato all'endpoint non è valido.	L'applicazione della piattaforma associata al token del dispositivo non dispone dell'autorizzazione per l'invio al token del dispositivo. Verifica di utilizzare le credenziali FCM corrette nell'applicazione della piattaforma Amazon SNS.
UNAVAILABLE	DependencyUnavailable	La dipendenza non è disponibile.	FCM non è riuscito a elaborare la richiesta in tempo. Tutti i nuovi tentativi eseguiti da Amazon SNS non sono riusciti. Puoi archiviare questi messaggi in una coda di lettere morte (DLQ) e reindirizzarli in un secondo momento.

error/status Codice FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
INTERNAL	UnexpectedFailure	Errore imprevisto; contattata Amazon. Frase di errore [Errore interno].	Il server FCM ha riscontrato un errore durante il tentativo di elaborare la richiesta. Tutti i nuovi tentativi eseguiti da Amazon SNS non sono riusciti. Puoi archiviare questi messaggi in una coda di lettere morte (DLQ) e reindirizzarli in un secondo momento.
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	Le credenziali dell'applicazione della piattaforma non sono valide.	Non è stato possibile inviare un messaggio indirizzato a un dispositivo iOS o a un dispositivo Webpush. Verifica che le tue credenziali di sviluppo e produzione siano valide.

error/status Codice FCM	Tipo di errore Amazon SNS	Messaggio di errore	Causa e mitigazione
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 ErrorCode documentazione di Firebase di Google.
PERMISSION_DENIED	InvalidNotification	Il corpo della notifica non è valido.	In caso di PERMISSION_DENIED eccezione, il chiamante (l'applicazione FCM) non è autorizzato a eseguire l'operazione specificata nel payload. Accedi alla console FCM e verifica che le tue credenziali abbiano abilitato le azioni API richieste.

Attributi dell'app mobile Amazon SNS

Amazon Simple Notification Service (Amazon SNS) fornisce supporto per la registrazione dello 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:

- Sapere se un messaggio di notifica push è stato consegnato da Amazon SNS al servizio di notifica push.
- Identificare la risposta inviata dal servizio di notifica push a Amazon SNS.
- Determinare il tempo di attesa dei messaggi (il periodo di tempo tra il timestamp di pubblicazione e l'istante immediatamente precedente alla consegna a un servizio di notifica push).

Per configurare gli attributi dell'applicazione per lo stato di consegna dei messaggi, puoi utilizzare i Console di gestione AWS kit di sviluppo AWS software (SDKs) o l'API di interrogazione.

Configurazione degli attributi dello stato di recapito dei messaggi utilizzando Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, scegliere Mobile (Dispositivi mobili), Push notifications (Notifiche push).
3. Nella sezione Applicazioni della piattaforma, scegli l'applicazione che contiene gli endpoint per i quali desideri ricevere CloudWatch i registri.
4. Scegli Application Actions (Operazioni applicazione), quindi scegli Delivery status (Stato consegna).
5. Nella finestra di dialogo Delivery Status (Stato consegna), scegli Create IAM Roles (Crea ruoli IAM).

Si viene reindirizzati alla console IAM.

6. Scegli Consenti per consentire ad Amazon SNS l'accesso in scrittura per utilizzare CloudWatch i log per tuo conto.
7. Ora, tornando alla finestra di dialogo Delivery Status, inserisci un numero nel campo Percentuale di successo su campione (0-100) per indicare 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 log sullo stato CloudWatch di consegna dei messaggi di Amazon SNS

Dopo aver configurato gli attributi dello stato di consegna dei messaggi per un endpoint dell'applicazione, verranno CloudWatch generati i log. Di seguito vengono forniti degli esempi di log in formato JSON:

RIUSCITO

```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
    "token": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\": [{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```

NON RIUSCITO

```
{
  "status": "FAILURE",
  "notification": {
    "timestamp": "2015-01-26 23:29:35.678",
    "messageId": "c3ad79b0-8996-550a-8bfa-24f05989898f"
  },
  "delivery": {
    "statusCode": 8,
    "dwellTimeMs": 1451,
    "token": "example29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken, id=1, cause=null)",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

Per un elenco dei codici di risposta dei servizi di notifica push, consulta [Codici di risposta della piattaforma](#).

Configurazione degli attributi dello stato di consegna dei messaggi con AWS SDKs

[AWS SDKs](#) Forniscono APIs in diverse lingue per l'utilizzo degli attributi dello stato di consegna dei messaggi con Amazon SNS.

L'esempio Java seguente mostra come utilizzare l'API `SetPlatformApplicationAttributes` per configurare attributi di applicazione per lo stato di consegna dei messaggi di notifica push. Puoi utilizzare i seguenti attributi per lo stato di consegna dei messaggi: `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn` e `SuccessFeedbackSampleRate`. `FailureFeedbackRoleArn` Gli attributi `SuccessFeedbackRoleArn` and vengono utilizzati per consentire ad Amazon SNS l'accesso in 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 log. CloudWatch

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
    SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
```

```
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

Per ulteriori informazioni su SDK per Java, consulta [Nozioni di base su AWS SDK per 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
Amazon Device Messaging (ADM)	Vedi Formato di risposta nella documentazione ADM.
Servizio di notifica push Apple () APNs	Vedi la risposta HTTP/2 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 di notifica push Microsoft per Windows Phone (MPNS)	Vedi Push Notification Service Response Codes for Windows Phone 8 nella documentazione di Windows 8 Development.
Windows Push Notification Services (WNS)	Vedi "Codici di risposta" in Intestazioni delle richieste e delle risposte per il servizio di notifica Push (app di Windows Runtime) nella documentazione di sviluppo per Windows 8.

Notifiche degli eventi delle applicazioni Amazon SNS per applicazioni mobili

Amazon SNS aiuta a capire come attivare le notifiche quando si verificano determinati eventi applicazione. Puoi intraprendere alcune azioni programmatiche su quell'evento. L'applicazione deve

includere il supporto per un servizio di notifica push come Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM) e Windows Push Notification Services (WNS). Puoi impostare le notifiche degli eventi dell'applicazione utilizzando la console Amazon SNS o AWS CLI o le AWS SDKs.

Eventi applicazione disponibili

Le notifiche eventi applicazione tengono traccia degli eventi di creazione, eliminazione e aggiornamento dei singoli endpoint della piattaforma, nonché degli errori di consegna. Di seguito sono elencati i nomi degli attributi per gli eventi applicazione.

Nome attributo	Trigger di notifica
EventEndpointCreated	Viene aggiunto un nuovo endpoint della piattaforma all'applicazione.
EventEndpointDeleted	Viene eliminato un endpoint della piattaforma associato all'applicazione.
EventEndpointUpdated	Viene modificato un attributo degli endpoint della piattaforma associati all'applicazione.
EventDeliveryFailure	Una consegna a un qualsiasi endpoint della piattaforma associato all'applicazione restituisce un errore permanente.  Note Per tenere traccia degli errori di consegna relativamente alle applicazioni della piattaforma, effettua la sottoscrizione agli eventi sullo stato di consegna dei messaggi per l'applicazione. Per ulteriori informazioni, consulta la pagina sull'utilizzo degli attributi di applicazione di Amazon SNS per lo stato di consegna dei messaggi.

È possibile associare qualsiasi attributo a un'applicazione che può quindi ricevere le notifiche di eventi.

Invio di notifiche push per dispositivi mobili

Per inviare notifiche di eventi dell'applicazione, specifica un argomento per ricevere le notifiche per ciascun tipo di evento. Come Amazon SNS invia le notifiche, l'argomento può indirizzarle agli endpoint che intraprenderanno un'azione programmatica.

Important

Le applicazioni a volume elevato creeranno un gran numero di notifiche di eventi dell'applicazione (ad esempio, decine di migliaia), che sovraccaricheranno gli endpoint destinati all'uso umano, come indirizzi e-mail, numeri di telefono e applicazioni mobili. Considera le seguenti linee guida quando invii notifiche di eventi dell'applicazione a un argomento:

- Ogni argomento che riceve notifiche deve contenere solo sottoscrizioni per endpoint programmatici, come endpoint HTTP o HTTPS, code o funzioni Amazon SQS. AWS Lambda
- Per ridurre la quantità di elaborazione attivata dalle notifiche, limitare le sottoscrizioni di ciascun argomento a un numero ridotto (ad esempio, cinque o meno).

Puoi inviare notifiche sugli eventi dell'applicazione utilizzando la console Amazon SNS, il AWS Command Line Interface (AWS CLI) o il. AWS SDKs

Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, scegli Mobile (Dispositivi mobili), Push notifications (Notifiche push).
3. Nella pagina delle notifiche push per dispositivi mobili, nella sezione Applicazioni della piattaforma, scegli un'applicazione, quindi scegli Modifica.
4. Espandere la sezione Event notifications (Notifiche evento).
5. Seleziona Actions (Azioni), Configure events (Configura eventi).
6. 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 argomento Amazon SNS per tutti e quattro gli eventi dell'applicazione:

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDKs

Imposta le notifiche degli eventi dell'applicazione inviando una `SetPlatformApplicationAttributes` richiesta con l'API Amazon SNS utilizzando AWS un SDK.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, tra cui assistenza per iniziare e informazioni sulle versioni precedenti, consulta [Utilizzo di Amazon SNS con un SDK AWS](#)

Operazioni API push per dispositivi mobili

Per utilizzare il push mobile di Amazon SNS APIs, 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 utente Amazon SNS](#).

Per inviare un messaggio di notifica push a un'app e a un dispositivo mobile utilizzando il APIs, devi prima utilizzare `CreatePlatformApplication`, 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](#).

I push mobili di Amazon SNS APIs sono i seguenti:

[CreatePlatformApplication](#)

Crea un oggetto applicativo della piattaforma per uno dei servizi di notifica push supportati, come APNs FCM, 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 è 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.

Errori comuni dell'API push mobile di Amazon SNS

Gli errori restituiti da Amazon SNS APIs per dispositivi mobili push sono elencati nella tabella seguente. Per ulteriori informazioni su Amazon SNS APIs for mobile push, consulta. [Operazioni API push per dispositivi mobili](#)

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Application Name is null string (Nome applicazione è una stringa null)	Il nome di applicazione richiesto è impostato su null.	400	CreatePlatformApplication

Errore	Descrizione	Codice di stato HTTPS	Operazione API
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 piattaforma non valido)	È stato fornito un out-of-range valore o non valido per il nome della piattaforma.	400	CreatePlatformApplication
APNs — Il principal è non è un certificato valido	È stato fornito un certificato non valido per il APNs principale, che è il certificato SSL. Per ulteriori informazioni, consulta il riferimento CreatePlatformApplication all'API di Amazon Simple Notification Service.	400	CreatePlatformApplication
APNs — Principal è un certificato valido ma non in formato.pem	Per il APNs principale, che è il certificato SSL, è stato fornito un certificato valido non in formato.pem.	400	CreatePlatformApplication
APNs — Principal è un certificato scaduto	È stato fornito un certificato scaduto per il APNs principale, che è il certificato SSL.	400	CreatePlatformApplication

Errore	Descrizione	Codice di stato HTTPS	Operazione API
APNs — Il principal e non è un certificato emesso da Apple	Un certificato non Apple è stato fornito per il principale APNs, ovvero il certificato SSL.	400	CreatePlatformApplication
APNs — Il preside non è fornito	Il APNs principal e, che è il certificato SSL, non è stato fornito.	400	CreatePlatformApplication
APNs — La credenziale non è stata fornita	La APNs credenziale, che è la chiave privata, non è stata fornita. Per ulteriori informazioni, consulta il riferimento CreatePlatformApplication all'API di Amazon Simple Notification Service.	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

Errore	Descrizione	Codice di stato HTTPS	Operazione API
FCM: il server non è fornito APIKey	La credenziale FCM, ovvero la chiave API, non è stata fornita. Per ulteriori informazioni, consulta il riferimento CreatePlatformApplication all'API di Amazon Simple Notification Service.	400	CreatePlatformApplication
FCM: il server APIKey è vuoto	La credenziale FCM, ovvero la chiave API, è vuota.	400	CreatePlatformApplication
FCM: il server APIKey è una stringa nulla	La credenziale FCM, ovvero la chiave API, è null.	400	CreatePlatformApplication
FCM: il server non è valido APIKey	La credenziale FCM, ovvero la chiave API, non è valida.	400	CreatePlatformApplication
ADM - clientsecret is not provided (ADM - clientsecret non fornita)	Il segreto client richiesto non è fornito.	400	CreatePlatformApplication
ADM - clientsecret is a null string (ADM - clientsecret è una stringa null)	La stringa richiesta per il segreto client è null.	400	CreatePlatformApplication

Errore	Descrizione	Codice di stato HTTPS	Operazione API
ADM - client_secret is empty string (ADM - client_secret è una stringa vuota)	La stringa richiesta per il segreto client è vuota.	400	CreatePlatformApplication
ADM - client_secret is not valid (ADM - client_secret non valida)	La stringa richiesta per il segreto client non è valida.	400	CreatePlatformApplication
ADM - client_id is empty string (ADM - client_id è una stringa vuota)	La stringa richiesta per l'ID client è vuota.	400	CreatePlatformApplication
ADM - clientId is not provided (ADM - clientId non fornita)	La stringa richiesta per l'ID client non è fornita.	400	CreatePlatformApplication
ADM - clientid is a null string (ADM - clientid è una stringa null)	La stringa richiesta per l'ID client è null.	400	CreatePlatformApplication
ADM - client_id is not valid (ADM - client_id non valida)	La stringa richiesta per l'ID client non è valida.	400	CreatePlatformApplication
EventEndpointCreated ha un formato ARN non valido	EventEndpointCreated ha un formato ARN non valido.	400	CreatePlatformApplication
EventEndpointDeleted ha un formato ARN non valido	EventEndpointDeleted ha un formato ARN non valido.	400	CreatePlatformApplication

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventEndpointUpdated ha un formato ARN non valido	EventEndpointUpdated ha un formato ARN non valido.	400	CreatePlatformApplication
EventDeliveryAttemptFailure ha un formato ARN non valido	EventDeliveryAttemptFailure ha un formato ARN non valido.	400	CreatePlatformApplication
EventDeliveryFailure ha un formato ARN non valido	EventDeliveryFailure ha un formato ARN non valido.	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	Codice di stato HTTPS	Operazione API
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	SetPlatformAttributes
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	400	SetPlatformAttributes
APNs — Il principal è un certificato non valido	È stato fornito un certificato non valido per il APNs principale, che è il certificato SSL. Per ulteriori informazioni, consulta il riferimento CreatePlatformApplication all'API di Amazon Simple Notification Service.	400	SetPlatformAttributes
APNs — Principal è un certificato valido ma non in formato.pem	Per il APNs principale, che è il certificato SSL, è stato fornito un certificato valido non in formato.pem.	400	SetPlatformAttributes
APNs — Principal è un certificato scaduto	È stato fornito un certificato scaduto per il APNs principale, che è il certificato SSL.	400	SetPlatformAttributes

Errore	Descrizione	Codice di stato HTTPS	Operazione API
APNs — Il principal e non è un certificato emesso da Apple	Un certificato non Apple è stato fornito per il principale APNs, ovvero il certificato SSL.	400	SetPlatformAttributes
APNs — Il preside non è fornito	Il APNs principal e, che è il certificato SSL, non è stato fornito.	400	SetPlatformAttributes
APNs — La credenziale non è stata fornita	La APNs credenziale, che è la chiave privata, non è stata fornita. Per ulteriori informazioni, consulta il riferimento CreatePlatformApplication all'API di Amazon Simple Notification Service.	400	SetPlatformAttributes
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	SetPlatformAttributes

Errore	Descrizione	Codice di stato HTTPS	Operazione API
FCM: il server non è fornito APIKey	La credenziale FCM, ovvero la chiave API, non è stata fornita. Per ulteriori informazioni, consulta il riferimento CreatePlatformApplication all'API di Amazon Simple Notification Service.	400	SetPlatformAttributes
FCM: il server APIKey è una stringa nulla	La credenziale FCM, ovvero la chiave API, è null.	400	SetPlatformAttributes
ADM - clientId is not provided (ADM - clientId non fornita)	La stringa richiesta per l'ID client non è fornita.	400	SetPlatformAttributes
ADM - clientId is a null string (ADM - clientId è una stringa null)	La stringa richiesta per l'ID client è null.	400	SetPlatformAttributes
ADM - clientsecret is not provided (ADM - clientsecret non fornita)	Il segreto client richiesto non è fornito.	400	SetPlatformAttributes
ADM - clientsecret is a null string (ADM - clientsecret è una stringa null)	La stringa richiesta per il segreto client è null.	400	SetPlatformAttributes

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventEndpointUpdated ha un formato ARN non valido	EventEndpointUpdated ha un formato ARN non valido.	400	SetPlatformAttributes
EventEndpointDeleted ha un formato ARN non valido	EventEndpointDeleted ha un formato ARN non valido.	400	SetPlatformAttributes
EventEndpointUpdated ha un formato ARN non valido	EventEndpointUpdated ha un formato ARN non valido.	400	SetPlatformAttributes
EventDeliveryAttemptFailure ha un formato ARN non valido	EventDeliveryAttemptFailure ha un formato ARN non valido.	400	SetPlatformAttributes
EventDeliveryFailure ha un formato ARN non valido	EventDeliveryFailure ha un formato ARN non valido.	400	SetPlatformAttributes
EventEndpointCreated non è un argomento esistente	EventEndpointCreated non è un argomento esistente.	400	SetPlatformAttributes
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

Errore	Descrizione	Codice di stato HTTPS	Operazione API
EventDeliveryFailure non è un argomento esistente	EventDeliveryFailure non è un argomento esistente.	400	SetPlatformAttributes
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	GetPlatformApplicationAttributes
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	403	GetPlatformApplicationAttributes
Token specified is invalid (Token specificato non valido)	Il token specificato non è valido.	400	ListPlatformApplications
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	ListEndpointsByPlatformApplication
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	404	ListEndpointsByPlatformApplication
Token specified is invalid (Token specificato non valido)	Il token specificato non è valido.	400	ListEndpointsByPlatformApplication

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	DeletePlatformApplication
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	403	DeletePlatformApplication
Platform ARN is invalid (ARN piattaforma non valido)	L'ARN della piattaforma non è valido.	400	CreatePlatformEndpoint
Platform ARN is valid but does not belong to the user (ARN piattaforma valido ma non appartiene all'utente)	L'ARN della piattaforma è valido ma non appartiene all'utente.	404	CreatePlatformEndpoint
Token is not specified (Token non specificato)	Il token non è specificato.	400	CreatePlatformEndpoint
Token is not of correct length (Lunghezza token non corretta)	La lunghezza del token non è corretta.	400	CreatePlatformEndpoint
Customer User data is too large (Dimensione dati utente cliente troppo grande)	La dimensione dei dati utente cliente non può essere superiore a 2048 byte nella codifica UTF-8.	400	CreatePlatformEndpoint

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Endpoint ARN is invalid (ARN endpoint non valido)	L'ARN dell'endpoint non è valido.	400	DeleteEndpoint
Endpoint ARN is valid but does not belong to the user (ARN endpoint valido ma non appartiene all'utente)	L'ARN dell'endpoint è valido ma non appartiene all'utente.	403	DeleteEndpoint
Endpoint ARN is invalid (ARN endpoint non valido)	L'ARN dell'endpoint non è valido.	400	SetEndpointAttributes
Endpoint ARN is valid but does not belong to the user (ARN endpoint valido ma non appartiene all'utente)	L'ARN dell'endpoint è valido ma non appartiene all'utente.	403	SetEndpointAttributes
Token is not specified (Token non specificato)	Il token non è specificato.	400	SetEndpointAttributes
Token is not of correct length (Lunghezza token non corretta)	La lunghezza del token non è corretta.	400	SetEndpointAttributes
Customer User data is too large (Dimensione dati utente cliente troppo grande)	La dimensione dei dati utente cliente non può essere superiore a 2048 byte nella codifica UTF-8.	400	SetEndpointAttributes

Errore	Descrizione	Codice di stato HTTPS	Operazione API
Endpoint ARN is invalid (ARN endpoint non valido)	L'ARN dell'endpoint non è valido.	400	GetEndpointAttributes
Endpoint ARN is valid but does not belong to the user (ARN endpoint valido ma non appartiene all'utente)	L'ARN dell'endpoint è valido ma non appartiene all'utente.	403	GetEndpointAttributes
Target ARN is invalid (ARN destinazione non valido)	L'ARN di destinazione non è valido.	400	Publish
Target ARN is valid but does not belong to the user (ARN destinazione valido ma non appartiene all'utente)	L'ARN di destinazione è valido ma non appartiene all'utente.	403	Publish
Message format is invalid (Formato messaggio non valido)	Il formato del messaggio non è valido.	400	Publish
Message size is larger than supported by protocol/end-service (Dimensione messaggio superiore a quella supportata da protocollo/servizio finale)	La dimensione del messaggio è superiore a quella supportata dal protocollo/servizio finale.	400	Publish

Utilizzo dell'attributo time to live message di Amazon SNS per le notifiche push mobili

Amazon Simple Notification Service (Amazon SNS) ti aiuta a impostare un attributo messaggio TTL (Time To Live) per messaggi di notifica push per dispositivi mobili. Questa funzionalità si aggiunge alla capacità esistente di impostare il TTL all'interno del corpo del messaggio Amazon SNS per i servizi di notifica push mobili che lo supportano, come Amazon Device Messaging (ADM) e Firebase Cloud Messaging (FCM) durante l'invio ad Android.

L'attributo di messaggio TTL è utilizzato per specificare i metadati di scadenza strutturati su un messaggio. Ciò consente di specificare il periodo di tempo a disposizione del servizio di notifica push, come Apple Push Notification Service (APNs) o FCM, per recapitare il messaggio all'endpoint. Se per qualche motivo (per esempio il dispositivo mobile è stato disattivato) il messaggio non può essere consegnato entro il TTL specificato, verrà eliminato senza ulteriori tentativi di consegna. Per specificare il TTL all'interno degli attributi del messaggio, puoi utilizzare i Console di gestione AWS kit di sviluppo AWS software (SDKs) o l'API di interrogazione.

Gli attributi di messaggio TTL riservati ai servizi di notifica push

Di seguito è riportato un elenco degli attributi dei messaggi TTL per i servizi di notifica push che puoi utilizzare per impostare quando utilizzi l'API AWS SDKs o query:

Servizio di notifiche push	Attributo di messaggio TTL
Amazon Device Messaging (ADM)	<code>AWS.SNS.MOBILE.ADM.TTL</code>
Servizio di notifica push Apple () APNs	<code>AWS.SNS.MOBILE.APNS.TTL</code>
Sandbox del servizio di notifica push Apple (APNs_SANDBOX)	<code>AWS.SNS.MOBILE.APNS_SANDBOX.TTL</code>
Baidu Cloud Push (Baidu)	<code>AWS.SNS.MOBILE.BAIDU.TTL</code>
Firebase Cloud Messaging (FCM in caso di invio ad Android)	<code>AWS.SNS.MOBILE.FCM.TTL</code>
Windows Push Notification Services (WNS)	<code>AWS.SNS.MOBILE.WNS.TTL</code>

Ogni servizio di notifica push gestisce il TTL in modo diverso. Amazon SNS offre una vista astratta del TTL su tutti i servizi di notifica push, il che rende più semplice specificare il TTL. Quando utilizzi il TTL Console di gestione AWS per specificare (in secondi), devi inserire il valore TTL una sola volta e Amazon SNS calcolerà il TTL per ciascuno dei servizi di notifica push selezionati durante la pubblicazione del messaggio.

Il TTL è relativo all'ora di pubblicazione. Prima di consegnare un messaggio di notifica push a un servizio di notifica push specifico, Amazon SNS calcola il tempo di sosta (il tempo tra il timestamp di pubblicazione e l'istante immediatamente precedente il passaggio a un servizio di notifica push) per la notifica push e passa il TTL rimanente al servizio di notifica push specifico. Se il TTL è più breve del tempo di sosta, Amazon SNS non proverà a pubblicare.

Se si specifica un TTL per un messaggio di notifica push, il valore TTL deve essere un numero intero positivo, a meno che il valore di non 0 abbia un significato specifico per il servizio di notifica push, ad esempio con APNs e FCM (quando si invia ad Android). Se il valore TTL è impostato su 0 e il servizio di notifica push non ha un significato specifico per 0, Amazon SNS non consegnerà il messaggio. [Per ulteriori informazioni sul parametro TTL impostato su 0 quando si utilizza APNs, vedere la Tabella A-3 Identificatori di elementi per le notifiche remote nella documentazione dell'API Binary Provider.](#)

Ordine di precedenza per determinare il TTL

La precedenza che Amazon SNS utilizza per determinare il TTL per un messaggio di notifica push è basata sul seguente ordine, in cui il numero più basso ha la priorità più alta:

1. TTL attributo di messaggio
2. TTL testo messaggio
3. TTL di default per il servizio di notifica push (varia in base al servizio)
4. TTL di default Amazon SNS (4 settimane)

Se imposti diversi valori TTL (uno negli attributi di messaggio e un altro nel corpo del messaggio) per lo stesso messaggio, Amazon SNS modificherà il TTL nel corpo del messaggio in modo che corrisponda al TTL specificato nell'attributo del messaggio stesso.

Specificare il TTL utilizzando Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione, scegli Mobile (Dispositivi mobili), Push notifications (Notifiche push).

3. Nella pagina Mobile push notifications (Notifiche push per dispositivi mobili), nella sezione Platform applications (Applicazioni di piattaforma), selezionare un'applicazione.
4. Nella **MyApplication** pagina, nella sezione Endpoints, scegli un endpoint dell'applicazione, quindi scegli Pubblica messaggio.
5. Nella sezione Message details (Dettagli messaggio), immettere il TTL (numero di secondi a disposizione del servizio di notifica push per recapitare il messaggio all'endpoint).
6. Seleziona Publish message (Pubblica messaggio).

Regioni supportate dall'applicazione mobile Amazon SNS

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 Pacifico (Osaka)
- Asia Pacifico (Seoul)
- Asia Pacifico (Singapore)
- Asia Pacifico (Sydney)
- Asia Pacifico (Tokyo)
- Canada (Centrale)
- Europa (Francoforte)
- Europa (Irlanda)
- Europa (Londra)
- Europa (Milano)
- Europa (Spagna)
- Europa (Parigi)

- Europa (Stoccolma)
- Medio Oriente (Bahrein)
- Medio Oriente (Emirati Arabi Uniti)
- Sud America (San Paolo)
- AWS GovCloud (Stati Uniti occidentali)

Best practice per la gestione delle notifiche push mobili di Amazon SNS

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'API idempotente [CreatePlatformEndpoint](#). Tuttavia, ciò può introdurre duplicati per lo stesso dispositivo nei casi in cui il token stesso non è valido o se l'endpoint è valido ma disabilitato (ad esempio, una mancata corrispondenza tra gli ambienti di produzione e sandbox).

Può essere utilizzato un meccanismo di gestione dei token del dispositivo, come quello nello [pseudocodice](#).

Per informazioni sulla gestione e la manutenzione dei token dei dispositivi FCM v1, vedere. [Gestione Amazon SNS 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 di consegna per l'applicazione della piattaforma Amazon SNS. In questo modo è possibile risolvere i problemi di consegna, poiché i log contengono [codici di risposta](#) di provider restituiti dal servizio push della piattaforma. Per informazioni dettagliate sull'abilitazione della registrazione dello stato di consegna, vedere [Come posso accedere ai registri di consegna degli argomenti Amazon SNS per le notifiche push?](#)

Notifiche degli eventi

Per gestire gli endpoint in modo che siano basati su eventi, è possibile utilizzare la funzionalità [Notifiche eventi](#). Ciò consente all'argomento Amazon SNS configurato di eseguire il fanout degli eventi agli abbonati, come una funzione Lambda, per eventi applicativi della piattaforma di creazione, eliminazione, aggiornamenti e errori di consegna degli endpoint.

Configurazione e gestione dell'abbonamento e-mail Amazon SNS

Puoi iscrivere un [indirizzo e-mail](#) a un argomento di Amazon SNS utilizzando Console di gestione AWS, AWS SDK per Java, o. AWS SDK per .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 [quote di Amazon SNS](#).

Important

- Per impedire ai destinatari della mailing list di annullare la sottoscrizione di tutti i destinatari dalle e-mail dell'argomento Amazon SNS, consulta [Configurare un abbonamento e-mail che richiede l'autenticazione per annullare l'iscrizione](#) da AWS Support.
- Il superamento del limite di 10 messaggi al secondo (TPS) per un endpoint e-mail o email-json attiverà una sospensione automatica dell'abbonamento, che passerà allo stato di conferma in sospeso. L'abbonamento rimane attivo per 30 giorni, dopodiché verrà automaticamente eliminato a meno che non venga intrapresa alcuna azione. PendingConfirmation
- Se un indirizzo e-mail sottoscritto risulta respinto, l'indirizzo viene eliminato da ulteriori consegne per 7 giorni. I rimbalzi possono verificarsi per vari motivi, tra cui indirizzi non validi o problemi con il server di posta elettronica ricevente. Per rimuovere l'indirizzo

dall'elenco di soppressione prima della scadenza del periodo di 7 giorni, risolvi il problema sottostante con l'indirizzo e-mail, quindi contatta tramite AWS il Support Center.

Sottoscrizione di un indirizzo e-mail a un argomento di Amazon SNS utilizzando il Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione sinistro scegli Sottoscrizioni.
3. Nella pagina Sottoscrizioni scegli Crea sottoscrizione.
4. Nella pagina Crea sottoscrizione, nella sezione Dettagli, eseguire queste operazioni:
 - a. Per ARN argomento scegli l'Amazon Resource Name (ARN) di un argomento.
 - b. Per Protocol, scegliere Email.
 - c. Per Endpoint, immettere l'indirizzo e-mail.
 - d. (Facoltativo) Per configurare un criterio filtro, espandere la sezione Policy di filtro per sottoscrizione. Per ulteriori informazioni, consulta [Policy di filtro degli abbonamenti Amazon SNS](#).
 - e. (Facoltativo) Per abilitare il filtro basato sul payload, imposta Filter Policy Scope su MessageBody. Per ulteriori informazioni, consulta [Ambito delle policy di filtro per le sottoscrizioni Amazon SNS](#).
 - f. (Facoltativo) Per configurare una coda DLQ per la sottoscrizione, espandere la sezione Policy di redrive (coda DLQ). Per ulteriori informazioni, consulta [Code di lettere non ricevute di Amazon SNS](#).
 - g. Scegli Create Subscription (Crea sottoscrizione).

La console crea la sottoscrizione e apre la pagina Dettagli.

È necessario confermare la sottoscrizione prima che l'indirizzo e-mail possa iniziare a ricevere messaggi.

Per confermare una sottoscrizione

1. Controlla la tua casella di posta elettronica e scegli Conferma sottoscrizione nell'e-mail di Amazon SNS.

2. Amazon SNS apre il browser Web e visualizza una conferma dell'abbonamento con il tuo ID abbonamento.

Sottoscrizione di un indirizzo e-mail a un argomento Amazon SNS utilizzando un SDK AWS

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

Gli esempi di codice seguenti mostrano come utilizzare `Subscribe`.

.NET

SDK per .NET

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](#).

Sottoscrive un indirizzo e-mail a un argomento.

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
```

```

        Protocol = "email",
        Endpoint = "recipient@example.com",
    };

    var response = await client.SubscribeAsync(request);

    return response;
}

```

Iscrivi una coda a un argomento con filtri opzionali.

```

/// <summary>
/// Subscribe a queue to a topic with optional filters.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
/// <param name="queueArn">The ARN of the queue.</param>
/// <returns>The ARN of the new subscription.</returns>
public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
{
    var subscribeRequest = new SubscribeRequest()
    {
        TopicArn = topicArn,
        Protocol = "sqs",
        Endpoint = queueArn
    };

    if (!string.IsNullOrEmpty(filterPolicy))
    {
        subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
    }

    var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
    return subscribeResponse.SubscriptionArn;
}

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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();
}

```

Sottoscrive 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();
}

```

Sottoscrive 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();
}

```

Sottoscrive una coda SQS a un argomento.

```

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

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

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);

    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
                    << "' has been subscribed to the topic '"
                    << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                    << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,

```

```

        sqsClient);

    return false;
}

```

Sottoscrive con un filtro a un argomento.

```

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

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

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

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have
filters."
                        << std::endl;
            std::cout
                << "If you add a filter to this subscription, then
only the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                << std::endl;
            std::cout << "For this example, you can filter messages by a
\""
                        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
        }

        std::ostringstream ostream;
        ostream << "Filter messages for \"" << queueName

```

```

        << "\"'s subscription to the topic \""
        << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostringstream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

            std::cout << "This is the filter policy for this
subscription."
                        << std::endl;
            std::cout << jsonPolicy << std::endl;

            request.AddAttributes("FilterPolicy", jsonPolicy);
        }
        else {
            std::cout
                << "Because you did not select any attributes, no
filter "
                << "will be added to this subscription." <<
std::endl;
        }
    } // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
                    << "' has been subscribed to the topic '"
                    << "\"" << topicName << "\"" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                    << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
}

```



```

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

        return false;
    }

    //! Routine that lets the user select attributes for a subscription filter
    policy.
    /*!
    \sa getFilterPolicyFromUser()
    \return Aws::String: The filter policy as JSON.
    */
    Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
        std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

        std::vector<Aws::String> filterSelections;
        int selection;
        do {
            for (size_t j = 0; j < TONES.size(); ++j) {
                std::cout << "  " << (j + 1) << ". " << TONES[j]
                    << std::endl;
            }
            selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

            if (selection != 0) {
                const Aws::String &selectedTone(TONES[selection - 1]);
                // Add the tone to the selection if it is not already added.
                if (std::find(filterSelections.begin(),
                    filterSelections.end(),
                    selectedTone)
                    == filterSelections.end()) {
                    filterSelections.push_back(selectedTone);
                }
            }
        } while (selection != 0);

        Aws::String result;
    }

```

```

    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"\" << TONE_ATTRIBUTE << "\": [";

        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

CLI

AWS CLI

Effettuare la sottoscrizione a un argomento

Attraverso il comando `subscribe` seguente viene effettuata la sottoscrizione all'argomento specificato utilizzando un indirizzo e-mail.

```

aws sns subscribe \
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
  --protocol email \
  --notification-endpoint my-email@example.com

```

Output:

```

{
  "SubscriptionArn": "pending confirmation"
}

```

```
}
```

- Per informazioni dettagliate sull'API, consulta [Subscribe](#) nella documentazione di riferimento dei comandi della AWS CLI .

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Iscrivi una coda a un argomento con filtri opzionali.

```
import (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sns"  
    "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// 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 subscribe queue %v to topic %v. Here's why: %v\n",
            queueArn, topicArn, err)
    } else {
        subscriptionArn = *output.SubscriptionArn
    }

    return subscriptionArn, err
}
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Go .

Java

SDK for Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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);
        }
    }
}

```

Sottoscrive un endpoint HTTP a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.

```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <url>

            Where:
                topicArn - The ARN of the topic to subscribe.
                url - The HTTPS endpoint that you want to receive
notifications.

            """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
    {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("https")
                .endpoint(url)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
        }
    }
}
```

```

        System.out.println("Subscription ARN is " + result.subscriptionArn()
+ "\n\n Status is "
        + result.sdkHttpResponse().statusCode());

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

```

Sottoscrive una funzione Lambda a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            "";

        if (args.length != 2) {

```



```
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    String lambdaArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    String arnValue = subLambda(snsClient, topicArn, lambdaArn);
    System.out.println("Subscription ARN: " + arnValue);
    snsClient.close();
}

public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("lambda")
            .endpoint(lambdaArn)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        return result.subscriptionArn();

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

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 * a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 * topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "user@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'
// }
};

```

Sottoscrive 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;
};
```

Sottoscrive 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;
};
```

Sottoscrive una coda SQS a un argomento.

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

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
  test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx'
  // }
  return response;
};
```

Sottoscrive con un filtro a un argomento.

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

```

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
      // set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
  // test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx'
  // }
  return response;
};

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per JavaScript .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail a un argomento.

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

Sottoscrive 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento degli SDK AWS per Kotlin.

PHP

SDK for PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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());
    }
}

```

Sottoscrive un endpoint HTTP a un argomento.

```

require 'vendor/autoload.php';

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

/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 * message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',

```

```

        'version' => '2010-03-31'
    ]);

    $protocol = 'https';
    $endpoint = 'https://';
    $topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

    try {
        $result = $SnSClient->subscribe([
            'Protocol' => $protocol,
            'Endpoint' => $endpoint,
            'ReturnSubscriptionArn' => true,
            'TopicArn' => $topic,
        ]);
        var_dump($result);
    } catch (AwsException $e) {
        // output error message if fails
        error_log($e->getMessage());
    }
}

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

Python

SDK for Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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

```

Iscrivi una coda a un argomento con filtri opzionali.

```
class SnsWrapper:
```

```
"""Wrapper class for managing Amazon SNS operations."""

def __init__(self, sns_client: Any) -> None:
    """
    Initialize the SnsWrapper.

    :param sns_client: A Boto3 Amazon SNS client.
    """
    self.sns_client = sns_client

@classmethod
def from_client(cls) -> 'SnsWrapper':
    """
    Create an SnsWrapper instance using a default boto3 client.

    :return: An instance of this class.
    """
    sns_client = boto3.client('sns')
    return cls(sns_client)

def subscribe_queue_to_topic(
    self,
    topic_arn: str,
    queue_arn: str,
    filter_policy: Optional[str] = None
) -> str:
    """
    Subscribe an SQS queue to an SNS topic.

    :param topic_arn: The ARN of the SNS topic.
    :param queue_arn: The ARN of the SQS queue.
    :param filter_policy: Optional JSON filter policy for message filtering.
    :return: The ARN of the subscription.
    :raises ClientError: If the subscription fails.
    """
    try:
        attributes = {}
        if filter_policy:
            attributes['FilterPolicy'] = filter_policy

        response = self.sns_client.subscribe(
            TopicArn=topic_arn,
            Protocol='sqs',
```

```

        Endpoint=queue_arn,
        Attributes=attributes
    )

    subscription_arn = response['SubscriptionArn']
    logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
    return subscription_arn

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
    raise

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API dell'SDK for Python (Boto3)AWS .

Ruby

SDK for Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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
  end
end

```

```

    @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
end

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

Rust

SDK per 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](#).

Sottoscrive un indirizzo e-mail a un argomento.

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

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

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

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

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

    Ok(())
}
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per 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](#).

Sottoscrive un indirizzo e-mail 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_snssubscriptionlmt00.  
        MESSAGE 'Unable to create subscriptions. You have reached the maximum  
number of subscriptions allowed.' TYPE 'E'.  
ENDTRY.
```

- Per informazioni dettagliate sull'API, consulta [Subscribe](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail a un argomento.

```
import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

let output = try await snsClient.subscribe(
    input: SubscribeInput(
        endpoint: email,
        protocol: "email",
        returnSubscriptionArn: true,
        topicArn: arn
    )
)

guard let subscriptionArn = output.subscriptionArn else {
    print("No subscription ARN received from Amazon SNS.")
    return
}

print("Subscription \(subscriptionArn) created.")
```

Per sottoscrivere un numero di telefono a un argomento per ricevere le notifiche via SMS.

```
import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

let output = try await snsClient.subscribe(
```

```
        input: SubscribeInput(
            endpoint: phone,
            protocol: "sms",
            returnSubscriptionArn: true,
            topicArn: arn
        )
    )

    guard let subscriptionArn = output.subscriptionArn else {
        print("No subscription ARN received from Amazon SNS.")
        return
    }

    print("Subscription \(subscriptionArn) created.")
```

- Per informazioni dettagliate sull'API, consulta [Subscribe](#) nella documentazione di riferimento dell'API AWS SDK per Swift.

Esempi di codice per l'utilizzo di Amazon SNS AWS SDKs

I seguenti esempi di codice mostrano come usare Amazon SNS con un kit di sviluppo AWS software (SDK).

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di codice

- [Esempi di base per l'utilizzo di Amazon SNS AWS SDKs](#)
 - [Hello Amazon SNS](#)
 - [Azioni per l'utilizzo di Amazon SNS AWS SDKs](#)
 - [Utilizzo CheckIfPhoneNumberIsOptedOut con un AWS SDK o una CLI](#)
 - [Utilizzo ConfirmSubscription con un AWS SDK o una CLI](#)
 - [Utilizzo CreateTopic con un AWS SDK o una CLI](#)
 - [Utilizzo DeleteTopic con un AWS SDK o una CLI](#)
 - [Utilizzo GetSMSAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo GetTopicAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo ListPhoneNumbersOptedOut con un AWS SDK o una CLI](#)
 - [Utilizzo ListSubscriptions con un AWS SDK o una CLI](#)
 - [Utilizzo ListTopics con un AWS SDK o una CLI](#)
 - [Utilizzo Publish con un AWS SDK o una CLI](#)
 - [Utilizzo SetSMSAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo SetSubscriptionAttributes con un AWS SDK o una CLI](#)
 - [Utilizzare SetSubscriptionAttributesRedrivePolicy con un SDK AWS](#)
 - [Utilizzo SetTopicAttributes con un AWS SDK o una CLI](#)

- [Utilizzo Subscribe con un AWS SDK o una CLI](#)
- [Utilizzo TagResource con un AWS SDK o una CLI](#)
- [Utilizzo Unsubscribe con un AWS SDK o una CLI](#)
- [Scenari per l'utilizzo di Amazon SNS AWS SDKs](#)
 - [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 di piattaforma per le notifiche push di Amazon SNS utilizzando un SDK AWS](#)
 - [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 un argomento Amazon SNS e pubblica messaggi](#)
 - [Crea e pubblica su un argomento FIFO Amazon SNS utilizzando un SDK AWS](#)
 - [Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS](#)
 - [Guida introduttiva a IoT Device Defender](#)
 - [Guida introduttiva a Config](#)
 - [Pubblica messaggi SMS su un argomento Amazon SNS utilizzando un SDK AWS](#)
 - [Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un SDK AWS](#)
 - [Pubblica un messaggio di testo SMS Amazon SNS utilizzando un SDK AWS](#)
 - [Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS](#)
 - [Utilizzo di Gateway API per invocare una funzione Lambda](#)
 - [Utilizzo degli eventi pianificati per invocare una funzione Lambda](#)
- [Esempi serverless per Amazon SNS](#)
 - [Richiamo di una funzione Lambda da un trigger Amazon SNS](#)

Esempi di base per l'utilizzo di Amazon SNS AWS SDKs

I seguenti esempi di codice mostrano come utilizzare le nozioni di base di Amazon Simple Notification Service con AWS SDKs.

Esempi

- [Hello Amazon SNS](#)

- [Azioni per l'utilizzo di Amazon SNS AWS SDKs](#)
 - [Utilizzo CheckIfPhoneNumberIsOptedOut con un AWS SDK o una CLI](#)
 - [Utilizzo ConfirmSubscription con un AWS SDK o una CLI](#)
 - [Utilizzo CreateTopic con un AWS SDK o una CLI](#)
 - [Utilizzo DeleteTopic con un AWS SDK o una CLI](#)
 - [Utilizzo GetSMSAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo GetTopicAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo ListPhoneNumbersOptedOut con un AWS SDK o una CLI](#)
 - [Utilizzo ListSubscriptions con un AWS SDK o una CLI](#)
 - [Utilizzo ListTopics con un AWS SDK o una CLI](#)
 - [Utilizzo Publish con un AWS SDK o una CLI](#)
 - [Utilizzo SetSMSAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo SetSubscriptionAttributes con un AWS SDK o una CLI](#)
 - [Utilizzare SetSubscriptionAttributesRedrivePolicy con un SDK AWS](#)
 - [Utilizzo SetTopicAttributes con un AWS SDK o una CLI](#)
 - [Utilizzo Subscribe con un AWS SDK o una CLI](#)
 - [Utilizzo TagResource con un AWS SDK o una CLI](#)
 - [Utilizzo Unsubscribe con un AWS SDK o una CLI](#)

Hello Amazon SNS

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon SNS.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- Per i dettagli sull'API, [ListTopics](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Codice per il CMake file CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
    libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
        "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
        ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine `hello_sns.cpp`.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 * A "Hello SNS" starter application which initializes an Amazon Simple
 * Notification
 * Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 * main function
 *
 * Usage: 'hello_sns'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
        response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
        }
```



```

        const Aws::SNS::Model::ListTopicsOutcome outcome =
snsClient.ListTopics(
            request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                outcome.GetResult().GetTopics();
            if (!paginatedTopics.empty()) {
                allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                    paginatedTopics.cend());
            }
        }
        else {
            std::cerr << "Error listing topics " <<
outcome.GetError().GetMessage()
                << std::endl;
            return 1;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
topic"
                << (allTopics.size() == 1 ? "" : "s") << " in your account."
                << std::endl;

    if (!allTopics.empty()) {
        std::cout << "Here are your topic ARNs." << std::endl;
        for (const Aws::SNS::Model::Topic &topic: allTopics) {
            std::cout << " * " << topic.GetTopicArn() << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per C++ API Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
// Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    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 i dettagli sull'API, [ListTopics](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
```

```

        .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                .flatMap(r -> r.topics().stream())
                .forEach(content -> System.out.println(" Topic ARN: " +
                    content.topicArn()));
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, [ListTopics](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Inizializza un client SNS ed elenca gli argomenti nel tuo account.

```

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

export const helloSns = async () => {
    // The configuration object (`{}`) is required. If the region and credentials
    // are omitted, the SDK uses your local configuration if it exists.
    const client = new SNSClient({});

```

```
// You can also use `ListTopicsCommand`, but to use that command you must
// handle the pagination yourself. You can do that by sending the
`ListTopicsCommand`
// with the `NextToken` parameter from the previous request.
const paginatedTopics = paginateListTopics({ client }, {});
const topics = [];

for await (const page of paginatedTopics) {
  if (page.Topics?.length) {
    topics.push(...page.Topics);
  }
}

const suffix = topics.length === 1 ? "" : "s";

console.log(
  `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
  account.` ,
);
console.log(topics.map((t) => ` * ${t.TopicArn}`).join("\n"));
};
```

- Per i dettagli sull'API, [ListTopics](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
```

Before running this Kotlin code example, set up your development environment, including your credentials.

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html>

```
*/
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient
            .listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- Per i dettagli sull'API, [ListTopics](#) consulta AWS SDK for Kotlin API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Il file Package.swift.

```
import PackageDescription

let package = Package(
    name: "sns-basics",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
```

```

        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/aws-labs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
        // or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "sns-basics",
            dependencies: [
                .product(name: "AWSSNS", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")
    ]
)

```

Il programma Swift principale.

```

import ArgumentParser
import AWSClientRuntime
import AWSSNS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use (default: us-east-1)")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(

```

```

        commandName: "sns-basics",
        abstract: ""
        This example shows how to list all of your available Amazon SNS topics.
        "",
        discussion: ""
        ""
    )

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await SNSClient.SNSClientConfiguration(region: region)
        let snsClient = SNSClient(config: config)

        var topics: [String] = []
        let outputPages = snsClient.listTopicsPaginated(
            input: ListTopicsInput()
        )

        // Each time a page of results arrives, process its contents.

        for try await output in outputPages {
            guard let topicList = output.topics else {
                print("Unable to get a page of Amazon SNS topics.")
                return
            }

            // Iterate over the topics listed on this page, adding their ARNs
            // to the `topics` array.

            for topic in topicList {
                guard let arn = topic.topicArn else {
                    print("Topic has no ARN.")
                    return
                }
                topics.append(arn)
            }
        }

        print("You have \(topics.count) topics:")
        for topic in topics {
            print("    \(topic)")
        }
    }
}

```



```
/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListTopics](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di Amazon SNS AWS SDKs

I seguenti esempi di codice mostrano come eseguire singole azioni Amazon SNS 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 richiamano l'API Amazon RDS e sono estratti di codice da programmi più grandi che devono essere eseguiti nel contesto. È possibile visualizzare le azioni nel contesto in [Scenari per l'utilizzo di Amazon SNS AWS SDKs](#).

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la [documentazione di riferimento dell'API Amazon Simple Notification Service](#).

Esempi

- [Utilizzo CheckIfPhoneNumberIsOptedOut con un AWS SDK o una CLI](#)
- [Utilizzo ConfirmSubscription con un AWS SDK o una CLI](#)
- [Utilizzo CreateTopic con un AWS SDK o una CLI](#)

- [Utilizzo DeleteTopic con un AWS SDK o una CLI](#)
- [Utilizzo GetSMSAttributes con un AWS SDK o una CLI](#)
- [Utilizzo GetTopicAttributes con un AWS SDK o una CLI](#)
- [Utilizzo ListPhoneNumbersOptedOut con un AWS SDK o una CLI](#)
- [Utilizzo ListSubscriptions con un AWS SDK o una CLI](#)
- [Utilizzo ListTopics con un AWS SDK o una CLI](#)
- [Utilizzo Publish con un AWS SDK o una CLI](#)
- [Utilizzo SetSMSAttributes con un AWS SDK o una CLI](#)
- [Utilizzo SetSubscriptionAttributes con un AWS SDK o una CLI](#)
- [Utilizzare SetSubscriptionAttributesRedrivePolicy con un SDK AWS](#)
- [Utilizzo SetTopicAttributes con un AWS SDK o una CLI](#)
- [Utilizzo Subscribe con un AWS SDK o una CLI](#)
- [Utilizzo TagResource con un AWS SDK o una CLI](#)
- [Utilizzo Unsubscribe con un AWS SDK o una CLI](#)

Utilizzo **CheckIfPhoneNumberIsOptedOut** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare CheckIfPhoneNumberIsOptedOut.

.NET

SDK per .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 (AuthorizationException ex)

```

```
        {  
            Console.WriteLine($"{ex.Message}");  
        }  
    }  
}
```

- Per i dettagli sull'API, [CheckIfPhoneNumberIsOptedOut](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Controllo delle impostazioni di opt-out dei messaggi SMS per un numero di telefono

L'`check-if-phone-number-is-opted-out` esempio seguente verifica se al numero di telefono specificato è stata disattivata la ricezione di messaggi SMS dall' AWS account corrente.

```
aws sns check-if-phone-number-is-opted-out \  
--phone-number +1555550100
```

Output:

```
{  
  "isOptedOut": false  
}
```

- Per i dettagli sull'API, vedere [CheckIfPhoneNumberIsOptedOut](#) in AWS CLI Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

        final String usage = ""

            Usage:    <phoneNumber>

            Where:
                phoneNumber - The mobile phone number to look up (for example,
+1XXX5550100).

            "";

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String phoneNumber = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    checkPhone(snsClient, phoneNumber);
    snsClient.close();
}

public static void checkPhone(SnsClient snsClient, String phoneNumber) {
    try {
        CheckIfPhoneNumberIsOptedOutRequest request =
        CheckIfPhoneNumberIsOptedOutRequest.builder()
            .phoneNumber(phoneNumber)
            .build();

        CheckIfPhoneNumberIsOptedOutResponse result =
        snsClient.checkIfPhoneNumberIsOptedOut(request);
        System.out.println(
            result.isOptedOut() + "Phone Number " + phoneNumber + " has
Opted Out of receiving sns messages." +
            "\n\nStatus was " +
            result.sdkHttpResponse().statusCode());

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

- Per i dettagli sull'API, [CheckIfPhoneNumberIsOptedOut](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
//      totalRetryDelay: 0
//    },
//    isOptedOut: false
//  }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [CheckIfPhoneNumberIsOptedOut](#) consulta AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 * messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

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



```
]);

$phone = '+1XXX5550100';

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

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per i dettagli sull'API, [CheckIfPhoneNumberIsOptedOut](#) consulta AWS SDK per PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **ConfirmSubscription** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare ConfirmSubscription.

CLI

AWS CLI

Conferma di una sottoscrizione

Attraverso il comando `confirm-subscription` seguente viene completato il processo di conferma avviato effettuando la sottoscrizione a un argomento SNS denominato `my-topic`. Il parametro `--token` proviene dal messaggio di conferma inviato all'endpoint di notifica specificato nella chiamata `subscribe`.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
```

```
--
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Output:

```
{
  "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Per i dettagli sull'API, consulta [ConfirmSubscription AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:    <subscriptionToken> <topicArn>

        Where:
            subscriptionToken - A short-lived token sent to an endpoint
            during the Subscribe action.
            topicArn - The ARN of the topic.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
        String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
            ConfirmSubscriptionRequest.builder()
                .token(subscriptionToken)
                .topicArn(topicArn)
                .build();

            ConfirmSubscriptionResponse result =
            snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
            result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                + result.subscriptionArn());

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

```

- Per i dettagli sull'API, [ConfirmSubscription](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                        that are not AWS services (HTTP/S, email) need to be
 *                        confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 *                        a subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
```

```

// A subscription only needs to be confirmed if the endpoint type is
// HTTP/S, email, or in another AWS account.
new ConfirmSubscriptionCommand({
    Token: token,
    TopicArn: topicArn,
    // If this is true, the subscriber cannot unsubscribe while
    unauthenticated.
    AuthenticateOnUnsubscribe: "false",
}),
);
console.log(response);
// {
//   '$metadata': {
//     httpStatusCode: 200,
//     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
//     extendedRequestId: undefined,
//     cfId: undefined,
//     attempts: 1,
//     totalRetryDelay: 0
//   },
//   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxxx'
// }
return response;
};

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [ConfirmSubscription](#) consulta AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

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

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

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

- Per i dettagli sull'API, [ConfirmSubscription](#) consulta AWS SDK per PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **CreateTopic** con un AWS SDK o una CLI

Gli esempi di codice seguenti 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 un argomento Amazon SNS e pubblica messaggi](#)
- [Creazione e pubblicazione su un argomento FIFO](#)
- [Guida introduttiva a Config](#)
- [Guida introduttiva a IoT Device Defender](#)
- [Pubblicazione di messaggi nelle code](#)

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un argomento con un nome di specifico.

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

/// <summary>
/// This example shows how to use Amazon Simple Notification Service
/// (Amazon SNS) to add a new Amazon SNS topic.
/// </summary>
public class CreateSNSTopic
```

```

{
    public static async Task Main()
    {
        string topicName = "ExampleSNSTopic";

        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

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

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

        var response = await client.CreateTopicAsync(request);

        return response.TopicArn;
    }
}

```

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

```

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
    attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>

```



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

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

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

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

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }
}
```

```
    return outcome.IsSuccess();  
}
```

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Creazione di un argomento SNS

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

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

Output:

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

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

- Per i dettagli sull'API, consulta AWS CLI Command [CreateTopic](#) Reference.

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// 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 i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

```

    * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
    */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

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

```

```
    }  
    return "";  
  }  
}
```

- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";  
import { snsClient } from "../libs/snsClient.js";  
  
/**  
 * @param {string} topicName - The name of the topic to create.  
 */  
export const createTopic = async (topicName = "TOPIC_NAME") => {  
  const response = await snsClient.send(  
    new CreateTopicCommand({ Name: topicName }),  
  );  
  console.log(response);  
}
```

```
// {
//   '$metadata': {
//     httpStatusCode: 200,
//     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
//     extendedRequestId: undefined,
//     cfId: undefined,
//     attempts: 1,
//     totalRetryDelay: 0
//   },
//   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
// }
return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun createSNSTopic(topicName: String): String {
    val request =
        CreateTopicRequest {
            name = topicName
        }

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


- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

$topicname = 'myTopic';

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

```
error_log($e->getMessage());  
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class SnsWrapper:  
    """Encapsulates Amazon SNS topic and subscription functions."""  
  
    def __init__(self, sns_resource):  
        """  
        :param sns_resource: A Boto3 Amazon SNS resource.  
        """  
        self.sns_resource = sns_resource  
  
    def create_topic(self, name):  
        """  
        Creates a notification topic.  
  
        :param name: The name of the topic to create.  
        :return: The newly created topic.  
        """  
        try:  
            topic = self.sns_resource.create_topic(Name=name)  
            logger.info("Created topic %s with ARN %s.", name, topic.arn)  
        except ClientError:  
            logger.exception("Couldn't create topic %s.", name)  
            raise
```

```
else:
    return topic
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """
        Create an SNS topic.

        :param topic_name: The name of the topic to create.
        :param is_fifo: Whether to create a FIFO topic.
        :param content_based_deduplication: Whether to use content-based
        deduplication for FIFO topics.
        :return: The ARN of the created topic.
        :raises ClientError: If the topic creation fails.
        """
```

```

try:
    # Add .fifo suffix for FIFO topics
    if is_fifo and not topic_name.endswith('.fifo'):
        topic_name += '.fifo'

    attributes = {}
    if is_fifo:
        attributes['FifoTopic'] = 'true'
        if content_based_deduplication:
            attributes['ContentBasedDeduplication'] = 'true'

    response = self.sns_client.create_topic(
        Name=topic_name,
        Attributes=attributes
    )

    topic_arn = response['TopicArn']
    logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
    return topic_arn

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error creating topic {topic_name}: {error_code} - {e}")
    raise

```

- Per i dettagli sull'API, consulta [CreateTopic AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

# This class demonstrates how to create an Amazon Simple Notification Service
(SNS) topic.

```

```
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

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

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

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

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per i dettagli sull'API, [CreateTopic](#) consulta AWS SDK per Ruby API Reference.

Rust

SDK per 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 i dettagli sulle API, consulta la [CreateTopic](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

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

- Per i dettagli sulle API, [CreateTopic](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS  
  
let config = try await SNSClient.SNSClientConfiguration(region: region)  
let snsClient = SNSClient(config: config)  
  
let output = try await snsClient.createTopic(  
    input: CreateTopicInput(name: name)  
)  
  
guard let arn = output.topicArn else {  
    print("No topic ARN returned by Amazon SNS.")  
    return  
}
```

- Per i dettagli sull'API, consulta la [CreateTopic](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **DeleteTopic** con un AWS SDK o una CLI

Gli esempi di codice seguenti 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 nei seguenti esempi di codice:

- [Crea un argomento Amazon SNS e pubblica messaggi](#)
- [Guida introduttiva a Config](#)
- [Guida introduttiva a IoT Device Defender](#)
- [Pubblicazione di messaggi nelle code](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un argomento in base all'ARN dell'argomento.

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


- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Eliminazione di un argomento SNS

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

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

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteTopic AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
  "context"  
  "encoding/json"  
  "log"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/service/sns"  
  "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// 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 i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

```

    * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
    */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <topicArn>

            Where:
                topicArn - The ARN of the topic to delete.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

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

```

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

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

```
//     extendedRequestId: undefined,  
//     cfId: undefined,  
//     attempts: 1,  
//     totalRetryDelay: 0  
//   }  
// }  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun deleteSNSTopic(topicArnVal: String) {  
    val request =  
        DeleteTopicRequest {  
            topicArn = topicArnVal  
        }  
  
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->  
        snsClient.deleteTopic(request)  
        println("$topicArnVal was successfully deleted.")  
    }  
}
```

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

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

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

- Per i dettagli sull'API, [DeleteTopic](#) consulta AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""
```



```
def __init__(self, sns_client: Any) -> None:
    """
    Initialize the SnsWrapper.

    :param sns_client: A Boto3 Amazon SNS client.
    """
    self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def delete_topic(self, topic_arn: str) -> bool:
        """
        Delete an SNS topic.

        :param topic_arn: The ARN of the topic to delete.
        :return: True if successful.
        :raises ClientError: If the topic deletion fails.
        """
        try:
            self.sns_client.delete_topic(TopicArn=topic_arn)

            logger.info(f"Deleted topic: {topic_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Topic not found: {topic_arn}")
                return True # Already deleted
            else:
                logger.error(f"Error deleting topic: {error_code} - {e}")
                raise
```

- Per i dettagli sull'API, consulta [DeleteTopic AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per 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 i dettagli sulle API, [DeleteTopic](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS  
  
let config = try await SNSClient.SNSClientConfiguration(region: region)  
let snsClient = SNSClient(config: config)
```

```

    _ = try await snsClient.deleteTopic(
        input: DeleteTopicInput(topicArn: arn)
    )

```

- Per i dettagli sull'API, consulta la [DeleteTopic](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **GetSMSAttributes** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetSMSAttributes.

C++

SDK per C++

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

//! Retrieve the default settings for sending SMS messages from your AWS account
  by using
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

```

```

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
snsClient.GetSMSAttributes(
    request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
            outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ": " << att.second << std::endl;
            }
        }
        else {
            std::cout
                << "AwsDoc::SNS::getSMSType - an empty map of attributes was
retrieved."
                << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
            << outcome.GetError().GetMessage()
            << "'" << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, consulta [Get SMSAttributes](#) in AWS SDK per C++ API Reference.

CLI

AWS CLI

Elencare gli attributi predefiniti dei messaggi SMS

Nell'esempio `get-sms-attributes` seguente vengono elencati gli attributi predefiniti per l'invio di messaggi SMS.

```
aws sns get-sms-attributes
```

Output:

```
{
  "attributes": {
    "DefaultSenderId": "MyName"
  }
}
```

- Per informazioni dettagliate sull'API, consulta [GetSMSAttributes](#) in AWS CLI Command Reference.

Java**SDK per Java 2.x** **Note**

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
  software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
  software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class GetSMSAttributes {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <topicArn>

        Where:
            topicArn - The ARN of the topic from which to retrieve
attributes.
        "";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    getSNSAttrutes(snsClient, topicArn);
    snsClient.close();
}

public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
    try {
        GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
            .subscriptionArn(topicArn)
            .build();

        // Get the Subscription attributes
        GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
        Map<String, String> map = res.attributes();

        // Iterate through the map
        Iterator iter = map.entrySet().iterator();
        while (iter.hasNext()) {
            Map.Entry entry = (Map.Entry) iter.next();
            System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
        }
    }
}
```

```
    } catch (SnsException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
  
    System.out.println("\n\nStatus was good");  
}  
}
```

- Per i dettagli sull'API, consulta [Get SMSAttributes](#) in AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";  
import { snsClient } from "../libs/snsClient.js";  
  
export const getSmsAttributes = async () => {  
    const response = await snsClient.send(  
        // If you have not modified the account-level mobile settings of SNS,  
        // the DefaultSMSType is undefined. For this example, it was set to
```

```
// Transactional.
new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
);

console.log(response);
// {
//   '$metadata': {
//     httpStatusCode: 200,
//     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
//     extendedRequestId: undefined,
//     cfId: undefined,
//     attempts: 1,
//     totalRetryDelay: 0
//   },
//   attributes: { DefaultSMSType: 'Transactional' }
// }
return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta [Get SMSAttributes](#) in AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
```



```
* 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 per PHP](#).
- Per i dettagli sull'API, consulta [Get SMSAttributes](#) in AWS SDK per PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **GetTopicAttributes** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetTopicAttributes.

.NET

SDK per .NET

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.SimpleNotificationService;

/// <summary>
/// This example shows how to retrieve the attributes of an Amazon Simple
/// Notification Service (Amazon SNS) topic.
/// </summary>
public class GetTopicAttributes
{
    public static async Task Main()
    {
        string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        var attributes = await GetTopicAttributesAsync(client, topicArn);
        DisplayTopicAttributes(attributes);
    }

    /// <summary>
    /// Given the ARN of the Amazon SNS topic, this method retrieves the
    topic
    /// attributes.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to retrieve the attributes for the Amazon SNS topic.</param>
    /// <param name="topicArn">The ARN of the topic for which to retrieve
    /// the attributes.</param>
    /// <returns>A Dictionary of topic attributes.</returns>
```

```

        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
    {
        var response = await client.GetTopicAttributesAsync(topicArn);

        return response.Attributes;
    }

    /// <summary>
    /// This method displays the attributes for an Amazon SNS topic.
    /// </summary>
    /// <param name="topicAttributes">A Dictionary containing the
    /// attributes for an Amazon SNS topic.</param>
    public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
    {
        foreach (KeyValuePair<string, string> entry in topicAttributes)
        {
            Console.WriteLine($"{entry.Key}: {entry.Value}\n");
        }
    }
}

```

- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
topic.
/*!

```

```

\param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
    snsClient.GetTopicAttributes(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Recupero degli attributi di un argomento

Nell'esempio `get-topic-attributes` seguente vengono visualizzati gli attributi per l'argomento specificato.

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

Output:

```
{
  "Attributes": {
    "SubscriptionsConfirmed": "1",
    "DisplayName": "my-topic",
    "SubscriptionsDeleted": "0",
    "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy\":"
    "\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,"
    "\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,"
    "\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
    "Owner": "123456789012",
    "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID\","
    "\"Statement\": [{\"Sid\":\"__default_statement_ID\",\"Effect\":\""
    "\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\": [\"SNS:Subscribe\",
    \"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes"
    "\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
    \"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-"
    "topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
    \"0123456789012\"}}}]}",
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
    "SubscriptionsPending": "0"
  }
}
```

- Per i dettagli sull'API, consulta [GetTopicAttributes AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

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

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to look up.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        System.out.println("Getting attributes for a topic with name: " +
            topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
        topicArn) {
        try {
```

```
        GetTopicAttributesRequest request =
        GetTopicAttributesRequest.builder()
            .topicArn(topicArn)
            .build();

        GetTopicAttributesResponse result =
        snsClient.getTopicAttributes(request);
        System.out.println("\n\nStatus is " +
        result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
            + result.attributes());

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

- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDelayRetries":0,"initialInterval":0,"backoffFunction":"linear","retryPolicyType":"Standard"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).

- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK per JavaScript API Reference.

SDK per 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](#).

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

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun getSnsTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

```

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

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

```

- Per i dettagli sull'API, [GetTopicAttributes](#) consulta AWS SDK per PHP API Reference.

SAP ABAP

SDK per 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 i dettagli sulle API, [GetTopicAttributes](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **ListPhoneNumbersOptedOut** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Elencare le opzioni di opt-out dei messaggi SMS

Nell'esempio `list-phone-numbers-opted-out` seguente sono elencati i numeri di telefono per i quali sono abilitate le opzioni di opt-out per la ricezione di SMS.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
  "phoneNumbers": [
    "+15555550100"
  ]
}
```

- Per i dettagli sull'API, consulta [ListPhoneNumbersOptedOut AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

```
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
    software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

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

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
                ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
                snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
                result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

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

- Per i dettagli sull'API, [ListPhoneNumbersOptedOut](#) consulta AWS SDK for Java 2.x API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 * from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

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

try {
    $result = $SnSClient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per i dettagli sull'API, [ListPhoneNumbersOptedOut](#) consulta AWS SDK per PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **ListSubscriptions** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare ListSubscriptions.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/// <summary>
/// This example will retrieve a list of the existing Amazon Simple
/// Notification Service (Amazon SNS) subscriptions.
/// </summary>
public class ListSubscriptions
{
    public static async Task Main()
    {
        IAmazonSimpleNotificationService client = new
        AmazonSimpleNotificationServiceClient();
```

```

        Console.WriteLine("Enter a topic ARN to list subscriptions for a
specific topic, " +
                           "or press Enter to list subscriptions for all
topics.");
        var topicArn = Console.ReadLine();
        Console.WriteLine();

        var subscriptions = await GetSubscriptionsListAsync(client,
topicArn);

        DisplaySubscriptionList(subscriptions);
    }

    /// <summary>
    /// Gets a list of the existing Amazon SNS subscriptions, optionally by
specifying a topic ARN.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to obtain the list of subscriptions.</param>
    /// <param name="topicArn">The optional ARN of a specific topic. Defaults
to null.</param>
    /// <returns>A list containing information about each subscription.</
returns>
    public static async Task<List<Subscription>>
GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
topicArn = null)
    {
        var results = new List<Subscription>();

        if (!string.IsNullOrEmpty(topicArn))
        {
            var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                new ListSubscriptionsByTopicRequest()
                {
                    TopicArn = topicArn,
                });

            // Get the entire list using the paginator.
            await foreach (var subscription in paginateByTopic.Subscriptions)
            {
                results.Add(subscription);
            }
        }
    }

```



```

        else
        {
            var paginateAllSubscriptions =
client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

            // Get the entire list using the paginator.
            await foreach (var subscription in
paginateAllSubscriptions.Subscriptions)
            {
                results.Add(subscription);
            }
        }

        return results;
    }

    /// <summary>
    /// Display a list of Amazon SNS subscription information.
    /// </summary>
    /// <param name="subscriptionList">A list containing details for existing
    /// Amazon SNS subscriptions.</param>
    public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
    {
        foreach (var subscription in subscriptionList)
        {
            Console.WriteLine($"Owner: {subscription.Owner}");
            Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
            Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
            Console.WriteLine($"Endpoint: {subscription.Endpoint}");
            Console.WriteLine($"Protocol: {subscription.Protocol}");
            Console.WriteLine();
        }
    }
}

```

- Per i dettagli sull'API, [ListSubscriptions](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
subscriptions.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
            snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "

```

```

        << outcome.GetError().GetMessage()
        <<
        std::endl;
        result = false;
        break;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

if (result) {
    if (subscriptions.empty()) {
        std::cout << "No subscriptions found" << std::endl;
    }
    else {
        std::cout << "Subscriptions list:" << std::endl;
        for (auto const &subscription: subscriptions) {
            std::cout << "    * " << subscription.GetSubscriptionArn() <<
std::endl;
        }
    }
}
return result;
}

```

- Per i dettagli sull'API, [ListSubscriptions](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Elencare le sottoscrizioni SNS

L'`list-subscription`esempio seguente mostra un elenco degli abbonamenti SNS presenti nel tuo AWS account.

```
aws sns list-subscriptions
```

Output:

```
{
```

```
"Subscriptions": [  
  {  
    "Owner": "123456789012",  
    "Endpoint": "my-email@example.com",  
    "Protocol": "email",  
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",  
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-topic:8a21d249-4329-4871-acc6-7be709c6ea7f"  
  }  
]  
}
```

- Per i dettagli sull'API, consulta AWS CLI Command [ListSubscriptions](#) Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;  
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class ListSubscriptions {  
    public static void main(String[] args) {  
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    listSNSSubscriptions(snsClient);
    snsClient.close();
}

public static void listSNSSubscriptions(SnsClient snsClient) {
    try {
        ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
            .build();

        ListSubscriptionsResponse result =
snsClient.listSubscriptions(request);
        System.out.println(result.subscriptions());

    } catch (SnsException e) {

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

- Per i dettagli sull'API, [ListSubscriptions](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [ListSubscriptions](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun listSNSSubscriptions() {  
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->  
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})  
        response.subscriptions?.forEach { sub ->  
            println("Sub ARN is ${sub.subscriptionArn}")  
            println("Sub protocol is ${sub.protocol}")  
        }  
    }  
}
```

- Per i dettagli sull'API, [ListSubscriptions](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;

/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

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

try {
    $result = $SnSClient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la [ListSubscriptions](#) sezione AWS SDK per PHP API Reference.

Python

SDK per 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 i dettagli sull'API, consulta [ListSubscriptions AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
# This class demonstrates how to list subscriptions to an Amazon Simple
Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
```

```

rescue StandardError => e
  puts "Failed to list subscriptions: #{e.message}"
  exit 1
end
end

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per i dettagli sull'API, consulta la [ListSubscriptions](#) sezione AWS SDK per Ruby API Reference.

SAP ABAP

SDK per 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 i dettagli sulle API, [ListSubscriptions](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **ListTopics** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare `ListTopics`.

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:

- [Guida introduttiva a IoT Device Defender](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/// <summary>
/// Lists the Amazon Simple Notification Service (Amazon SNS)
/// topics for the current account.
/// </summary>
public class ListSNSTopics
{
    public static async Task Main()
    {
        IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

        await GetTopicListAsync(client);
    }

    /// <summary>
    /// Retrieves the list of Amazon SNS topics in groups of up to 100
    /// topics.
    /// </summary>
    /// <param name="client">The initialized Amazon SNS client object used
    /// to retrieve the list of topics.</param>
```

```

    public static async Task
    GetTopicListAsync(IAmazonSimpleNotificationService client)
    {
        // If there are more than 100 Amazon SNS topics, the call to
        // ListTopicsAsync will return a value to pass to the
        // method to retrieve the next 100 (or less) topics.
        string nextToken = string.Empty;

        do
        {
            var response = await client.ListTopicsAsync(nextToken);
            DisplayTopicsList(response.Topics);
            nextToken = response.NextToken;
        }
        while (!string.IsNullOrEmpty(nextToken));
    }

    /// <summary>
    /// Displays the list of Amazon SNS Topic ARNs.
    /// </summary>
    /// <param name="topicList">The list of Topic ARNs.</param>
    public static void DisplayTopicsList(List<Topic> topicList)
    {
        foreach (var topic in topicList)
        {
            Console.WriteLine($"{topic.TopicArn}");
        }
    }
}

```

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per .NET API Reference.

C++

SDK per 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 i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Elenco degli argomenti SNS

L'`list-topics` esempio seguente elenca tutti gli argomenti SNS del tuo AWS account.

```
aws sns list-topics
```

Output:

```
{
  "Topics": [
    {
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
    }
  ]
}
```

- Per i dettagli sull'API, consulta [ListTopics AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package main

import (
    "context"
    "fmt"
```

```
"log"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/sns"
"github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
// Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    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 i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
```

```

        .build();

        ListTopicsResponse result = snsClient.listTopics(request);
        System.out.println(
            "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

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

```

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

```

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

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

```

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

```

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

```

```
export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

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](#).

```
suspend fun listSNSTopics() {
  SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
    val response = snsClient.listTopics(ListTopicsRequest { })
    response.topics?.forEach { topic ->
      println("The topic ARN is ${topic.topicArn}")
    }
  }
}
```

- Per i dettagli sull'API, [ListTopics](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Returns a list of the requester's topics from your AWS SNS account in the
 * region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

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

try {
    $result = $SnSClient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per PHP API Reference.

Python

SDK per 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 i dettagli sull'API, consulta [ListTopics AWS](#) SDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-sns' # v2: require 'aws-sdk'

def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  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 per Ruby](#).

- Per i dettagli sull'API, consulta la [ListTopics](#) sezione AWS SDK per Ruby API Reference.

Rust

SDK per 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 i dettagli sulle API, consulta la [ListTopics](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per 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 i dettagli sulle API, [ListTopics](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

var topics: [String] = []
let outputPages = snsClient.listTopicsPaginated(
    input: ListTopicsInput()
)

// Each time a page of results arrives, process its contents.

for try await output in outputPages {
    guard let topicList = output.topics else {
        print("Unable to get a page of Amazon SNS topics.")
        return
    }

    // Iterate over the topics listed on this page, adding their ARNs
    // to the `topics` array.

```



```
        for topic in topicList {
            guard let arn = topic.topicArn else {
                print("Topic has no ARN.")
                return
            }
            topics.append(arn)
        }
    }
```

- Per i dettagli sull'API, consulta la [ListTopics](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **Publish** con un AWS SDK o una CLI

Gli esempi di codice seguenti 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 un argomento Amazon SNS e pubblica messaggi](#)
- [Creazione e pubblicazione su un argomento FIFO](#)
- [Pubblicazione di un SMS](#)
- [Pubblicazione di messaggi nelle code](#)

.NET

SDK per .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 messageId = 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($"{i + 1}. {_tones[i]}");
        }

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

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

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

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

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

```

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

```

/// <summary>
/// Publish a message to a topic with an attribute and optional deduplication
and group IDs.

```

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

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

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

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

```
    return outcome.IsSuccess();
}
```

Pubblica un messaggio con un attributo.

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

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

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

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

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

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

if (outcome.IsSuccess()) {
    std::cout << "Your message was successfully published." << std::endl;
}
else {
```

```
std::cerr << "Error with TopicsAndQueues::Publish. "  
          << outcome.GetError().GetMessage()  
          << std::endl;  
  
cleanUp(topicARN,  
        queueURLS,  
        subscriptionARNS,  
        snsClient,  
        sqsClient);  
  
return false;  
}
```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

CLI

AWS CLI

Esempio 1: pubblicazione di un messaggio in un argomento

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

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

Contenuto di `message.txt`:

```
Hello World  
Second Line
```

Output:

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


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

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

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

Output:

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

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dei comandi della AWS CLI .

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
  "context"  
  "encoding/json"  
  "log"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/service/sns"  
  "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// 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 informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Go .

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

            Usage:    <message> <topicArn>

            Where:
                message - The message text to send.
                topicArn - The ARN of the topic to publish.
            "";

        if (args.length != 2) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String message = args[0];
    String topicArn = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();
    pubTopic(snsClient, message, topicArn);
    snsClient.close();
}

public static void pubTopic(SnsClient snsClient, String message, String
topicArn) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

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

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

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

```
);  
console.log(response);  
// {  
//   '$metadata': {  
//     httpStatusCode: 200,  
//     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',  
//     extendedRequestId: undefined,  
//     cfId: undefined,  
//     attempts: 1,  
//     totalRetryDelay: 0  
//   },  
//   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx'  
// }  
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 per JavaScript](#).

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per JavaScript .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun pubTopic(
    topicArnVal: String,
    messageVal: String,
) {
    val request =
        PublishRequest {
            message = messageVal
            topicArn = topicArnVal
        }

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

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Kotlin.

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

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

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

```
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

PowerShell

Strumenti per PowerShell V4

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

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -  
Message "Hello" -MessageAttribute  
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{'DataType='String'  
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 informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dei cmdlet di AWS Strumenti per PowerShell (V4).

Strumenti per PowerShell V5

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

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -  
Message "Hello" -MessageAttribute  
@{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='String'  
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 informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dei cmdlet di AWS Strumenti per PowerShell (V5).

Python

SDK per Python (Boto3)

Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Pubblicare un messaggio 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

```

Publicare 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

```

```

class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod

```

```
def from_client(cls) -> 'SnsWrapper':
    """
    Create an SnsWrapper instance using a default boto3 client.

    :return: An instance of this class.
    """
    sns_client = boto3.client('sns')
    return cls(sns_client)


def publish_message(
    self,
    topic_arn: str,
    message: str,
    tone_attribute: Optional[str] = None,
    deduplication_id: Optional[str] = None,
    message_group_id: Optional[str] = None
) -> str:
    """
    Publish a message to an SNS topic.

    :param topic_arn: The ARN of the SNS topic.
    :param message: The message content to publish.
    :param tone_attribute: Optional tone attribute for message filtering.
    :param deduplication_id: Optional deduplication ID for FIFO topics.
    :param message_group_id: Optional message group ID for FIFO topics.
    :return: The message ID of the published message.
    :raises ClientError: If the message publication fails.
    """
    try:
        publish_args = {
            'TopicArn': topic_arn,
            'Message': message
        }

        # Add message attributes if tone is specified
        if tone_attribute:
            publish_args['MessageAttributes'] = {
                'tone': {
                    'DataType': 'String',
                    'StringValue': tone_attribute
                }
            }
    
```

```

        # Add FIFO-specific parameters
        if message_group_id:
            publish_args['MessageGroupId'] = message_group_id

        if deduplication_id:
            publish_args['MessageDeduplicationId'] = deduplication_id

        response = self.sns_client.publish(**publish_args)

        message_id = response['MessageId']
        logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
        return message_id

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error publishing message to topic: {error_code} -
{e}")
        raise

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API SDK for Python (Boto3)AWS .

Ruby

SDK per Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

# Service class for sending messages using Amazon Simple Notification Service
(SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client

```



```

def initialize(sns_client)
  @sns_client = sns_client
  @logger = Logger.new($stdout)
end

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

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

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

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

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

Rust

SDK per 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 informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per 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.
```

Pubblica un messaggio con attributi relativi a un argomento.

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

Pubblica un messaggio multiformato su un argomento.

```
" Build JSON message structure for multi-format message
DATA(lv_json_message) = |\{ "default": "{ iv_default_message }", "sms":
"{ iv_sms_message }", "email": "{ iv_email_message }" \}|.
```

```

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

```

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

let output = try await snsClient.publish(
    input: PublishInput(
        message: message,
        topicArn: arn
    )
)

guard let messageId = output.messageId else {
    print("No message ID received from Amazon SNS.")
    return
}

```

```
}

print("Published message with ID \"(messageId)")
```

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dell'API AWS SDK per Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **SetSMSAttributes** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare SetSMSAttributes.

C++

SDK per C++

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Come usare Amazon SNS per impostare l'attributo predefinito SMSType .

```
//! 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 i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK per C++ API Reference.

CLI

AWS CLI

Impostazione degli attributi dei messaggi SMS

Nell'esempio `set-sms-attributes` seguente l'ID mittente predefinito per i messaggi SMS viene impostato su `MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderId=MyName
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS CLI Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
String> attributes) {
        try {
```

```
        SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
            .attributes(attributes)
            .build();

        SetSmsAttributesResponse result =
snsClient.setSMSAttributes(request);
        System.out.println("Set default Attributes to " + attributes + ".
Status was "
            + result.sdkHttpResponse().statusCode());

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

- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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


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

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional - (Default) Noncritical messages, such as marketing
        // messages.
        // Transactional - Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
$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 per PHP](#).
- Per i dettagli sull'API, consulta [Set SMSAttributes](#) in AWS SDK per PHP API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **SetSubscriptionAttributes** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare SetSubscriptionAttributes.

CLI

AWS CLI

Impostazione degli attributi della sottoscrizione

Nell'esempio `set-subscription-attributes` seguente viene impostato l'attributo `RawMessageDelivery` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
  --subscription-arn arn:aws:sns:us-east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
  --attribute-name RawMessageDelivery \
  --attribute-value true
```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene impostato un attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
  --subscription-arn arn:aws:sns:us-east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
  --attribute-name FilterPolicy \
  --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

Questo comando non produce alcun output.

Nell'esempio `set-subscription-attributes` seguente viene rimosso l'attributo `FilterPolicy` su una sottoscrizione SQS.

```
aws sns set-subscription-attributes \
  --subscription-arn arn:aws:sns:us-east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
  --attribute-name FilterPolicy \
  --attribute-value "{}"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [SetSubscriptionAttributes AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

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

            Usage:    <subscriptionArn>

            Where:
                subscriptionArn - The ARN of a subscription.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    usePolicy(snsClient, subscriptionArn);
    snsClient.close();
}

public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);

        // Add a numeric attribute with a range
        fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

        // Apply the filter policy attributes to an Amazon SNS subscription
        fp.apply(snsClient, subscriptionArn);

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

- Per i dettagli sull'API, consulta la [SetSubscriptionAttributes](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per 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 i dettagli sull'API, consulta [SetSubscriptionAttributes AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per 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->setsubscriptionattributes(
        iv_subscriptionarn = iv_subscription_arn
        iv_attributename   = 'FilterPolicy'
        iv_attributevalue  = iv_filter_policy ).
    MESSAGE 'Added filter policy to subscription.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Subscription does not exist.' TYPE 'E'.
ENDTRY.

```

- Per i dettagli sulle API, [SetSubscriptionAttributes](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **SetSubscriptionAttributesRedrivePolicy** con un SDK AWS

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

Java

SDK per Java 1.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
// attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **SetTopicAttributes** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare SetTopicAttributes.

CLI

AWS CLI

Impostazione di un attributo per un argomento

Nell'esempio `set-topic-attributes` seguente vengono impostati gli attributi `DisplayName` per l'argomento specificato.

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name DisplayName \  
  --attribute-value MyTopicDisplayName
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [SetTopicAttributes AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;  
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;  
import software.amazon.awssdk.services.sns.model.SnsException;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 */
```

```

* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <attribute> <topicArn> <value>

            Where:
                attribute - The attribute action to use. Valid parameters are:
Policy | DisplayName | DeliveryPolicy .
                topicArn - The ARN of the topic.\s
                value - The value for the attribute.
            "";

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
SetTopicAttributesRequest.builder()
                .attributeName(attribute)
                .attributeValue(value)
                .topicArn(topicArn)

```

```
        .build();

        SetTopicAttributesResponse result =
snsClient.setTopicAttributes(request);
        System.out.println(
            "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
"\n\nTopic " + request.topicArn()
            + " updated " + request.attributeName() + " to " +
request.attributeValue());

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

- Per i dettagli sull'API, consulta la [SetTopicAttributes](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [SetTopicAttributes](#) sezione AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

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](#).

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- Per i dettagli sull'API, [SetTopicAttributes](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
 * guide_credentials.html
 */

$SnSClient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSClient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per i dettagli sull'API, consulta la [SetTopicAttributes](#) sezione AWS SDK per PHP API Reference.

Ruby

SDK per Ruby

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
# 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 per Ruby](#).
- Per i dettagli sull'API, consulta la [SetTopicAttributes](#) sezione AWS SDK per Ruby API Reference.

SAP ABAP

SDK per 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 i dettagli sulle API, [SetTopicAttributes](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **Subscribe** con un AWS SDK o una CLI

Gli esempi di codice seguenti 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 un argomento Amazon SNS e pubblica messaggi](#)
- [Creazione e pubblicazione su un argomento FIFO](#)
- [Pubblicazione di messaggi nelle code](#)

.NET

SDK per .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](#).

Sottoscrive un indirizzo e-mail a un argomento.

```
/// <summary>
/// Creates a new subscription to a topic.
/// </summary>
/// <param name="client">The initialized Amazon SNS client object, used
/// to create an Amazon SNS subscription.</param>
/// <param name="topicArn">The ARN of the topic to subscribe to.</param>
/// <returns>A SubscribeResponse object which includes the subscription
/// ARN for the new subscription.</returns>
public static async Task<SubscribeResponse> TopicSubscribeAsync(
    IAmazonSimpleNotificationService client,
    string topicArn)
{
    SubscribeRequest request = new SubscribeRequest()
    {
        TopicArn = topicArn,
        ReturnSubscriptionArn = true,
        Protocol = "email",
        Endpoint = "recipient@example.com",
    };

    var response = await client.SubscribeAsync(request);

    return response;
}
```

Iscrivi una coda a un argomento con filtri opzionali.

```

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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();
}

```

Sottoscrive 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();
}

```

Sottoscrive 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();
}

```

Sottoscrive una coda SQS a un argomento.

```

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

```

```

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

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);

    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
                    << "' has been subscribed to the topic '"
                    << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN <<
". "
                    << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;

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

        return false;
    }
}

```

Sottoscrive con un filtro a un argomento.

```

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

```

```

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

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

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have
filters."
                        << std::endl;
            std::cout
                << "If you add a filter to this subscription, then
only the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                << std::endl;
            std::cout << "For this example, you can filter messages by a
\""
                << TONE_ATTRIBUTE << "\" attribute." << std::endl;
        }

        std::ostringstream ostream;
        ostream << "Filter messages for \"" << queueName
                << "\"'s subscription to the topic \""
                << topicName << "\"? (y/n)";

        // Add filter if user answers yes.
        if (askYesNoQuestion(ostream.str())) {
            Aws::String jsonPolicy = getFilterPolicyFromUser();
            if (!jsonPolicy.empty()) {
                filteringMessages = true;

                std::cout << "This is the filter policy for this
subscription."
                        << std::endl;
                std::cout << jsonPolicy << std::endl;
            }
        }
    }
}

```



```

        request.AddAttributes("FilterPolicy", jsonPolicy);
    }
    else {
        std::cout
            << "Because you did not select any attributes, no
filter "
            << "will be added to this subscription." <<
std::endl;
    }
}
} // if (isFifoTopic)
Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

if (outcome.IsSuccess()) {
    Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
    std::cout << "The queue '" << queueName
        << "' has been subscribed to the topic '"
        << "'" << topicName << "'" << std::endl;
    std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
        << std::endl;
    subscriptionARNS.push_back(subscriptionARN);
}
else {
    std::cerr << "Error with TopicsAndQueues::Subscribe. "
        << outcome.GetError().GetMessage()
        << std::endl;

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

    return false;
}

//! Routine that lets the user select attributes for a subscription filter
policy.
/*!
\sa getFilterPolicyFromUser()

```

```

\return Aws::String: The filter policy as JSON.
*/
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
        << "You can filter messages by one or more of the following \""
        << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                << std::endl;
        }
        selection = askQuestionForIntRange(
            "Enter a number (or enter zero to stop adding more). ",
            0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                filterSelections.end(),
                selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";

        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";
    }
}

```

```
        result = jsonPolicyStream.str();  
    }  
  
    return result;  
}
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

CLI

AWS CLI

Effettuare la sottoscrizione a un argomento

Attraverso il comando `subscribe` seguente viene effettuata la sottoscrizione all'argomento specificato utilizzando un indirizzo e-mail.

```
aws sns subscribe \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \  
  --protocol email \  
  --notification-endpoint my-email@example.com
```

Output:

```
{  
  "SubscriptionArn": "pending confirmation"  
}
```

- Per informazioni dettagliate sull'API, consulta [Subscribe](#) nella documentazione di riferimento dei comandi della AWS CLI .

Go

SDK per Go V2

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Iscrivi una coda a un argomento con filtri opzionali.

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// 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),
    ReturnSubscription: true,
})
if err != nil {
    log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
        queueArn, topicArn, err)
} else {
    subscriptionArn = *output.SubscriptionArn
}

return subscriptionArn, err
}

```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Go .

Java

SDK for Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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);
    }
}
}

```

Sottoscrive un endpoint HTTP a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

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

            Usage:    <topicArn> <url>

            Where:
                topicArn - The ARN of the topic to subscribe.

```

```

        url - The HTTPS endpoint that you want to receive
        notifications.
        """;

    if (args.length < 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String topicArn = args[0];
    String url = args[1];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    subHTTPS(snsClient, topicArn, url);
    snsClient.close();
}

public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
{
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("https")
            .endpoint(url)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        System.out.println("Subscription ARN is " + result.subscriptionArn()
+ "\n\n Status is "
            + result.sdkHttpResponse().statusCode());

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

```

Sottoscrive una funzione Lambda a un argomento.


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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = ""

            Usage:    <topicArn> <lambdaArn>

            Where:
                topicArn - The ARN of the topic to subscribe.
                lambdaArn - The ARN of an AWS Lambda function.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }
}
```

```
public static String subLambda(SnsClient snsClient, String topicArn, String
lambdaArn) {
    try {
        SubscribeRequest request = SubscribeRequest.builder()
            .protocol("lambda")
            .endpoint(lambdaArn)
            .returnSubscriptionArn(true)
            .topicArn(topicArn)
            .build();

        SubscribeResponse result = snsClient.subscribe(request);
        return result.subscriptionArn();

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

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

// The AWS Region can be provided here using the `region` property. If you leave
it blank
```

```
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

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

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 * a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 * topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "user@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'
  // }
};
```

Sottoscrive 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;
};

```

Sottoscrive 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;
};

```

Sottoscrive una coda SQS a un argomento.

```

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

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,

```

```

    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx'
  // }
  return response;
};

```

Sottoscrive con un filtro a un argomento.

```

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

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
      set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

```

```

    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
  test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx'
  // }
  return response;
};

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per JavaScript .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail a un argomento.

```

suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {

```

```
val request =
    SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

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

Sottoscrive 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.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento degli SDK AWS per Kotlin.

PHP

SDK for PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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());  
}
```

Sottoscrive un endpoint HTTP a un argomento.

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\Sns\SnsClient;  
  
/**  
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation  
 * message.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
guide\_credentials.html  
 */  
  
$SnSClient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);  
  
$protocol = 'https';  
$endpoint = 'https://';  
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';  
  
try {  
    $result = $SnSClient->subscribe([  
        'Protocol' => $protocol,  
        'Endpoint' => $endpoint,  
        'ReturnSubscriptionArn' => true,  
        'TopicArn' => $topic,  
    ]);
```

```
var_dump($result);  
} catch (AwsException $e) {  
    // output error message if fails  
    error_log($e->getMessage());  
}
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

Python

SDK for Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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

```

Iscrivi una coda a un argomento con filtri opzionali.

```

class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

```

```
:return: An instance of this class.
"""

sns_client = boto3.client('sns')
return cls(sns_client)

def subscribe_queue_to_topic(
    self,
    topic_arn: str,
    queue_arn: str,
    filter_policy: Optional[str] = None
) -> str:
    """
    Subscribe an SQS queue to an SNS topic.

    :param topic_arn: The ARN of the SNS topic.
    :param queue_arn: The ARN of the SQS queue.
    :param filter_policy: Optional JSON filter policy for message filtering.
    :return: The ARN of the subscription.
    :raises ClientError: If the subscription fails.
    """
    try:
        attributes = {}
        if filter_policy:
            attributes['FilterPolicy'] = filter_policy

        response = self.sns_client.subscribe(
            TopicArn=topic_arn,
            Protocol='sqs',
            Endpoint=queue_arn,
            Attributes=attributes
        )

        subscription_arn = response['SubscriptionArn']
        logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
        return subscription_arn

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
        raise
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API dell'SDK for Python (Boto3)AWS .

Ruby

SDK for Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail 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
end
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per Ruby](#).
- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Ruby .

Rust

SDK per 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](#).

Sottoscrive un indirizzo e-mail a un argomento.

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

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

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

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

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

    Ok(())
}
```

- Per informazioni dettagliate sull'API, consulta [Sottoscrizione](#) nella documentazione di riferimento degli SDK AWS per l'API Rust.

SAP ABAP

SDK per 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](#).

Sottoscrive un indirizzo e-mail a un argomento.

```

TRY.
    oo_result = lo_sns->subscribe(
        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_snssubscriptionlmt00.
    MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
ENDTRY.

```

- Per informazioni dettagliate sull'API, consulta [Subscribe](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Sottoscrive un indirizzo e-mail a un argomento.

```

import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

let output = try await snsClient.subscribe(
    input: SubscribeInput(
        endpoint: email,

```

```
        protocol: "email",
        returnSubscriptionArn: true,
        topicArn: arn
    )
)

guard let subscriptionArn = output.subscriptionArn else {
    print("No subscription ARN received from Amazon SNS.")
    return
}

print("Subscription \(subscriptionArn) created.")
```

Per sottoscrivere un numero di telefono a un argomento per ricevere le notifiche via SMS.

```
import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

let output = try await snsClient.subscribe(
    input: SubscribeInput(
        endpoint: phone,
        protocol: "sms",
        returnSubscriptionArn: true,
        topicArn: arn
    )
)

guard let subscriptionArn = output.subscriptionArn else {
    print("No subscription ARN received from Amazon SNS.")
    return
}

print("Subscription \(subscriptionArn) created.")
```

- Per informazioni dettagliate sull'API, consulta [Subscribe](#) nella documentazione di riferimento dell'API AWS SDK per Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **TagResource** con un AWS SDK o una CLI

Gli esempi di codice seguenti mostrano come utilizzare TagResource.

CLI

AWS CLI

Aggiungere un tag a un argomento

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

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

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [TagResource AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sns.SnsClient;  
import software.amazon.awssdk.services.sns.model.SnsException;  
import software.amazon.awssdk.services.sns.model.Tag;  
import software.amazon.awssdk.services.sns.model.TagResourceRequest;  
import java.util.ArrayList;  
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn>

            Where:
                topicArn - The ARN of the topic to which tags are added.

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                .key("Team")
                .value("Development")
                .build();

            Tag tag2 = Tag.builder()
                .key("Environment")
```

```
        .value("Gamma")
        .build();

    List<Tag> tagList = new ArrayList<>();
    tagList.add(tag);
    tagList.add(tag2);

    TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
        .resourceArn(topicArn)
        .tags(tagList)
        .build();

    snsClient.tagResource(tagResourceRequest);
    System.out.println("Tags have been added to " + topicArn);

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

- Per i dettagli sull'API, consulta la [TagResource](#) sezione AWS SDK for Java 2.x API Reference.

Kotlin

SDK per Kotlin

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](#).

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

- Per i dettagli sull'API, [TagResource](#) consulta AWS SDK for Kotlin API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo **Unsubscribe** con un AWS SDK o una CLI

Gli esempi di codice seguenti 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 nei seguenti esempi di codice:

- [Crea un argomento Amazon SNS e pubblica messaggi](#)
- [Pubblicazione di messaggi nelle code](#)

.NET

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Annulla l'iscrizione a un argomento tramite un ARN di sottoscrizione.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

Note

C'è dell'altro 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 informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

CLI

AWS CLI

Annullamento della sottoscrizione a un argomento

Nell'esempio unsubscribe seguente viene eliminata la sottoscrizione specificata a un argomento.

```
aws sns unsubscribe \
  --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Questo comando non produce alcun output.

- Per informazioni dettagliate sull'API, consulta [Unsubscribe](#) nella documentazione di riferimento dei comandi della AWS CLI .

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = ""
```

```

        Usage:    <subscriptionArn>

        Where:
            subscriptionArn - The ARN of the subscription to delete.
            "";

    if (args.length < 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String subscriptionArn = args[0];
    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    unSub(snsClient, subscriptionArn);
    snsClient.close();
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode()
            + "\n\nSubscription was removed for " +
request.subscriptionArn());

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

```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare il client in un modulo separato ed esportarlo.

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

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

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

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

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
  xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
```

```
//    extendedRequestId: undefined,  
//    cfId: undefined,  
//    attempts: 1,  
//    totalRetryDelay: 0  
//  }  
// }  
return response;  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per JavaScript .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun unSub(subscriptionArnVal: String) {  
    val request =  
        UnsubscribeRequest {  
            subscriptionArn = subscriptionArnVal  
        }  
  
    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->  
        snsClient.unsubscribe(request)  
        println("Subscription was removed for ${request.subscriptionArn}")  
    }  
}
```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per Kotlin.

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

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

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

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSClient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

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

    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
                             subscription.arn)
            raise
```

```
class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def unsubscribe(self, subscription_arn: str) -> bool:
        """
        Unsubscribe from an SNS topic.

        :param subscription_arn: The ARN of the subscription to remove.
        :return: True if successful.
        :raises ClientError: If the unsubscribe operation fails.
        """
        try:
            self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

            logger.info(f"Unsubscribed: {subscription_arn}")
            return True

        except ClientError as e:
            error_code = e.response.get('Error', {}).get('Code', 'Unknown')

            if error_code == 'NotFound':
                logger.warning(f"Subscription not found: {subscription_arn}")
                return True # Already unsubscribed
            else:
```

```
logger.error(f"Error unsubscribing: {error_code} - {e}")
raise
```

- Per informazioni dettagliate sull'API, consulta [Annullamento della sottoscrizione](#) nella documentazione di riferimento dell'API SDK for Python (Boto3)AWS .

SAP ABAP

SDK per 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 informazioni dettagliate sull'API, consulta [Unsubscribe](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSSNS

let config = try await SNSClient.SNSClientConfiguration(region: region)
let snsClient = SNSClient(config: config)

_ = try await snsClient.unsubscribe(
    input: UnsubscribeInput(
        subscriptionArn: arn
    )
)

print("Unsubscribed.")
```

- Per informazioni dettagliate sull'API, consulta [Unsubscribe](#) nella documentazione di riferimento dell'API AWS SDK per Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l'utilizzo di Amazon SNS AWS SDKs

I seguenti esempi di codice mostrano come implementare scenari comuni in Amazon SNS con AWS SDKs. Questi scenari illustrano come eseguire attività specifiche chiamando più funzioni all'interno di Amazon SNS o in combinazione con altri Servizi AWS. Ogni scenario include un collegamento al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice.

Gli scenari sono relativi a 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 di piattaforma per le notifiche push di Amazon SNS utilizzando un SDK AWS](#)
- [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 un argomento Amazon SNS e pubblica messaggi](#)
- [Crea e pubblica su un argomento FIFO Amazon SNS utilizzando un SDK AWS](#)
- [Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS](#)
- [Guida introduttiva a IoT Device Defender](#)
- [Guida introduttiva a Config](#)
- [Pubblica messaggi SMS su un argomento Amazon SNS utilizzando un SDK AWS](#)
- [Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un SDK AWS](#)
- [Pubblica un messaggio di testo SMS Amazon SNS utilizzando un SDK AWS](#)
- [Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS](#)
- [Utilizzo di Gateway API per invocare una funzione Lambda](#)
- [Utilizzo degli eventi pianificati per invocare una funzione Lambda](#)

Costruisci un'applicazione per inviare dati a una tabella DynamoDB

Gli esempi di codice riportati di seguito mostrano come costruire un'applicazione che invia dati a una tabella Amazon DynamoDB e che ti avvisa quando un utente aggiorna la tabella.

Java

SDK per Java 2.x

Mostra come creare un'applicazione web dinamica che invia dati utilizzando l'API Java di Amazon DynamoDB e invia un messaggio di testo utilizzando l'API Java di Amazon Simple Notification Service.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

Questo esempio mostra come creare un'app che consenta agli utenti di inviare dati a una tabella Amazon DynamoDB e un messaggio di testo all'amministratore utilizzando Amazon Simple Notification Service (Amazon SNS).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Questo esempio è anche disponibile nella [Guida per lo sviluppatore di AWS SDK per JavaScript v3](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

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

SDK per .NET

Mostra come utilizzare l'API .NET di Amazon Simple Notification Service per creare un'applicazione web con funzionalità di sottoscrizione e pubblicazione. Inoltre, questa applicazione di esempio traduce anche i messaggi.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Simple Notification Service per creare un'applicazione web con funzionalità di sottoscrizione e pubblicazione. Inoltre, questa applicazione di esempio traduce anche i messaggi.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per il codice sorgente completo e le istruzioni su come configurare ed eseguire l'esempio che utilizza l'API Java Async, vedi l'esempio completo su [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Kotlin

SDK per Kotlin

Mostra come utilizzare l'API Kotlin di Amazon SNS per creare un'applicazione con funzionalità di sottoscrizione e pubblicazione. Inoltre, questa applicazione di esempio traduce anche i messaggi.

Per il codice sorgente completo e le istruzioni su come creare un'app web, guarda l'esempio completo su [GitHub](#)

Per il codice sorgente completo e le istruzioni su come creare un'app Android nativa, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon SNS
- Amazon Translate

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Crea un endpoint di piattaforma per le notifiche push di Amazon SNS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come creare un endpoint di piattaforma per notifiche push di Amazon SNS.

CLI

AWS CLI

Creazione di un endpoint dell'applicazione della piattaforma

Nell'esempio `create-platform-endpoint` seguente viene creato un endpoint per l'applicazione della piattaforma indicata utilizzando il token specificato.

```
aws sns create-platform-endpoint \  
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/  
MyApplication \  
  --token EXAMPLE12345...
```

Output:

```
{  
  "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/  
MyApplication/12345678-abcd-9012-efgh-345678901234"
```

```
}
```

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

        Usage:      <token> <platformApplicationArn>
```

Where:

token - The 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 delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette

L'esempio di codice seguente mostra come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

.NET

SDK per .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

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

C++

SDK per C++

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Java

SDK per Java 2.x

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#)

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Kotlin

SDK per Kotlin

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (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

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Rust

SDK per Rust

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione Amazon Textract explorer

Gli esempi di codice seguenti mostrano come esplorare l'output di Amazon Textract tramite un'applicazione interattiva.

JavaScript

SDK per JavaScript (v3)

Mostra come utilizzare per AWS SDK per 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 SQS) sottoscritta a un argomento 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
- Simple Storage Service (Amazon S3)
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK per Python (Boto3)

Mostra come utilizzarlo AWS SDK per 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 (Amazon SNS) al termine del processo.
- Esegue il polling di una coda Amazon Simple Queue Service (Amazon SQS) per un messaggio di completamento del processo e visualizza i risultati.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon Cognito Identity
- Simple Storage Service (Amazon S3)
- Amazon SNS
- Amazon SQS
- Amazon Textract

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Crea un argomento Amazon SNS e pubblica messaggi

L'esempio di codice seguente mostra come:

- Creare un argomento Amazon SNS.
- Sottoscrivere un endpoint di posta elettronica all'argomento
- Verificare il tuo abbonamento
- Pubblica un messaggio sull'argomento
- Eseguire la pulizia delle risorse

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# Amazon SNS Getting Started Script
# This script demonstrates how to create an SNS topic, subscribe to it, publish a
# message,
# and clean up resources.

set -euo pipefail

# Set up logging with secure file permissions
LOG_FILE="sns-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1
```

```

echo "Starting Amazon SNS Getting Started Tutorial..."
echo "$(date)"
echo "===== "

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    if [ -n "${SUBSCRIPTION_ARN:-}" ] && [ "$SUBSCRIPTION_ARN" != "pending
confirmation" ] && [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ]; then
        echo "Deleting subscription: $SUBSCRIPTION_ARN"
        if ! aws sns unsubscribe --subscription-arn "$SUBSCRIPTION_ARN" --region
"$AWS_REGION" 2>/dev/null; then
            echo "Warning: Failed to delete subscription" >&2
        fi
    fi

    if [ -n "${TOPIC_ARN:-}" ]; then
        echo "Deleting topic: $TOPIC_ARN"
        if ! aws sns delete-topic --topic-arn "$TOPIC_ARN" --region "$AWS_REGION"
2>/dev/null; then
            echo "Warning: Failed to delete topic" >&2
        fi
    fi

    return $exit_code
}

# Validate AWS region
AWS_REGION="${AWS_REGION:-us-east-1}"
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Set trap to cleanup on exit

```

```
trap cleanup_resources EXIT

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed or not in PATH"
fi

if ! command -v jq &> /dev/null; then
    handle_error "jq is not installed or not in PATH"
fi

if ! aws sts get-caller-identity --region "$AWS_REGION" &> /dev/null; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random topic name suffix using secure method
RANDOM_SUFFIX=$(openssl rand -hex 4)
TOPIC_NAME="my-topic-${RANDOM_SUFFIX}"

# Validate topic name length (max 256 characters)
if [ ${#TOPIC_NAME} -gt 256 ]; then
    handle_error "Topic name exceeds maximum length of 256 characters"
fi

# Step 1: Create an SNS topic
echo "Creating SNS topic: $TOPIC_NAME"
TOPIC_RESULT=$(aws sns create-topic \
    --name "$TOPIC_NAME" \
    --region "$AWS_REGION" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
    --output json) || handle_error "Failed to create SNS topic"

# Extract the topic ARN using jq for reliable parsing
TOPIC_ARN=$(echo "$TOPIC_RESULT" | jq -r '.TopicArn // empty') || handle_error
"Failed to parse topic result"

if [ -z "$TOPIC_ARN" ]; then
    handle_error "Failed to extract topic ARN from result: $TOPIC_RESULT"
fi

# Validate ARN format
if [[ ! "$TOPIC_ARN" =~ ^arn:aws:sns:[a-z0-9-]+:[0-9]{12}:[a-zA-Z0-9_-]+$ ]];
then
```



```

    handle_error "Invalid SNS topic ARN format: $TOPIC_ARN"
fi

echo "Successfully created topic with ARN: $TOPIC_ARN"

# Step 2: Subscribe to the topic using Email-JSON protocol to reduce costs
echo ""
echo "======"
echo "EMAIL SUBSCRIPTION"
echo "======"
EMAIL_ADDRESS="test-${RANDOM_SUFFIX}@example.com"

# Validate email format (basic validation)
if [[ ! "$EMAIL_ADDRESS" =~ ^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]];
then
    handle_error "Invalid email format: $EMAIL_ADDRESS"
fi

echo "Subscribing email: $EMAIL_ADDRESS to topic using Email-JSON protocol"
SUBSCRIPTION_RESULT=$(aws sns subscribe \
    --topic-arn "$TOPIC_ARN" \
    --protocol email-json \
    --notification-endpoint "$EMAIL_ADDRESS" \
    --region "$AWS_REGION" \
    --output json) || handle_error "Failed to create subscription"

# Extract the subscription ARN using jq
SUBSCRIPTION_ARN=$(echo "$SUBSCRIPTION_RESULT" | jq -r '.SubscriptionArn //
    empty') || handle_error "Failed to parse subscription result"

echo "Subscription created: $SUBSCRIPTION_ARN"
echo "A confirmation email has been sent to $EMAIL_ADDRESS"
echo ""

# Tag the subscription
if [ "$SUBSCRIPTION_ARN" != "PendingConfirmation" ] && [ "$SUBSCRIPTION_ARN" !=
    "pending confirmation" ]; then
    aws sns tag-resource \
        --resource-arn "$SUBSCRIPTION_ARN" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-simple-
notification-service-gs \
        --region "$AWS_REGION" 2>/dev/null || echo "Warning: Failed to tag
    subscription"
fi

```

```
# Step 3: List subscriptions to verify
echo "Listing subscriptions for topic: $TOPIC_ARN"
SUBSCRIPTIONS=$(aws sns list-subscriptions-by-topic --topic-arn "$TOPIC_ARN"
  --region "$AWS_REGION" --output json) || handle_error "Failed to list
  subscriptions"

echo "Current subscriptions:"
echo "$SUBSCRIPTIONS" | jq '.'

# Get the confirmed subscription ARN with optimized jq query and improved error
  handling
CONFIRMED_SUBSCRIPTION=$(echo "$SUBSCRIPTIONS" | jq -r '.Subscriptions[]? |
  select(.SubscriptionArn != "PendingConfirmation") | .SubscriptionArn' 2>/dev/
  null | head -n 1)

if [ -n "$CONFIRMED_SUBSCRIPTION" ]; then
  SUBSCRIPTION_ARN="$CONFIRMED_SUBSCRIPTION"
else
  echo "Warning: No confirmed subscription found. You may not have confirmed
  the subscription yet."
  echo "The script will continue, but you may not receive the test message."
fi

# Step 4: Publish a message to the topic
echo ""
echo "Publishing a test message to the topic"
MESSAGE="Hello from Amazon SNS! This is a test message sent at $(date)."
```

```
# Validate message length (max 256 KB for SNS)
if [ ${#MESSAGE} -gt 262144 ]; then
  handle_error "Message exceeds maximum size of 256 KB"
fi

PUBLISH_RESULT=$(aws sns publish \
  --topic-arn "$TOPIC_ARN" \
  --message "$MESSAGE" \
  --region "$AWS_REGION" \
  --output json) || handle_error "Failed to publish message"

MESSAGE_ID=$(echo "$PUBLISH_RESULT" | jq -r '.MessageId // empty') ||
  handle_error "Failed to parse publish result"

if [ -z "$MESSAGE_ID" ]; then
```

```

    handle_error "No message ID returned from publish operation"
fi

# Validate message ID format (UUID v4)
if [[ ! "$MESSAGE_ID" =~ ^[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}$ ]]; then
    handle_error "Unexpected message ID format: $MESSAGE_ID"
fi

echo "Message published successfully with ID: $MESSAGE_ID"
echo "Check your email for the message."

# Pause to allow the user to check their email
echo ""
echo "Pausing for 3 seconds to allow message delivery..."
sleep 3

# Step 5: Clean up resources
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Resources created:"
echo "- SNS Topic: $TOPIC_ARN"
echo "- Subscription: ${SUBSCRIPTION_ARN:-N/A}"
echo ""
echo "Cleaning up resources to avoid unnecessary charges..."

echo ""
echo "Tutorial completed successfully!"
echo "$(date)"
echo "=====

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [CreateTopic](#)
 - [DeleteTopic](#)
 - [ListSubscriptionsByTopic](#)
 - [Pubblicare](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Crea e pubblica su un argomento FIFO Amazon SNS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come creare un argomento FIFO Amazon SNS e pubblicare un messaggio su di esso.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

In questo esempio

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

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

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

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
```

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

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

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

    // Create queues.
    createQueues(queues);

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

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

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

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

    // Clean up resources.
```

```

        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

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

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
            SNS FIFO
            // topic.

```

```
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

    try {
        // Create and publish a message that updates the wholesale price.
        String subject = "Price Update";
        String dedupId = UUID.randomUUID().toString();
        String attributeName = "business";
        String attributeValue = "wholesale";

        MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
            .dataType("String")
            .stringValue(attributeValue)
            .build();

        Map<String, MessageAttributeValue> attributes = new HashMap<>();
        attributes.put(attributeName, msgAttValue);
        PublishRequest pubRequest = PublishRequest.builder()
            .topicArn(topicArn)
            .subject(subject)
            .message(payload)
            .messageGroupId(groupId)
            .messageDeduplicationId(dedupId)
            .messageAttributes(attributes)
            .build();

        final PublishResponse response = snsClient.publish(pubRequest);
        System.out.println(response.messageId());
        System.out.println(response.sequenceNumber());
        System.out.println("Message was published to " + topicArn);

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CreateTopic](#)
 - [Pubblicare](#)
 - [Subscribe](#)

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un argomento FIFO, sottoscrivi una coda FIFO di Amazon SQS all'argomento e pubblica un messaggio su un argomento Amazon SNS.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
```



```
        "ReceiveMessageWaitTimeSeconds": str(10),
        "VisibilityTimeout": str(300),
        "FifoQueue": str(True),
        "ContentBasedDeduplication": str(True),
    },
)
queues.add(wholesale_queue)
print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

retail_queue = sqs.create_queue(
    QueueName=prefix + "retail.fifo",
    Attributes={
        "MaximumMessageSize": str(4096),
        "ReceiveMessageWaitTimeSeconds": str(10),
        "VisibilityTimeout": str(300),
        "FifoQueue": str(True),
        "ContentBasedDeduplication": str(True),
    },
)
queues.add(retail_queue)
print(f"Created FIFO queue with URL: {retail_queue.url}.")

analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
queues.add(analytics_queue)
print(f"Created standard queue with URL: {analytics_queue.url}.")

topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

for q in queues:
    fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

for q in queues:
    sub = fifo_topic_wrapper.subscribe_queue_to_topic(
        topic, q.attributes["QueueArn"]
    )
    subscriptions.add(sub)

print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

input("Press Enter to publish a message to the topic.")
```

```

message_id = fifo_topic_wrapper.publish_price_update(
    topic, '{"product": 214, "price": 79.99}', "Consumables"
)

print(f"Published price update with message ID: {message_id}.")

# Clean up the subscriptions, queues, and topic.
input("Press Enter to clean up resources.")
for s in subscriptions:
    sns_wrapper.delete_subscription(s)

sns_wrapper.delete_topic(topic)

for q in queues:
    fifo_topic_wrapper.delete_queue(q)

print(f"Deleted subscriptions, queues, and topic.")

print("Thanks for watching!")
print("-" * 88)

class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
        letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
        characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.

```

```

"""
try:
    topic = self.sns_resource.create_topic(
        Name=topic_name,
        Attributes={
            "FifoTopic": str(True),
            "ContentBasedDeduplication": str(False),
            "FifoThroughputScope": "MessageGroup",
        },
    )
    logger.info("Created FIFO topic with name=%s.", topic_name)
    return topic
except ClientError as error:
    logger.exception("Couldn't create topic with name=%s!", topic_name)
    raise error

@staticmethod
def add_access_policy(queue, topic_arn):
    """
    Add the necessary access policy to a queue, so
    it can receive messages from a topic.

    :param queue: The queue resource.
    :param topic_arn: The ARN of the topic.
    :return: None.
    """
    try:
        queue.set_attributes(
            Attributes={
                "Policy": json.dumps(
                    {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Sid": "test-sid",
                                "Effect": "Allow",
                                "Principal": {"AWS": "*"},
                                "Action": "SQS:SendMessage",
                                "Resource": queue.attributes["QueueArn"],
                                "Condition": {
                                    "ArnLike": {"aws:SourceArn": topic_arn}
                                }
                            }
                        ],
                    }
                )
            }
        )

```

```

        ],
    }
)
}

logger.info("Added trust policy to the queue.")
except ClientError as error:
    logger.exception("Couldn't add trust policy to the queue!")
    raise error

@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error

@staticmethod
def publish_price_update(topic, payload, group_id):
    """
    Compose and publish a message that updates the wholesale price.

    :param topic: The topic to publish to.
    :param payload: The message to publish.
    :param group_id: The group ID for the message.
    :return: The ID of the message.
    """
    try:

```

```

        att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
        dedup_id = uuid.uuid4()
        response = topic.publish(
            Subject="Price Update",
            Message=payload,
            MessageAttributes=att_dict,
            MessageGroupId=group_id,
            MessageDeduplicationId=str(dedup_id),
        )
        message_id = response["MessageId"]
        logger.info("Published message to topic %s.", topic.arn)
    except ClientError as error:
        logger.exception("Couldn't publish message to topic %s.", topic.arn)
        raise error
    return message_id

@staticmethod
def delete_queue(queue):
    """
    Removes an SQS queue. When run against an AWS account, it can take up to
    60 seconds before the queue is actually deleted.

    :param queue: The queue to delete.
    :return: None
    """
    try:
        queue.delete()
        logger.info("Deleted queue with URL=%s.", queue.url)
    except ClientError as error:
        logger.exception("Couldn't delete queue with URL=%s!", queue.url)
        raise error

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CreateTopic](#)
 - [Pubblicare](#)

- [Subscribe](#)

SAP ABAP

SDK per 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 argomento FIFO, sottoscrivi una coda Amazon SQS FIFO all'argomento e pubblica un messaggio su un argomento Amazon SNS.

```
" Creates a FIFO topic. "
DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmmap_w=>tt_topicattributesmap.
DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmmap_w=>ts_topicattributesmap_maprow.
ls_tpc_attributes-key = 'FifoTopic'.
ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmmap_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_snssubscriptionlmt00.
    MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
ENDTRY.

" Publish message to SNS topic. "
TRY.
    DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
    DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
    ls_msg_attributes-key = 'Importance'.
    ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
'String'

iv_stringvalue = 'High' ).
    INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

    DATA(lo_result) = lo_sns->publish(
        iv_topicarn = lv_topic_arn
        iv_message = 'The price of your mobile plan has been increased from
$19 to $23'
        iv_subject = 'Changes to mobile plan'
        iv_messagegroupid = 'Update-2'
        iv_messagededuplicationid = 'Update-2.1'
        it_messageattributes = lt_msg_attributes ).
    ov_message_id = lo_result->get_messageid( ).
"
ov_message_id is returned for testing purposes. "
    MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.

```

```
ENDTRY.
```

- Per informazioni dettagliate sull'API, consulta gli argomenti seguenti nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.
 - [CreateTopic](#)
 - [Pubblicare](#)
 - [Subscribe](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come rilevare persone e oggetti in un video con Amazon Rekognition.

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che rileva volti e oggetti nei video situati in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES
- Amazon SNS

- Amazon SQS

Python

SDK per Python (Boto3)

Usa Amazon Rekognition per rilevare volti, oggetti e persone nei video avviando processi di rilevamento asincrono. Questo esempio, inoltre, configura Amazon Rekognition per notificare un argomento Amazon Simple Notification Service (Amazon SNS) al completamento dei processi e sottoscrive una coda Amazon Simple Queue Service (Amazon SQS) all'argomento. Quando la coda riceve un messaggio su un processo, questo viene recuperato e vengono restituiti i risultati.

Questo esempio è visualizzato al meglio su GitHub. Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES
- Amazon SNS
- Amazon SQS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva a IoT Device Defender

L'esempio di codice seguente mostra come:

- Crea ruoli IAM richiesti
- Abilita i controlli di controllo di IoT Device Defender
- Esecuzione di un audit on demand
- Crea un'azione di mitigazione
- Applica azioni di mitigazione ai risultati

- Configura le notifiche SNS (opzionale)
- Abilita la registrazione IoT

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
# checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "AWS IoT Device Defender Getting Started Script"
echo "====="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
}
```

```
        return 0
    }

    # Function to safely extract JSON values using jq
    extract_json_value() {
        local json="$1"
        local key="$2"
        echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
    }

    # Function to validate JSON
    validate_json() {
        local json="$1"
        echo "$json" | jq empty 2>/dev/null
    }

    # Function to check AWS CLI availability
    check_aws_cli() {
        if ! command -v aws &> /dev/null; then
            echo "ERROR: AWS CLI is not installed or not in PATH"
            return 1
        fi
        if ! command -v jq &> /dev/null; then
            echo "ERROR: jq is not installed or not in PATH"
            return 1
        fi
        return 0
    }

    # Function to get AWS account ID
    get_account_id() {
        local account_id
        account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/dev/null) || true
        if [ -z "$account_id" ]; then
            echo "ERROR: Could not retrieve AWS account ID"
            return 1
        fi
        echo "$account_id"
        return 0
    }

    # Function to create IAM roles with retry logic
    create_iam_role() {
```

```
local ROLE_NAME=$1
local TRUST_POLICY=$2
local MANAGED_POLICY=$3
local RETRY_COUNT=0
local MAX_RETRIES=3

echo "Creating IAM role: $ROLE_NAME"

# Validate trust policy JSON
if ! validate_json "$TRUST_POLICY"; then
    echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
    return 1
fi

# Check if role already exists
if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
    echo "Role $ROLE_NAME already exists, skipping creation"
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
--output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
fi

# Create the role with trust policy and retry logic
while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
    ROLE_RESULT=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

    if check_error "$ROLE_RESULT"; then
        break
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))
    if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
        echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
        sleep $((RETRY_COUNT * 2))
    fi
done
```

```

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
    policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

    if ! validate_json "$LOGGING_POLICY"; then
        echo "ERROR: Invalid logging policy JSON"
        return 1
    fi

    POLICY_RESULT=$(aws iam put-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-name "${ROLE_NAME}Policy" \
        --policy-document "$LOGGING_POLICY" 2>&1) || true

```

```

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \

```

```

        --policy-name "${ROLE_NAME}Policy" \
        --policy-document "$MITIGATION_POLICY" 2>&1) || true

    if ! check_error "$POLICY_RESULT"; then
        echo "Failed to attach inline policy to role $ROLE_NAME"
        return 1
    fi
else
    # Attach managed policy to role if provided
    if [ -n "$MANAGED_POLICY" ]; then
        ATTACH_RESULT=$(aws iam attach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$MANAGED_POLICY" 2>&1) || true

        if ! check_error "$ATTACH_RESULT"; then
            echo "Failed to attach policy to role $ROLE_NAME"
            return 1
        fi
    fi
fi

echo "Role $ROLE_NAME created successfully"

# Get the role ARN with error handling
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
    return 1
fi
echo "Role ARN: $ROLE_ARN"
return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

```

```

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "=====
echo "Step 1: Creating required IAM roles"
echo "=====

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
"$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
  echo "ERROR: Failed to create audit role"
  exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"

```



```

        },
        "Action": "sts:AssumeRole"
    }
]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
"$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

```

```

# Step 2: Enable audit checks
echo ""
echo "=====
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "=====

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
|| true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
    configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
    --role-arn "$AUDIT_ROLE_ARN" \
    --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "=====
echo "Step 3: Running an on-demand audit"
echo "=====

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
    --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

```

```
if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:${aws configure get region}:
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))

    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
```

```
        echo "WARNING: Audit task failed, continuing with script..."
        FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
"taskStatistics.failedChecksNotApplicable")
        if [ -n "$FAILURE_REASON" ]; then
            echo "Reason: $FAILURE_REASON"
        fi
        break
    fi
else
    echo "WARNING: Could not parse task details, retrying..."
fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
    --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi
```

```

# Step 4: Create a mitigation action
echo ""
echo "=====
echo "Step 4: Creating a mitigation action"
echo "=====

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
    "enableIoTLoggingParams": {
        "roleArnForLogging": "$LOGGING_ROLE_ARN",
        "logLevel": "ERROR"
    }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
    --action-name "EnableErrorLoggingAction" \
    --role-arn "$MITIGATION_ROLE_ARN" \
    --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

```

```

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
        Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
    || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
    have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "=====
    echo "Step 5: Applying mitigation action to findings"
    echo "=====

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
    "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
    "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}

```

```

EOF
)

if ! validate_json "$AUDIT_CHECK_MAPPING"; then
    echo "ERROR: Invalid audit check mapping JSON"
    exit 1
fi

MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
    --task-id "$MITIGATION_TASK_ID" \
    --target "$TARGET_JSON" \
    --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
|| true

if ! check_error "$MITIGATION_TASK_RESULT"; then
    echo "WARNING: Failed to start mitigation actions task, continuing..."
else
    echo "Mitigation actions task started successfully"
    CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
fi
else
    echo ""
    echo "======"
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "======"
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "======"
echo "Step 6: Setting up SNS notifications"
echo "======"

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn | contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"

```

```

else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
    output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi

    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
        --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
notifications"
    fi

```



```

        else
            echo "SNS notifications enabled successfully"
        fi
    else
        echo "Skipping SNS configuration due to topic creation failure"
    fi

# Step 7: Enable AWS IoT logging
echo ""
echo "=====
echo "Step 7: Enabling AWS IoT logging"
echo "=====

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
    --role-arn "$LOGGING_ROLE_ARN" \
    --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
    "roleArn": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
        --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true

    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi

```

```

else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
"$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "=====
echo "AWS IoT Device Defender setup completed successfully!"
echo "=====
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "=====
echo "CLEANUP"
echo "=====
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."

DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
    --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else

```

```

    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
      echo "V1 logging disabled successfully"
    else
      echo "WARNING: Could not disable logging"
    fi
  fi

  # Delete mitigation action
  echo "Deleting mitigation action..."
  aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
    true

  # Reset audit configuration
  echo "Resetting IoT Device Defender audit configuration..."
  RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
  aws iot update-account-audit-configuration \
    --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

  # Delete SNS topic
  echo "Deleting SNS topic..."
  if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
  fi

  # Clean up IAM roles with improved error handling
  echo "Cleaning up IAM roles..."

  cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then

```

```

    ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
    if validate_json "$ROLE_POLICIES"; then
        while IFS= read -r policy_name; do
            if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                echo "  Deleting inline policy: $policy_name"
                aws iam delete-role-policy \
                    --role-name "$role_name" \
                    --policy-name "$policy_name" 2>&1 || true
            fi
        done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
|| echo "")
    fi

    ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
"$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
    if validate_json "$ATTACHED_POLICIES"; then
        while IFS= read -r policy_arn; do
            if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                echo "  Detaching managed policy: $policy_arn"
                aws iam detach-role-policy \
                    --role-name "$role_name" \
                    --policy-arn "$policy_arn" 2>&1 || true
            fi
        done < <(echo "$ATTACHED_POLICIES" | jq -r
'.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
    fi

    echo "  Deleting role: $role_name"
    aws iam delete-role --role-name "$role_name" 2>&1 || true
else
    echo "  Role $role_name does not exist or already deleted"
fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"

```

```
echo "Log file: $LOG_FILE"
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [AttachRolePolicy](#)
 - [CreateMitigationAction](#)
 - [CreateRole](#)
 - [CreateTopic](#)
 - [DeleteAccountAuditConfiguration](#)
 - [DeleteMitigationAction](#)
 - [DeleteRole](#)
 - [DeleteRolePolicy](#)
 - [DeleteTopic](#)
 - [DescribeAccountAuditConfiguration](#)
 - [DescribeAuditTask](#)
 - [DetachRolePolicy](#)
 - [GetLoggingOptions](#)
 - [GetRole](#)
 - [GetV2 LoggingOptions](#)
 - [ListAuditFindings](#)
 - [ListAuditMitigationActionsTasks](#)
 - [ListMitigationActions](#)
 - [ListRolePolicies](#)
 - [ListTopics](#)
 - [PutRolePolicy](#)
 - [SetLoggingOptions](#)
 - [Imposta V2 LoggingOptions](#)
 - [StartAuditMitigationActionsTask](#)
 - [StartOnDemandAuditTask](#)
 - [UpdateAccountAuditConfiguration](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva a Config

L'esempio di codice seguente mostra come:

- Crea un bucket Amazon S3
- Creare un argomento Amazon SNS.
- Creare un ruolo IAM per Config
- Configurare il registratore di configurazione Config
- Configura il canale di distribuzione Config
- Avvia il registratore di configurazione
- Verifica la configurazione di Config

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1
```

```
# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
```

```

        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
echo "CREATED RESOURCES"

```



```

    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"

```

```

fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-`${RANDOM_ID}`"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config

```

```
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action": [
        "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
    "Condition": {
        "StringLike": {
            "s3:x-amz-acl": "bucket-owner-full-control"
        }
    }
},
{
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketAcl"
    ],
    "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
},
{
    "Effect": "Allow",
    "Action": [
        "sns:Publish"
    ],
    "Resource": "${SNS_TOPIC_ARN}"
}
]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."

```

```

EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",
    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"

```

```

fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",
    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Six_Hours"
    }
}
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",
    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",

```

```
"configSnapshotDeliveryProperties": {
  "deliveryFrequency": "Six_Hours"
}
}
EOF

echo "Creating delivery channel..."
CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file:///deliveryChannel.json)
check_command "$CHANNEL_RESULT"
echo "Delivery channel created"

aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
  echo "Configuration recorder is already running."
else
  echo "Starting configuration recorder..."
  START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
  check_command "$START_RESULT"
  echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
```

```
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "====="
echo "CLEANUP CONFIRMATION"
echo "====="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateRole](#)
 - [CreateTopic](#)
 - [DeleteBucket](#)
 - [DeleteConfigurationRecorder](#)
 - [DeleteDeliveryChannel](#)
 - [DeleteRole](#)
 - [DeleteRolePolicy](#)
 - [DeleteTopic](#)
 - [DescribeConfigurationRecorderStatus](#)

- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica messaggi SMS su un argomento Amazon SNS utilizzando un SDK AWS

Il codice di esempio seguente mostra come fare per:

- Creazione di un argomento Amazon SNS.
- Sottoscrivere un numero di telefono cellulare all'argomento.
- Pubblicazione di messaggi SMS nell'argomento in modo che tutti i numeri di telefono sottoscritti ricevano il messaggio in una sola volta.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare un argomento e restituire il suo ARN.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }
}
```

```

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}

```

Sottoscrivere un endpoint a un argomento.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicArn> <phoneNumber>

```

```

        Where:
            topicArn - The ARN of the topic to subscribe.
            phoneNumber - A mobile phone number that receives
notifications (for example, +1XXX5550100).
        """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                .protocol("sms")
                .endpoint(phoneNumber)
                .returnSubscriptionArn(true)
                .topicArn(topicArn)
                .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
"\n\n Status is "
                + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Impostare gli attributi del messaggio, ad esempio l'ID del mittente, il prezzo massimo e il relativo tipo. Gli attributi del messaggio sono facoltativi.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                .attributes(attributes)
                .build();

            SetSmsAttributesResponse result =
snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
Status was ")
        }
    }
}
```

```

        + result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

Publicare un messaggio in un argomento. Il messaggio viene inviato a ogni abbonato.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is
sent (for example, +1XXX5550100).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}

```

```
String message = args[0];
String phoneNumber = args[1];
SnsClient snsClient = SnsClient.builder()
    .region(Region.US_EAST_1)
    .build();
pubTextSMS(snsClient, message, phoneNumber);
snsClient.close();
}

public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .phoneNumber(phoneNumber)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica un messaggio di grandi dimensioni su Amazon SNS con Amazon S3 utilizzando un SDK AWS

L'esempio di codice seguente mostra come pubblicare un messaggio di grandi dimensioni su Amazon SNS utilizzando Amazon S3 per archiviare il payload del messaggio.

Java

SDK per Java 1.x

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Per pubblicare un messaggio di grandi dimensioni, utilizza Amazon SNS Extended Client Library per Java. Il messaggio che invii fa riferimento a un oggetto Amazon S3 contenente il contenuto effettivo del messaggio.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

    public static void main(String[] args) {
        final String BUCKET_NAME = "extended-client-bucket";
        final String TOPIC_NAME = "extended-client-topic";
        final String QUEUE_NAME = "extended-client-queue";
        final Regions region = Regions.DEFAULT_REGION;
```



```
// Message threshold controls the maximum message size that will
be allowed to
// be published
// through SNS using the extended client. Payload of messages
exceeding this
// value will be stored in
// S3. The default value of this parameter is 256 KB which is the
maximum

// message size in SNS (and SQS).
final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

// Initialize SNS, SQS and S3 clients
final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

// Create bucket, topic, queue and subscription
s3Client.createBucket(BUCKET_NAME);
final String topicArn = snsClient.createTopic(
    new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
final String queueUrl = sqsClient.createQueue(
    new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
final String subscriptionArn = Topics.subscribeQueue(
    snsClient, sqsClient, topicArn, queueUrl);

// To read message content stored in S3 transparently through SQS
extended
// client,
// set the RawMessageDelivery subscription attribute to TRUE
final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);
```

```

        // Initialize SNS extended client
        // PayloadSizeThreshold triggers message content storage in S3
when the
        // threshold is exceeded
        // To store all messages content in S3, use AlwaysThroughS3 flag
        final SNSExtendedClientConfiguration
snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

        .withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
        final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                        snsExtendedClientConfiguration);

        // Publish message via SNS with storage in S3
        final String message = "This message is stored in S3 as it
exceeds the threshold of 32 bytes set above.";
        snsExtendedClient.publish(topicArn, message);

        // Initialize SQS extended client
        final ExtendedClientConfiguration sqsExtendedClientConfiguration
= new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
BUCKET_NAME);
        final AmazonSQSExtendedClient sqsExtendedClient = new
AmazonSQSExtendedClient(sqsClient,
                        sqsExtendedClientConfiguration);

        // Read the message from the queue
        final ReceiveMessageResult result =
sqsExtendedClient.receiveMessage(queueUrl);
        System.out.println("Received message is " +
result.getMessages().get(0).getBody());
    }
}

```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica un messaggio di testo SMS Amazon SNS utilizzando un SDK AWS

Negli esempi di codice seguenti viene illustrato come pubblicare messaggi SMS utilizzando Amazon SNS.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
            AmazonSimpleNotificationServiceClient(regionEndpoint);
        }
    }
}
```

```

    /// <summary>
    /// Sends the SMS message passed in the text parameter to the phone
number
    /// in phoneNum.
    /// </summary>
    /// <param name="phoneNum">The ten-digit phone number to which the text
    /// message will be sent.</param>
    /// <param name="text">The text of the message to send.</param>
    /// <returns>Async task.</returns>
    public async Task SendTextMessageAsync(string phoneNum, string text)
    {
        if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
        {
            return;
        }

        // Now actually send the message.
        var request = new PublishRequest
        {
            Message = text,
            PhoneNumber = phoneNum,
        };

        try
        {
            var response = await snsClient.PublishAsync(request);
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error sending message: {ex}");
        }
    }
}

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per .NET .

C++

SDK per C++

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 * SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 * NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 * is in the SMS sandbox and you can only
 * use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
 * latest/dg/sns-sms-sandbox.html.
 * NOTE: If destination is in the US, you also have an additional restriction
 * that you have use a dedicated
 * origination ID (phone number). You can request an origination number using
 * Amazon Pinpoint for a fee.
 * See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
 * origination-numbers-with-amazon-sns/
 * for more information.
 *
 * <phone_number_value> input parameter uses E.164 format.
 * For example, in United States, this input value should be of the form:
 * +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
 \param message: The message to publish.
 \param phoneNumber: The phone number of the recipient in E.164 format.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```

        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                    << outcome.GetResult().GetMessageId() << "'."
                    << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per C++ .

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;

```

```
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <message> <phoneNumber>

            Where:
                message - The message text to send.
                phoneNumber - The mobile phone number to which a message is
sent (for example, +1XXX5550100).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .phoneNumber(phoneNumber)
```

```

        .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

    SnsClient.fromEnvironment { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}

```



```
}  
}
```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per Kotlin.

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'vendor/autoload.php';  
  
use Aws\Exception\AwsException;  
use Aws\Sns\SnsClient;  
  
/**  
 * Sends a text message (SMS message) directly to a phone number using Amazon  
 * SNS.  
 *  
 * This code expects that you have AWS credentials set up per:  
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/  
 * guide_credentials.html  
 */  
  
$SnsClient = new SnsClient([  
    'profile' => 'default',  
    'region' => 'us-east-1',  
    'version' => '2010-03-31'  
]);  
  
$message = 'This message is sent from a Amazon SNS code sample.';  
$phone = '+1XXX5550100';
```

```
try {
    $result = $SnSClient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per PHP](#).
- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API AWS SDK per PHP .

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.
        """
```

```

        :param phone_number: The phone number that receives the message. This
        must be
                                in E.164 format. For example, a United States phone
                                number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id

```

- Per informazioni dettagliate sull'API, consulta [Pubblicazione](#) nella documentazione di riferimento dell'API SDK for Python (Boto3)AWS .

SAP ABAP

SDK per 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](#).

```

" iv_phone_number = '+12065550101' - Phone number in E.164 format
TRY.
    oo_result = lo_sns->publish(
        " oo_result is returned for
testing purposes. "
        iv_phonenumber = iv_phone_number
        iv_message = iv_message ).

```

```
MESSAGE 'Message published to phone number.' TYPE 'I'.  
CATCH /aws1/cx_snsnotfoundexception.  
MESSAGE 'Phone number does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per informazioni dettagliate sull'API, consulta [Publish](#) nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Pubblica messaggi Amazon SNS nelle code Amazon SQS utilizzando un SDK AWS

Gli esempi di codice seguenti mostrano come:

- Creazione di un argomento (FIFO o non FIFO).
- Sottoscrizione di diverse code all'argomento con la possibilità di applicare un filtro.
- Pubblicazione di un messaggio nell'argomento.
- Esame delle code per i messaggi ricevuti.

.NET

SDK per .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 feature 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 scenario 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 scenario.
/// </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 scenario, 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\\n}and queue URL {queueUrl}" +
            $"{r\\n}has 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 messageId = GetUserResponse("Enter a message 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($"{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:" +
            $"{message.Body}");
    }

    Console.WriteLine(new string('-', 80));
    return messages;
}

/// <summary>
/// Delete the message using handles in a batch.
```

```
/// </summary>
/// <returns>Async task.</returns>
public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
    await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <returns>Async task.</returns>
private static async Task CleanupResources()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"Clean up resources.");

    try
    {
        foreach (var queueUrl in _queueUrls)
        {
            if (!string.IsNullOrEmpty(queueUrl))
            {
                var deleteQueue =
                    GetYesNoResponse($"Delete queue with url {queueUrl}?");
                if (deleteQueue)
                {
                    await SqsWrapper.DeleteQueueByUrl(queueUrl);
                }
            }
        }

        foreach (var subscriptionArn in _subscriptionArns)
        {
            if (!string.IsNullOrEmpty(subscriptionArn))
            {
                await SnsWrapper.UnsubscribeByArn(subscriptionArn);
            }
        }
    }
}
```

```

        var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
        if (deleteTopic)
        {
            await SnsWrapper.DeleteTopicByArn(_topicArn);
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
    }

    Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="defaultAnswer">Optional default answer to use.</param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question, bool defaultAnswer =
true)
{
    if (UseConsole)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
            ynResponse.Equals("y",
                StringComparison.InvariantCultureIgnoreCase);

        return response;
    }
    // If not using the console, use the default.
    return defaultAnswer;
}

/// <summary>
/// Helper method to get a string response from the user through the console.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="defaultAnswer">Optional default answer to use.</param>

```



```

    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}

```

Crea una classe che avvolge le operazioni di Amazon SQS.

```

    /// <summary>
    /// Wrapper for Amazon Simple Queue Service (SQS) operations.
    /// </summary>
    public class SQSWrapper
    {
        private readonly IAmazonSQS _amazonSQSClient;

        /// <summary>
        /// Constructor for the Amazon SQS wrapper.
        /// </summary>
        /// <param name="amazonSQS">The injected Amazon SQS client.</param>
        public SQSWrapper(IAmazonSQS amazonSQS)
        {
            _amazonSQSClient = amazonSQS;
        }

        /// <summary>
        /// Create a queue with a specific name.
        /// </summary>
        /// <param name="queueName">The name for the queue.</param>
        /// <param name="useFifoQueue">True to use a FIFO queue.</param>

```

```
/// <returns>The url for the queue.</returns>
public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
{
    int maxMessage = 256 * 1024;
    var queueAttributes = new Dictionary<string, string>
    {
        {
            QueueAttributeName.MaximumMessageSize,
            maxMessage.ToString()
        }
    };

    var createQueueRequest = new CreateQueueRequest()
    {
        QueueName = queueName,
        Attributes = queueAttributes
    };

    if (useFifoQueue)
    {
        // Update the name if it is not correct for a FIFO queue.
        if (!queueName.EndsWith(".fifo"))
        {
            createQueueRequest.QueueName = queueName + ".fifo";
        }

        // Add an attribute for a FIFO queue.
        createQueueRequest.Attributes.Add(
            QueueAttributeName.FifoQueue, "true");
    }

    var createResponse = await _amazonSQSClient.CreateQueueAsync(
        new CreateQueueRequest()
        {
            QueueName = queueName
        });
    return createResponse.QueueUrl;
}

/// <summary>
/// Get the ARN for a queue from its URL.
/// </summary>
/// <param name="queueUrl">The URL of the queue.</param>
```

```

    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
        _amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
    topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                    "\"Service\": " +
                        "\"sns.amazonaws.com\"" +
                    "}," +
                "\"Action\": \"sqs:SendMessage\"," +
                "\"Resource\": \"{queueArn}\"," +
                "\"Condition\": {" +
                    "\"ArnEquals\": {" +
                        "\"aws:SourceArn\":
\\\"{topicArn}\"" +
                    "}" +
                "}" +
            "}]}" +
        "}";
    }

```

```

        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()

```

```

    };
    foreach (var message in messages)
    {
        deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
        {
            ReceiptHandle = message.ReceiptHandle,
            Id = message.MessageId
        });
    }

    var deleteResponse = await
        _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

    return deleteResponse.Failed.Any();
}

/// <summary>
/// Delete a queue by its URL.
/// </summary>
/// <param name="queueUrl">The url of the queue.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteQueueByUrl(string queueUrl)
{
    var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
        new DeleteQueueRequest()
        {
            QueueUrl = queueUrl
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
}

```

Crea una classe che avvolge le operazioni di Amazon SNS.

```

/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;
}

```

```
/// <summary>
/// Constructor for the Amazon SNS wrapper.
/// </summary>
/// <param name="amazonSNS">The injected Amazon SNS client.</param>
public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
{
    _amazonSNSClient = amazonSNS;
}

/// <summary>
/// Create a new topic with a name and specific FIFO and de-duplication
attributes.
/// </summary>
/// <param name="topicName">The name for the topic.</param>
/// <param name="useFifoTopic">True to use a FIFO topic.</param>
/// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
/// <returns>The ARN of the new topic.</returns>
public async Task<string> CreateTopicWithName(string topicName, bool
useFifoTopic, bool useContentBasedDeduplication)
{
    var createTopicRequest = new CreateTopicRequest()
    {
        Name = topicName,
    };

    if (useFifoTopic)
    {
        // Update the name if it is not correct for a FIFO topic.
        if (!topicName.EndsWith(".fifo"))
        {
            createTopicRequest.Name = topicName + ".fifo";
        }

        // Add the attributes from the method parameters.
        createTopicRequest.Attributes = new Dictionary<string, string>
        {
            { "FifoTopic", "true" }
        };
        if (useContentBasedDeduplication)
        {
            createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
        }
    }
}
```

```

    }

    var createResponse = await
    _amazonSNSClient.CreateTopicAsync(createTopicRequest);
    return createResponse.TopicArn;
}

/// <summary>
/// Subscribe a queue to a topic with optional filters.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
/// <param name="queueArn">The ARN of the queue.</param>
/// <returns>The ARN of the new subscription.</returns>
public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
{
    var subscribeRequest = new SubscribeRequest()
    {
        TopicArn = topicArn,
        Protocol = "sqs",
        Endpoint = queueArn
    };

    if (!string.IsNullOrEmpty(filterPolicy))
    {
        subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
    }

    var subscribeResponse = await
    _amazonSNSClient.SubscribeAsync(subscribeRequest);
    return subscribeResponse.SubscriptionArn;
}

/// <summary>
/// Publish a message to a topic with an attribute and optional deduplication
and group IDs.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <param name="message">The message to publish.</param>
/// <param name="attributeName">The optional attribute for the message.</
param>

```

```

    /// <param name="attributeValue">The optional attribute value for the
message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String" } }
                };
        }

        var publishResponse = await
        _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }

    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>

```



```
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Pubblicare](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)

- [Unsubscribe](#)

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
        << NUMBER_OF_QUEUES <<
        " SQS queues to the topic." << std::endl;
    std::cout
        << "You can select from several options for configuring the topic and
the subscriptions for the "
        << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
queues."
        << std::endl;

    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();
```

```

std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
          << std::endl;
std::cout
    << "FIFO topics deliver messages in order and support deduplication
and message filtering."
    << std::endl;
bool isFifoTopic = askYesNoQuestion(
    "Would you like to work with FIFO topics? (y/n) ");

bool contentBasedDeduplication = false;
Aws::String topicName;
if (isFifoTopic) {
    printAsterisksLine();
    std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
              << std::endl;
    std::cout
        << "Deduplication IDs are either set in the message or
automatically generated "
        << "from content using a hash function." << std::endl;
    std::cout
        << "If a message is successfully published to an SNS FIFO topic,
any message "
        << "published and determined to have the same deduplication ID, "
        << std::endl;
    std::cout
        << "within the five-minute deduplication interval, is accepted
but not delivered."
        << std::endl;
    std::cout
        << "For more information about deduplication, "
        << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html."
        << std::endl;
    contentBasedDeduplication = askYesNoQuestion(
        "Use content-based deduplication instead of entering a
deduplication ID? (y/n) ");
}

printAsterisksLine();

Aws::SQS::SQSClient sqsClient(clientConfiguration);
Aws::Vector<Aws::String> queueURLS;
Aws::Vector<Aws::String> subscriptionARNS;

```

```
Aws::String topicARN;
{
    topicName = askQuestion("Enter a name for your SNS topic. ");

    // 1. Create an Amazon SNS topic, either FIFO or non-FIFO.
    Aws::SNS::Model::CreateTopicRequest request;

    if (isFifoTopic) {
        request.AddAttributes("FifoTopic", "true");
        if (contentBasedDeduplication) {
            request.AddAttributes("ContentBasedDeduplication", "true");
        }
        topicName = topicName + FIFO_SUFFIX;

        std::cout
            << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
            << std::endl;
    }

    request.SetName(topicName);

    Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARN = outcome.GetResult().GetTopicArn();
        std::cout << "Your new topic with the name '" << topicName
            << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
        std::cout << "'" << topicARN << "' has been created." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::CreateTopic. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);
    }
}
```

```

        return false;
    }
}

printAsterisksLine();

std::cout << "Now you will create " << NUMBER_OF_QUEUES
           << " SQS queues to subscribe to the topic." << std::endl;
Aws::Vector<Aws::String> queueNames;
bool filteringMessages = false;
bool first = true;
for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
    Aws::String queueURL;
    Aws::String queueName;
    {
        printAsterisksLine();
        std::ostringstream ostream;
        ostream << "Enter a name for " << (first ? "an" : "the next")
                << " SQS queue. ";
        queueName = askQuestion(ostream.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 ostream;
            ostream << "Filter messages for \"" << queueName
                << "\"'s subscription to the topic \""
                << topicName << "\"? (y/n)";

            // Add filter if user answers yes.
            if (askYesNoQuestion(ostream.str())) {
                Aws::String jsonPolicy = getFilterPolicyFromUser();
                if (!jsonPolicy.empty()) {
                    filteringMessages = true;

                    std::cout << "This is the filter policy for this
subscription."
                                << std::endl;
                    std::cout << jsonPolicy << std::endl;

                    request.AddAttributes("FilterPolicy", jsonPolicy);
                }
                else {
                    std::cout
                        << "Because you did not select any attributes, no
filter "
                        << "will be added to this subscription." <<
std::endl;
                }
            }
        }
    }
}

```

```

        } // if (isFifoTopic)
        Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

        if (outcome.IsSuccess()) {
            Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
            std::cout << "The queue '" << queueName
                << "' has been subscribed to the topic '"
                << "'" << topicName << "'" << std::endl;
            std::cout << "with the subscription ARN '" << subscriptionARN <<
". "
                << std::endl;
            subscriptionARNS.push_back(subscriptionARN);
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Subscribe. "
                << outcome.GetError().GetMessage()
                << std::endl;

            cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

            return false;
        }
    }

    first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");
    request.SetMessage(message);
    if (isFifoTopic) {
        if (first) {

```

```

        std::cout
            << "Because you are using a FIFO topic, you must set a
message group ID."
            << std::endl;
        std::cout
            << "All messages within the same group will be received
in the "
            << "order they were published." << std::endl;
    }
    Aws::String messageGroupId = askQuestion(
        "Enter a message group ID for this message. ");
    request.SetMessageGroupId(messageGroupId);
    if (!contentBasedDeduplication) {
        if (first) {
            std::cout
                << "Because you are not using content-based
deduplication, "
                << "you must enter a deduplication ID." << std::endl;
        }
        Aws::String deduplicationID = askQuestion(
            "Enter a deduplication ID for this message. ");
        request.SetMessageDeduplicationId(deduplicationID);
    }
}

if (filteringMessages && askYesNoQuestion(
    "Add an attribute to this message? (y/n) ")) {
    for (size_t i = 0; i < TONES.size(); ++i) {
        std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
    }
    int selection = askQuestionForIntRange(
        "Enter a number for an attribute. ",
        1, static_cast<int>(TONES.size()));
    Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
    messageAttributeValue.SetDataType("String");
    messageAttributeValue.SetStringValue(TONES[selection - 1]);
    request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
}

Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

if (outcome.IsSuccess()) {
    std::cout << "Your message was successfully published." << std::endl;
}

```

```

        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                << outcome.GetError().GetMessage()
                << std::endl;

            cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
        << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7. Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
            SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
                    outcome.GetResult().GetMessages();
                if (newMessages.empty()) {

```

```

        break;
    }
    else {
        for (const Aws::SQS::Model::Message &message: newMessages) {
            messages.push_back(message.GetBody());
            receiptHandles.push_back(message.GetReceiptHandle());
        }
    }
}
else {
    std::cerr << "Error with SQS::ReceiveMessage. "
               << outcome.GetError().GetMessage()
               << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNs,
            snsClient,
            sqsClient);

    return false;
}
}

printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
           << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << "  Message : '" << message << "'."
               << std::endl;
}

// 8. Delete a batch of messages from an SQS queue.
if (!receiptHandles.empty()) {

```

```

        Aws::SQS::Model::DeleteMessageBatchRequest request;
        request.SetQueueUrl(queueURLS[i]);
        int id = 1; // Ids must be unique within a batch delete request.
        for (const Aws::String &receiptHandle: receiptHandles) {
            Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
            entry.SetId(std::to_string(id));
            ++id;
            entry.SetReceiptHandle(receiptHandle);
            request.AddEntries(entry);
        }

        Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
            sqsClient.DeleteMessageBatch(request);

        if (outcome.IsSuccess()) {
            std::cout << "The batch deletion of messages was successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with SQS::DeleteMessageBatch. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient);

            return false;
        }
    }
}

return cleanUp(topicARN,
               queueURLS,
               subscriptionARNS,
               snsClient,
               sqsClient,
               true); // askUser
}

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                       const Aws::Vector<Aws::String> &queueURLS,

```

```

const Aws::Vector<Aws::String>
&subscriptionARNS,

const Aws::SNS::SNSClient &snsClient,
const Aws::SQS::SQSClient &sqsClient,
bool askUser) {

    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                    << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" <<
subscriptionARN
                    << "' was successful." << std::endl;
            }
            else {

```

```

        std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
        << outcome.GetError().GetMessage()
        << std::endl;
        result = false;
    }
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
        << "' was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
        << outcome.GetError().GetMessage()
        << std::endl;
        result = false;
    }
}

return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
from an SNS topic.
/*!
\sa createPolicyForQueue()
\param queueARN: The SQS queue Amazon Resource Name (ARN).
\param topicARN: The SNS topic ARN.
\return Aws::String: The policy as JSON.
*/
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
```



```

const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": ")" << topicARN << R"("
                    }
                }
            }
        ]
    })";

    return policyStream.str();
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per C++ .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Pubblicare](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui uno scenario interattivo al prompt dei comandi.

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"
    "strings"
    "topics_and_queues/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

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
message? (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 scenario, you will create an SNS topic and subscribe %v SQS queues to\n"+
        "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\n"+
        "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))
}
```

Definisce una struttura per il wrapping delle azioni di Amazon SNS utilizzate in questo esempio.

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// 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),
    ReturnSubscription: true,
})
if err != nil {
    log.Printf("Couldn't subscribe 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
}

```

Definisce una struttura per il wrapping delle azioni di Amazon SQS utilizzate in questo esempio.

```

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// 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,
    })
    if err != nil {
        log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
            err)
    }
    return err
}

```



```

    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
}

```

Eliminare le risorse.

```

import (
  "context"
  "fmt"
  "log"
  "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
  topicArn  string
  queueUrls []string
  snsActor  *actions.SnsActions
  sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.

```

```
func (resources Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Println("Something went wrong during cleanup. Use the AWS Management Console\n" +
                "to remove any remaining resources that were created for this scenario.")
        }
    }()

    var err error
    if resources.topicArn != "" {
        log.Printf("Deleting topic %v.\n", resources.topicArn)
        err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
        if err != nil {
            panic(err)
        }
    }

    for _, queueUrl := range resources.queueUrls {
        log.Printf("Deleting queue %v.\n", queueUrl)
        err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
        if err != nil {
            panic(err)
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Go .

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Pubblicare](#)
- [ReceiveMessage](#)

- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.sns;

import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
    software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
```

```
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
```

```
public static final String DASHES = new String(new char[80]).replace("\0",
"-");

public static void main(String[] args) {
    final String usage = "\n" +
        "Usage:\n" +
        "    <fifoQueueARN>\n\n" +
        "Where:\n" +
        "    accountId - Your AWS account Id value.";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    SnsClient snsClient = SnsClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

    SqsClient sqsClient = SqsClient.builder()
        .region(Region.US_EAST_1)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();

    Scanner in = new Scanner(System.in);
    String accountId = args[0];
    String useFIFO;
    String duplication = "n";
    String topicName;
    String deduplicationID = null;
    String groupId = null;

    String topicArn;
    String sqsQueueName;
    String sqsQueueUrl;
    String sqsQueueArn;
    String subscriptionArn;
    boolean selectFIFO = false;

    String message;
    List<Message> messageList;
    List<String> filterList = new ArrayList<>();
    String msgAttValue = "";
```

```

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
deduplication is supported.\n" +
                "            Deduplication IDs are either set in the message or
automatically generated from content using a hash function.\n"
                +
                "            If a message is successfully published to an SNS FIFO
topic, any message published and determined to have the same deduplication ID,
\n"
                +
                "            within the five-minute deduplication interval, is
accepted but not delivered.\n" +
                "            For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
            }
        }
    }
}

```

```
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    }
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a topic.");
System.out.println("Enter a name for your SNS topic.");
topicName = in.nextLine();
if (selectFIFO) {
    System.out.println("Because you have selected a FIFO topic, '.fifo'
must be appended to the topic name.");
    topicName = topicName + ".fifo";
    System.out.println("The name of the topic is " + topicName);
    topicArn = createFIFO(snsClient, topicName, duplication);
    System.out.println("The ARN of the FIFO topic is " + topicArn);

} else {
    System.out.println("The name of the topic is " + topicName);
    topicArn = createSNSTopic(snsClient, topicName);
    System.out.println("The ARN of the non-FIFO topic is " + topicArn);

}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create an SQS queue.");
System.out.println("Enter a name for your SQS queue.");
sqsQueueName = in.nextLine();
if (selectFIFO) {
    sqsQueueName = sqsQueueName + ".fifo";
}
sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
System.out.println("The queue URL is " + sqsQueueUrl);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the SQS queue ARN attribute.");
sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
System.out.println("The ARN of the new queue is " + sqsQueueArn);
System.out.println(DASHES);

System.out.println(DASHES);
```

```

        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = ""
        {
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "sns.amazonaws.com"
                    },
                    "Action": "sqs:SendMessage",
                    "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                    "Condition": {
                        "ArnEquals": {
                            "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                        }
                    }
                }
            ]
        }
        """".formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
messages will be received in the queue.\n"
                +
                "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                +
                "For this example, you can filter messages by a \"tone\"
attribute.");
            System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
                + topicName + "? (y/n)");
        }
    }
}

```



```

        String filterAns = in.nextLine();
        if (filterAns.compareTo("y") == 0) {
            boolean moreAns = false;
            System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
            System.out.println("1. cheerful");
            System.out.println("2. funny");
            System.out.println("3. serious");
            System.out.println("4. sincere");
            while (!moreAns) {
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        filterList.add("cheerful");
                        break;
                    case "2":
                        filterList.add("funny");
                        break;
                    case "3":
                        filterList.add("serious");
                        break;
                    case "4":
                        filterList.add("sincere");
                        break;
                    default:
                        moreAns = true;
                        break;
                }
            }
        }
    }
    subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Publish a message to the topic.");
    if (selectFIFO) {
        System.out.println("Would you like to add an attribute to this
message? (y/n)");
        String msgAns = in.nextLine();
        if (msgAns.compareTo("y") == 0) {
            System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");

```

```
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        System.out.println("Select a number or choose 0 to end.");
        String ans = in.nextLine();
        switch (ans) {
            case "1":
                msgAttValue = "cheerful";
                break;
            case "2":
                msgAttValue = "funny";
                break;
            case "3":
                msgAttValue = "serious";
                break;
            default:
                msgAttValue = "sincere";
                break;
        }

        System.out.println("Selected value is " + msgAttValue);
    }
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

    } else {
        System.out.println("Enter a message.");
        message = in.nextLine();
        pubMessage(snsClient, message, topicArn);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("8. Display the message. Press any key to continue.");
    in.nextLine();
    messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
    for (Message mes : messageList) {
        System.out.println("Message Id: " + mes.messageId());
        System.out.println("Full Message: " + mes.body());
    }
    System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {

```

```
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);
        System.out.println(queueName + " was successfully deleted.");

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("Status was " +
            result.sdkHttpResponse().statusCode()
            + "\nSubscription was removed for " + request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
    List<Message> messages) {
    try {
        List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
        for (Message msg : messages) {
            DeleteMessageBatchRequestEntry entry =
                DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();
```

```
        entries.add(entry);
    }

    DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
        .queueUrl(queueUrl)
        .entries(entries)
        .build();

    sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
    System.out.println("The batch delete of messages was successful");

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
    try {
        if (msgAttValue.isEmpty()) {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .maxNumberOfMessages(5)
                .build();
            return
sqsClient.receiveMessage(receiveMessageRequest).messages();
        } else {
            // We know there are filters on the message.
            ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                .maxNumberOfMessages(5)
                .build();

            return sqsClient.receiveMessage(receiveRequest).messages();
        }
    } catch (SqsException e) {
```

```

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
    try {
        PublishRequest request = PublishRequest.builder()
            .message(message)
            .topicArn(topicArn)
            .build();

        PublishResponse result = snsClient.publish(request);
        System.out
            .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void pubMessageFIFO(SnsClient snsClient,
                                String message,
                                String topicArn,
                                String msgAttValue,
                                String duplication,
                                String groupId,
                                String deduplicationID) {

    try {
        PublishRequest request;
        // Means the user did not choose to use a message attribute.
        if (msgAttValue.isEmpty()) {
            if (duplication.compareTo("y") == 0) {
                request = PublishRequest.builder()
                    .message(message)
                    .messageGroupId(groupId)
                    .topicArn(topicArn)
                    .build();
            } else {

```

```
        request = PublishRequest.builder()
            .message(message)
            .messageDeduplicationId(deduplicationID)
            .messageGroupId(groupId)
            .topicArn(topicArn)
            .build();
    }

    } else {
        Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
        messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
            .dataType("String")
            .stringValue("true")
            .build());

        if (duplication.compareTo("y") == 0) {
            request = PublishRequest.builder()
                .message(message)
                .messageGroupId(groupId)
                .topicArn(topicArn)
                .build();
        } else {
            // Create a publish request with the message and attributes.
            request = PublishRequest.builder()
                .topicArn(topicArn)
                .message(message)
                .messageDeduplicationId(deduplicationID)
                .messageGroupId(groupId)
                .messageAttributes(messageAttributes)
                .build();
        }
    }

    // Publish the message to the topic.
    PublishResponse result = snsClient.publish(request);
    System.out
        .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
```

```
    }  
}  
  
// Subscribe to the SQS queue.  
public static String subQueue(SnsClient snsClient, String topicArn, String  
queueArn, List<String> filterList) {  
    try {  
        SubscribeRequest request;  
        if (filterList.isEmpty()) {  
            // No filter subscription is added.  
            request = SubscribeRequest.builder()  
                .protocol("sqs")  
                .endpoint(queueArn)  
                .returnSubscriptionArn(true)  
                .topicArn(topicArn)  
                .build();  
  
            SubscribeResponse result = snsClient.subscribe(request);  
            System.out.println("The queue " + queueArn + " has been  
subscribed to the topic " + topicArn + "\n" +  
                "with the subscription ARN " + result.subscriptionArn());  
            return result.subscriptionArn();  
        } else {  
            request = SubscribeRequest.builder()  
                .protocol("sqs")  
                .endpoint(queueArn)  
                .returnSubscriptionArn(true)  
                .topicArn(topicArn)  
                .build();  
  
            SubscribeResponse result = snsClient.subscribe(request);  
            System.out.println("The queue " + queueArn + " has been  
subscribed to the topic " + topicArn + "\n" +  
                "with the subscription ARN " + result.subscriptionArn());  
  
            String attributeName = "FilterPolicy";  
            Gson gson = new Gson();  
            String jsonString = "{\"tone\": []}";  
            JsonObject jsonObject = gson.fromJson(jsonString,  
JsonObject.class);  
            JsonArray toneArray = jsonObject.getAsJsonArray("tone");  
            for (String value : filterList) {  
                toneArray.add(new JsonPrimitive(value));  
            }  
        }  
    }  
}
```



```
        String updatedJsonString = gson.toJson(jsonObject);
        System.out.println(updatedJsonString);
        SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
            .subscriptionArn(result.subscriptionArn())
            .attributeName(attributeName)
            .attributeValue(updatedJsonString)
            .build();

        snsClient.setSubscriptionAttributes(attRequest);
        return result.subscriptionArn();
    }

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

// Attach a policy to the queue.
public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
    try {
        Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
        attrMap.put(QueueAttributeName.POLICY, policy);

        SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributes(attrMap)
            .build();

        sqsClient.setQueueAttributes(attributesRequest);
        System.out.println("The policy has been successfully attached.");

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
public static String getSqsQueueAttrs(SqsClient sqsClient, String queueUrl) {
    // Specify the attributes to retrieve.
    List<QueueAttributeName> atts = new ArrayList<>();
    atts.add(QueueAttributeName.QUEUE_ARN);

    GetQueueAttributesRequest attributesRequest =
    GetQueueAttributesRequest.builder()
        .queueUrl(queueUrl)
        .attributeNames(atts)
        .build();

    GetQueueAttributesResponse response =
    sqsClient.getQueueAttributes(attributesRequest);
    Map<String, String> queueAttrs = response.attributesAsStrings();
    for (Map.Entry<String, String> queueAttr : queueAttrs.entrySet())
        return queueAttr.getValue();

    return "";
}

public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
    try {
        System.out.println("\nCreate Queue");
        if (selectFIFO) {
            Map<QueueAttributeName, String> attrs = new HashMap<>();
            attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
            CreateQueueRequest createQueueRequest =
            CreateQueueRequest.builder()
                .queueName(queueName)
                .attributes(attrs)
                .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient
            .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
        } else {
            CreateQueueRequest createQueueRequest =
            CreateQueueRequest.builder()
                .queueName(queueName)
                .build();
```

```
sqsClient.createQueue(createQueueRequest);
System.out.println("\nGet queue url");
GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
}

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

public static String createSNSTopic(SnsClient snsClient, String topicName) {
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = new HashMap<>();
        if (duplication.compareTo("n") == 0) {
            topicAttributes.put("FifoTopic", "true");
            topicAttributes.put("ContentBasedDeduplication", "false");
        } else {
            topicAttributes.put("FifoTopic", "true");
            topicAttributes.put("ContentBasedDeduplication", "true");
        }
    }
}
```

```
    }

    CreateTopicRequest topicRequest = CreateTopicRequest.builder()
        .name(topicName)
        .attributes(topicAttributes)
        .build();

    CreateTopicResponse response = snsClient.createTopic(topicRequest);
    return response.topicArn();

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Pubblicare](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Questo è il punto di ingresso per questo scenario.

```
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 lo scenario. 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 informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per JavaScript .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Pubblicare](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
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 scenario, 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 json0bject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = json0bject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(json0bject)
```

```

        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 informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Kotlin.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)

- [GetQueueAttributes](#)
- [Pubblicare](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui uno scenario interattivo al prompt dei comandi.

```
class TopicsAndQueuesScenario:
    """Manages the Topics and Queues feature scenario."""

    DASHES = "-" * 80

    def __init__(self, sns_wrapper: SnsWrapper, sqs_wrapper: SqsWrapper) -> None:
        """
        Initialize the Topics and Queues scenario.

        :param sns_wrapper: SnsWrapper instance for SNS operations.
        :param sqs_wrapper: SqsWrapper instance for SQS operations.
        """
        self.sns_wrapper = sns_wrapper
        self.sqs_wrapper = sqs_wrapper

        # Scenario state
        self.use_fifo_topic = False
        self.use_content_based_deduplication = False
        self.topic_name = None
        self.topic_arn = None
```

```
self.queue_count = 2
self.queue_urls = []
self.subscription_arns = []
self.tones = ["cheerful", "funny", "serious", "sincere"]

def run_scenario(self) -> None:
    """Run the Topics and Queues feature scenario."""
    print(self.DASHES)
    print("Welcome to messaging with topics and queues.")
    print(self.DASHES)
    print(f"""
    In this scenario, you will create an SNS topic and subscribe
    {self.queue_count} SQS queues to the topic.
    You can select from several options for configuring the topic and the
    subscriptions for the queues.
    You can then post to the topic and see the results in the queues.
    """)

    try:
        # Setup Phase
        print(self.DASHES)
        self._setup_topic()
        print(self.DASHES)

        self._setup_queues()
        print(self.DASHES)

        # Demonstration Phase
        self._publish_messages()
        print(self.DASHES)

        # Examination Phase
        self._poll_queues_for_messages()
        print(self.DASHES)

        # Cleanup Phase
        self._cleanup_resources()
        print(self.DASHES)

    except Exception as e:
        logger.error(f"Scenario failed: {e}")
        print(f"There was a problem with the scenario: {e}")
        print("\nInitiating cleanup...")
        try:
```

```

        self._cleanup_resources()
    except Exception as cleanup_error:
        logger.error(f"Error during cleanup: {cleanup_error}")

    print("Messaging with topics and queues scenario is complete.")
    print(self.DASHES)

def _setup_topic(self) -> None:
    """Set up the SNS topic to be used with the queues."""
    print("SNS topics can be configured as FIFO (First-In-First-Out).")
    print("FIFO topics deliver messages in order and support deduplication
and message filtering.")
    print()

    self.use_fifo_topic = q.ask("Would you like to work with FIFO topics? (y/
n): ", q.is_yesno)

    if self.use_fifo_topic:
        print(self.DASHES)
        self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)
        print("Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name.")
        print()

        print(self.DASHES)
        print("""
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.
""")

        self.use_content_based_deduplication = q.ask(
            "Use content-based deduplication instead of entering a
deduplication ID? (y/n): ",
            q.is_yesno
        )

```



```

        else:
            self.topic_name = q.ask("Enter a name for your SNS topic: ",
q.non_empty)

            print(self.DASHES)

            # Create the topic
            self.topic_arn = self.sns_wrapper.create_topic(
                self.topic_name,
                self.use_fifo_topic,
                self.use_content_based_deduplication
            )

            print(f"Your new topic with the name {self.topic_name}")
            print(f"  and Amazon Resource Name (ARN) {self.topic_arn}")
            print(f"  has been created.")
            print()

    def _setup_queues(self) -> None:
        """Set up the SQS queues and subscribe them to the topic."""
        print(f"Now you will create {self.queue_count} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.")

        for i in range(self.queue_count):
            queue_name = q.ask(f"Enter a name for SQS queue #{i+1}: ",
q.non_empty)

            if self.use_fifo_topic and i == 0:
                print("Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.")

            # Create the queue
            queue_url = self.sqs_wrapper.create_queue(queue_name,
self.use_fifo_topic)
            self.queue_urls.append(queue_url)

            print(f"Your new queue with the name {queue_name}")
            print(f"  and queue URL {queue_url}")
            print(f"  has been created.")
            print()

            if i == 0:
                print("The queue URL is used to retrieve the queue ARN,")
                print("which is used to create a subscription.")

```

```

        print(self.DASHES)

    # Get queue ARN
    queue_arn = self.sqs_wrapper.get_queue_arn(queue_url)

    if i == 0:
        print("An AWS Identity and Access Management (IAM) policy must be
attached to an SQS queue,")
        print("enabling it to receive messages from an SNS topic.")

    # Set queue policy to allow SNS to send messages
    self.sqs_wrapper.set_queue_policy_for_topic(queue_arn,
self.topic_arn, queue_url)

    # Set up message filtering if using FIFO
    subscription_arn = self._setup_subscription_with_filter(i, queue_arn,
queue_name)
    self.subscription_arns.append(subscription_arn)

    def _setup_subscription_with_filter(self, queue_index: int, queue_arn: str,
queue_name: str) -> str:
        """Set up subscription with optional message filtering."""
        filter_policy = None

        if self.use_fifo_topic:
            print(self.DASHES)
            if queue_index == 0:
                print("Subscriptions to a FIFO topic can have filters.")
                print("If you add a filter to this subscription, then only the
filtered messages")
                print("will be received in the queue.")
                print()
                print("For information about message filtering,")
                print("see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html")
                print()
                print("For this example, you can filter messages by a TONE
attribute.")

                use_filter = q.ask(f"Filter messages for {queue_name}'s subscription
to the topic? (y/n): ", q.is_yesno)

                if use_filter:
                    filter_policy = self._create_filter_policy()

```

```

        subscription_arn = self.sns_wrapper.subscribe_queue_to_topic(
            self.topic_arn, queue_arn, filter_policy
        )

        print(f"The queue {queue_name} has been subscribed to the topic
{self.topic_name}")
        print(f" with the subscription ARN {subscription_arn}")

        return subscription_arn

def _create_filter_policy(self) -> str:
    """Create a message filter policy based on user selections."""
    print(self.DASHES)
    print("You can filter messages by one or more of the following TONE
attributes.")

    filter_selections = []
    selection_number = 0

    while True:
        print("Enter a number to add a TONE filter, or enter 0 to stop adding
filters.")
        for i, tone in enumerate(self.tones, 1):
            print(f" {i}. {tone}")

        selection = q.ask("Your choice: ", q.is_int, q.in_range(0,
len(self.tones)))

        if selection == 0:
            break
        elif selection > 0 and self.tones[selection - 1] not in
filter_selections:
            filter_selections.append(self.tones[selection - 1])
            print(f"Added '{self.tones[selection - 1]}' to filter list.")

    if filter_selections:
        filters = {"tone": filter_selections}
        return json.dumps(filters)
    return None

def _publish_messages(self) -> None:
    """Publish messages to the topic with various options."""
    print("Now we can publish messages.")

```

```
keep_sending = True
while keep_sending:
    print()
    message = q.ask("Enter a message to publish: ", q.non_empty)

    message_group_id = None
    deduplication_id = None
    tone_attribute = None

    if self.use_fifo_topic:
        print("Because you are using a FIFO topic, you must set a message
group ID.")
        print("All messages within the same group will be received in the
order they were published.")
        print()
        message_group_id = q.ask("Enter a message group ID for this
message: ", q.non_empty)

        if not self.use_content_based_deduplication:
            print("Because you are not using content-based
deduplication,")
            print("you must enter a deduplication ID.")
            deduplication_id = q.ask("Enter a deduplication ID for this
message: ", q.non_empty)

        # Ask about tone attribute
        add_attribute = q.ask("Add an attribute to this message? (y/n):
", q.is_yn)
        if add_attribute:
            print("Enter a number for an attribute:")
            for i, tone in enumerate(self.tones, 1):
                print(f" {i}. {tone}")

            selection = q.ask("Your choice: ", q.is_int, q.in_range(1,
len(self.tones)))
            if 1 <= selection <= len(self.tones):
                tone_attribute = self.tones[selection - 1]

    # Publish the message
    message_id = self.sns_wrapper.publish_message(
        self.topic_arn,
        message,
        tone_attribute,
```

```

        deduplication_id,
        message_group_id
    )

    print(f"Message published with ID: {message_id}")

    keep_sending = q.ask("Send another message? (y/n): ", q.is_yesno)

def _poll_queues_for_messages(self) -> None:
    """Poll all queues for messages and display results."""
    for i, queue_url in enumerate(self.queue_urls):
        print(f"Polling queue #{i+1} at {queue_url} for messages...")

        q.ask("Press Enter to continue...")

        messages = self._poll_queue_for_messages(queue_url)

        if messages:
            print(f"{len(messages)} message(s) were received by queue #{i+1}")

            for j, message in enumerate(messages, 1):
                print(f"  Message {j}:")
                # Parse the SNS message body to get the actual message
                try:
                    sns_message = json.loads(message['Body'])
                    actual_message = sns_message.get('Message',
message['Body'])

                    print(f"    {actual_message}")
                except (json.JSONDecodeError, KeyError):
                    print(f"    {message['Body']}")

            # Delete the messages
            self.sqs_wrapper.delete_messages(queue_url, messages)
            print(f"Messages deleted from queue #{i+1}")
        else:
            print(f"No messages received by queue #{i+1}")

        print(self.DASHES)

def _poll_queue_for_messages(self, queue_url: str) -> List[Dict[str, Any]]:
    """Poll a single queue for messages."""
    all_messages = []
    max_polls = 3 # Limit polling to avoid infinite loops

```

```

    for poll_count in range(max_polls):
        messages = self.sqs_wrapper.receive_messages(queue_url, 10)

        if messages:
            all_messages.extend(messages)
            print(f" Received {len(messages)} messages in poll {poll_count +
1}")

            # Small delay between polls
            time.sleep(1)
        else:
            print(f" No messages in poll {poll_count + 1}")
            break

    return all_messages

def _cleanup_resources(self) -> None:
    """Clean up all resources created during the scenario."""
    print("Cleaning up resources...")

    # Delete queues
    for i, queue_url in enumerate(self.queue_urls):
        if queue_url:
            delete_queue = q.ask(f"Delete queue #{i+1} with URL {queue_url}?
(y/n): ", q.is_yesno)
            if delete_queue:
                try:
                    self.sqs_wrapper.delete_queue(queue_url)
                    print(f"Deleted queue #{i+1}")
                except Exception as e:
                    print(f"Error deleting queue #{i+1}: {e}")

    # Unsubscribe from topic
    for i, subscription_arn in enumerate(self.subscription_arns):
        if subscription_arn:
            try:
                self.sns_wrapper.unsubscribe(subscription_arn)
                print(f"Unsubscribed subscription #{i+1}")
            except Exception as e:
                print(f"Error unsubscribing #{i+1}: {e}")

    # Delete topic
    if self.topic_arn:
        delete_topic = q.ask(f"Delete topic {self.topic_name}? (y/n): ",
q.is_yesno)

```

```

        if delete_topic:
            try:
                self.sns_wrapper.delete_topic(self.topic_arn)
                print(f"Deleted topic {self.topic_name}")
            except Exception as e:
                print(f"Error deleting topic: {e}")

    print("Resource cleanup complete.")

```

Crea classi che racchiudono le operazioni di Amazon SNS e Amazon SQS da utilizzare nello scenario.

```

class SnsWrapper:
    """Wrapper class for managing Amazon SNS operations."""

    def __init__(self, sns_client: Any) -> None:
        """
        Initialize the SnsWrapper.

        :param sns_client: A Boto3 Amazon SNS client.
        """
        self.sns_client = sns_client

    @classmethod
    def from_client(cls) -> 'SnsWrapper':
        """
        Create an SnsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sns_client = boto3.client('sns')
        return cls(sns_client)

    def create_topic(
        self,
        topic_name: str,
        is_fifo: bool = False,
        content_based_deduplication: bool = False
    ) -> str:
        """

```

```

    Create an SNS topic.

    :param topic_name: The name of the topic to create.
    :param is_fifo: Whether to create a FIFO topic.
    :param content_based_deduplication: Whether to use content-based
deduplication for FIFO topics.
    :return: The ARN of the created topic.
    :raises ClientError: If the topic creation fails.
    """
    try:
        # Add .fifo suffix for FIFO topics
        if is_fifo and not topic_name.endswith('.fifo'):
            topic_name += '.fifo'

        attributes = {}
        if is_fifo:
            attributes['FifoTopic'] = 'true'
            if content_based_deduplication:
                attributes['ContentBasedDeduplication'] = 'true'

        response = self.sns_client.create_topic(
            Name=topic_name,
            Attributes=attributes
        )

        topic_arn = response['TopicArn']
        logger.info(f"Created topic: {topic_name} with ARN: {topic_arn}")
        return topic_arn

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error creating topic {topic_name}: {error_code} -
{e}")
        raise

    def subscribe_queue_to_topic(
        self,
        topic_arn: str,
        queue_arn: str,
        filter_policy: Optional[str] = None
    ) -> str:
        """
        Subscribe an SQS queue to an SNS topic.

```



```

:param topic_arn: The ARN of the SNS topic.
:param queue_arn: The ARN of the SQS queue.
:param filter_policy: Optional JSON filter policy for message filtering.
:return: The ARN of the subscription.
:raises ClientError: If the subscription fails.
"""
try:
    attributes = {}
    if filter_policy:
        attributes['FilterPolicy'] = filter_policy

    response = self.sns_client.subscribe(
        TopicArn=topic_arn,
        Protocol='sqs',
        Endpoint=queue_arn,
        Attributes=attributes
    )

    subscription_arn = response['SubscriptionArn']
    logger.info(f"Subscribed queue {queue_arn} to topic {topic_arn}")
    return subscription_arn

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error subscribing queue to topic: {error_code} - {e}")
    raise

def publish_message(
    self,
    topic_arn: str,
    message: str,
    tone_attribute: Optional[str] = None,
    deduplication_id: Optional[str] = None,
    message_group_id: Optional[str] = None
) -> str:
    """
    Publish a message to an SNS topic.

    :param topic_arn: The ARN of the SNS topic.
    :param message: The message content to publish.
    :param tone_attribute: Optional tone attribute for message filtering.
    :param deduplication_id: Optional deduplication ID for FIFO topics.

```

```

:param message_group_id: Optional message group ID for FIFO topics.
:return: The message ID of the published message.
:raises ClientError: If the message publication fails.
"""
try:
    publish_args = {
        'TopicArn': topic_arn,
        'Message': message
    }

    # Add message attributes if tone is specified
    if tone_attribute:
        publish_args['MessageAttributes'] = {
            'tone': {
                'DataType': 'String',
                'StringValue': tone_attribute
            }
        }

    # Add FIFO-specific parameters
    if message_group_id:
        publish_args['MessageGroupId'] = message_group_id

    if deduplication_id:
        publish_args['MessageDeduplicationId'] = deduplication_id

    response = self.sns_client.publish(**publish_args)

    message_id = response['MessageId']
    logger.info(f"Published message to topic {topic_arn} with ID:
{message_id}")
    return message_id

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error publishing message to topic: {error_code} -
{e}")
    raise

def unsubscribe(self, subscription_arn: str) -> bool:
    """
    Unsubscribe from an SNS topic.

```

```
    :param subscription_arn: The ARN of the subscription to remove.
    :return: True if successful.
    :raises ClientError: If the unsubscribe operation fails.
    """
    try:
        self.sns_client.unsubscribe(SubscriptionArn=subscription_arn)

        logger.info(f"Unsubscribed: {subscription_arn}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'NotFound':
            logger.warning(f"Subscription not found: {subscription_arn}")
            return True # Already unsubscribed
        else:
            logger.error(f"Error unsubscribing: {error_code} - {e}")
            raise

def delete_topic(self, topic_arn: str) -> bool:
    """
    Delete an SNS topic.

    :param topic_arn: The ARN of the topic to delete.
    :return: True if successful.
    :raises ClientError: If the topic deletion fails.
    """
    try:
        self.sns_client.delete_topic(TopicArn=topic_arn)

        logger.info(f"Deleted topic: {topic_arn}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'NotFound':
            logger.warning(f"Topic not found: {topic_arn}")
            return True # Already deleted
        else:
            logger.error(f"Error deleting topic: {error_code} - {e}")
            raise
```

```

def list_topics(self) -> list:
    """
    List all SNS topics in the account using pagination.

    :return: List of topic ARNs.
    :raises ClientError: If listing topics fails.
    """
    try:
        topics = []
        paginator = self.sns_client.get_paginator('list_topics')

        for page in paginator.paginate():
            topics.extend([topic['TopicArn'] for topic in page.get('Topics',
[])])

        logger.info(f"Found {len(topics)} topics")
        return topics

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        if error_code == 'AuthorizationError':
            logger.error("Authorization error listing topics - check IAM
permissions")
        else:
            logger.error(f"Error listing topics: {error_code} - {e}")
        raise

class SqsWrapper:
    """Wrapper class for managing Amazon SQS operations."""

    def __init__(self, sqs_client: Any) -> None:
        """
        Initialize the SqsWrapper.

        :param sqs_client: A Boto3 Amazon SQS client.
        """
        self.sqs_client = sqs_client

    @classmethod
    def from_client(cls) -> 'SqsWrapper':
        """

```

```

        Create an SqsWrapper instance using a default boto3 client.

        :return: An instance of this class.
        """
        sqs_client = boto3.client('sqs')
        return cls(sqs_client)

def create_queue(self, queue_name: str, is_fifo: bool = False) -> str:
    """
    Create an SQS queue.

    :param queue_name: The name of the queue to create.
    :param is_fifo: Whether to create a FIFO queue.
    :return: The URL of the created queue.
    :raises ClientError: If the queue creation fails.
    """
    try:
        # Add .fifo suffix for FIFO queues
        if is_fifo and not queue_name.endswith('.fifo'):
            queue_name += '.fifo'

        attributes = {}
        if is_fifo:
            attributes['FifoQueue'] = 'true'

        response = self.sqs_client.create_queue(
            QueueName=queue_name,
            Attributes=attributes
        )

        queue_url = response['QueueUrl']
        logger.info(f"Created queue: {queue_name} with URL: {queue_url}")
        return queue_url

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error creating queue {queue_name}: {error_code} - {e}")
        raise

def get_queue_arn(self, queue_url: str) -> str:
    """

```

```

    Get the ARN of an SQS queue.

    :param queue_url: The URL of the queue.
    :return: The ARN of the queue.
    :raises ClientError: If getting queue attributes fails.
    """
    try:
        response = self.sqs_client.get_queue_attributes(
            QueueUrl=queue_url,
            AttributeNames=['QueueArn']
        )

        queue_arn = response['Attributes']['QueueArn']
        logger.info(f"Queue ARN for {queue_url}: {queue_arn}")
        return queue_arn

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error getting queue ARN: {error_code} - {e}")
        raise

def set_queue_policy_for_topic(self, queue_arn: str, topic_arn: str,
queue_url: str) -> bool:
    """
    Set the queue policy to allow SNS to send messages to the queue.

    :param queue_arn: The ARN of the SQS queue.
    :param topic_arn: The ARN of the SNS topic.
    :param queue_url: The URL of the SQS queue.
    :return: True if successful.
    :raises ClientError: If setting the queue policy fails.
    """
    try:
        # Create policy that allows SNS to send messages to the queue
        policy = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "sns.amazonaws.com"
                    },
                    "Action": "sqs:SendMessage",

```

```

        "Resource": queue_arn,
        "Condition": {
            "ArnEquals": {
                "aws:SourceArn": topic_arn
            }
        }
    ]
}

self.sqs_client.set_queue_attributes(
    QueueUrl=queue_url,
    Attributes={
        'Policy': json.dumps(policy)
    }
)

logger.info(f"Set queue policy for {queue_url} to allow messages from
{topic_arn}")
return True

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error setting queue policy: {error_code} - {e}")
    raise

def receive_messages(self, queue_url: str, max_messages: int = 10) ->
List[Dict[str, Any]]:
    """
    Receive messages from an SQS queue.

    :param queue_url: The URL of the queue to receive messages from.
    :param max_messages: Maximum number of messages to receive (1-10).
    :return: List of received messages.
    :raises ClientError: If receiving messages fails.
    """
    try:
        # Ensure max_messages is within valid range
        max_messages = max(1, min(10, max_messages))

        response = self.sqs_client.receive_message(
            QueueUrl=queue_url,
            MaxNumberOfMessages=max_messages,

```

```

        WaitTimeSeconds=2, # Short polling
        MessageAttributeNames=['All']
    )

    messages = response.get('Messages', [])
    logger.info(f"Received {len(messages)} messages from {queue_url}")
    return messages

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error receiving messages: {error_code} - {e}")
    raise

def delete_messages(self, queue_url: str, messages: List[Dict[str, Any]]) ->
bool:
    """
    Delete messages from an SQS queue in batches.

    :param queue_url: The URL of the queue.
    :param messages: List of messages to delete.
    :return: True if successful.
    :raises ClientError: If deleting messages fails.
    """
    try:
        if not messages:
            return True

        # Build delete entries for batch delete
        delete_entries = []
        for i, message in enumerate(messages):
            delete_entries.append({
                'Id': str(i),
                'ReceiptHandle': message['ReceiptHandle']
            })

        # Delete messages in batches of 10 (SQS limit)
        batch_size = 10
        for i in range(0, len(delete_entries), batch_size):
            batch = delete_entries[i:i + batch_size]

            response = self.sqs_client.delete_message_batch(
                QueueUrl=queue_url,
                Entries=batch
            )

```



```

        )

        # Check for failures
        if 'Failed' in response and response['Failed']:
            for failed in response['Failed']:
                logger.warning(f"Failed to delete message: {failed}")

        logger.info(f"Deleted {len(messages)} messages from {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')
        logger.error(f"Error deleting messages: {error_code} - {e}")
        raise

def delete_queue(self, queue_url: str) -> bool:
    """
    Delete an SQS queue.

    :param queue_url: The URL of the queue to delete.
    :return: True if successful.
    :raises ClientError: If the queue deletion fails.
    """
    try:
        self.sqs_client.delete_queue(QueueUrl=queue_url)

        logger.info(f"Deleted queue: {queue_url}")
        return True

    except ClientError as e:
        error_code = e.response.get('Error', {}).get('Code', 'Unknown')

        if error_code == 'AWS.SimpleQueueService.NonExistentQueue':
            logger.warning(f"Queue not found: {queue_url}")
            return True # Already deleted
        else:
            logger.error(f"Error deleting queue: {error_code} - {e}")
            raise

def list_queues(self, queue_name_prefix: Optional[str] = None) -> List[str]:
    """
    List all SQS queues in the account using pagination.

```

```

:param queue_name_prefix: Optional prefix to filter queue names.
:return: List of queue URLs.
:raises ClientError: If listing queues fails.
"""
try:
    queue_urls = []
    paginator = self.sqs_client.get_paginator('list_queues')

    page_params = {}
    if queue_name_prefix:
        page_params['QueueNamePrefix'] = queue_name_prefix

    for page in paginator.paginate(**page_params):
        queue_urls.extend(page.get('QueueUrls', []))

    logger.info(f"Found {len(queue_urls)} queues")
    return queue_urls

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    if error_code == 'AccessDenied':
        logger.error("Access denied listing queues - check IAM
permissions")
    else:
        logger.error(f"Error listing queues: {error_code} - {e}")
    raise

def send_message(self, queue_url: str, message_body: str, **kwargs) -> str:
    """
    Send a message to an SQS queue.

    :param queue_url: The URL of the queue.
    :param message_body: The message content.
    :param kwargs: Additional message parameters (DelaySeconds,
MessageAttributes, etc.).
    :return: The message ID.
    :raises ClientError: If sending the message fails.
    """
    try:
        send_params = {
            'QueueUrl': queue_url,
            'MessageBody': message_body,
            **kwargs

```

```
}

response = self.sqs_client.send_message(**send_params)

message_id = response['MessageId']
logger.info(f"Sent message to {queue_url} with ID: {message_id}")
return message_id

except ClientError as e:
    error_code = e.response.get('Error', {}).get('Code', 'Unknown')
    logger.error(f"Error sending message: {error_code} - {e}")
    raise
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Pubblicare](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSSNS
import AWSSQS
import Foundation

struct ExampleCommand: ParsableCommand {
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "queue-scenario",
        abstract: ""
        This example interactively demonstrates how to use Amazon Simple
        Notification Service (Amazon SNS) and Amazon Simple Queue Service
        (Amazon SQS) together to publish and receive messages using queues.
        "",
        discussion: ""
        Supports filtering using a "tone" attribute.
        ""
    )

    /// Prompt for an input string. Only non-empty strings are allowed.
    ///
    /// - Parameter prompt: The prompt to display.
    ///
    /// - Returns: The string input by the user.
    func stringRequest(prompt: String) -> String {
        var str: String?

        while str == nil {
            print(prompt, terminator: "")
```

```
        str = readLine()

        if str != nil && str?.count == 0 {
            str = nil
        }
    }

    return str!
}

/// Ask a yes/no question.
///
/// - Parameter prompt: A prompt string to print.
///
/// - Returns: `true` if the user answered "Y", otherwise `false`.
func yesNoRequest(prompt: String) -> Bool {
    while true {
        let answer = stringRequest(prompt: prompt).lowercased()
        if answer == "y" || answer == "n" {
            return answer == "y"
        }
    }
}

/// Display a menu of options then request a selection.
///
/// - Parameters:
///   - prompt: A prompt string to display before the menu.
///   - options: An array of strings giving the menu options.
///
/// - Returns: The index number of the selected option or 0 if no item was
///   selected.
func menuRequest(prompt: String, options: [String]) -> Int {
    let numOptions = options.count

    if numOptions == 0 {
        return 0
    }

    print(prompt)

    for (index, value) in options.enumerated() {
        print("\(index) \(value)")
    }
}
```

```
repeat {
    print("Enter your selection (0 - \$(numOptions-1)): ", terminator: "")
    if let answer = readLine() {
        guard let answer = Int(answer) else {
            print("Please enter the number matching your selection.")
            continue
        }

        if answer >= 0 && answer < numOptions {
            return answer
        } else {
            print("Please enter the number matching your selection.")
        }
    }
} while true
}

/// Ask the user too press RETURN. Accepts any input but ignores it.
///
/// - Parameter prompt: The text prompt to display.
func returnRequest(prompt: String) {
    print(prompt, terminator: "")
    _ = readLine()
}

var attrValues = [
    "<none>",
    "cheerful",
    "funny",
    "serious",
    "sincere"
]

/// Ask the user to choose one of the attribute values to use as a filter.
///
/// - Parameters:
///   - message: A message to display before the menu of values.
///   - attrValues: An array of strings giving the values to choose from.
///
/// - Returns: The string corresponding to the selected option.
func askForFilter(message: String, attrValues: [String]) -> String? {
    print(message)
    for (index, value) in attrValues.enumerated() {
```

```

        print("  [\\(index)] \\(value)")
    }

    var answer: Int?
    repeat {
        answer = Int(stringRequest(prompt: "Select an value for the 'tone'
attribute or 0 to end: "))
    } while answer == nil || answer! < 0 || answer! > attrValues.count + 1

    if answer == 0 {
        return nil
    }
    return attrValues[answer!]
}

/// Prompts the user for filter terms and constructs the attribute
/// record that specifies them.
///
/// - Returns: A mapping of "FilterPolicy" to a JSON string representing
///   the user-defined filter.
func buildFilterAttributes() -> [String:String] {
    var attr: [String:String] = [:]
    var filterString = ""

    var first = true

    while let ans = askForFilter(message: "Choose a value to apply to the
'tone' attribute.",
                                attrValues: attrValues) {
        if !first {
            filterString += ","
        }
        first = false

        filterString += "\\\"\\(ans)\\\""
    }

    let filterJSON = "{ \"tone\": [\\(filterString)]}"
    attr["FilterPolicy"] = filterJSON

    return attr
}
/// Create a queue, returning its URL string.
///

```

```

    /// - Parameters:
    ///   - prompt: A prompt to ask for the queue name.
    ///   - isFIFO: Whether or not to create a FIFO queue.
    ///
    /// - Returns: The URL of the queue.
    func createQueue(prompt: String, sqsClient: SQSClient, isFIFO: Bool) async
    throws -> String? {
        repeat {
            var queueName = stringRequest(prompt: prompt)
            var attributes: [String: String] = [:]

            if isFIFO {
                queueName += ".fifo"
                attributes["FifoQueue"] = "true"
            }

            do {
                let output = try await sqsClient.createQueue(
                    input: CreateQueueInput(
                        attributes: attributes,
                        queueName: queueName
                    )
                )
                guard let url = output.queueUrl else {
                    return nil
                }

                return url
            } catch _ as QueueDeletedRecently {
                print("You need to use a different queue name. A queue by that
name was recently deleted.")
                continue
            }
        } while true
    }

    /// Return the ARN of a queue given its URL.
    ///
    /// - Parameter queueUrl: The URL of the queue for which to return the
    ///   ARN.
    ///
    /// - Returns: The ARN of the specified queue.
    func getQueueARN(sqsClient: SQSClient, queueUrl: String) async throws ->
    String? {

```



```

    let output = try await sqsClient.getQueueAttributes(
        input: GetQueueAttributesInput(
            attributeNames: [.queueArn],
            queueUrl: queueUrl
        )
    )

    guard let attributes = output.attributes else {
        return nil
    }

    return attributes["QueueArn"]
}

/// Applies the needed policy to the specified queue.
///
/// - Parameters:
///   - sqsClient: The Amazon SQS client to use.
///   - queueUrl: The queue to apply the policy to.
///   - queueArn: The ARN of the queue to apply the policy to.
///   - topicArn: The topic that should have access via the policy.
///
/// - Throws: Errors from the SQS `SetQueueAttributes` action.
func setQueuePolicy(sqsClient: SQSClient, queueUrl: String,
                    queueArn: String, topicArn: String) async throws {
    _ = try await sqsClient.setQueueAttributes(
        input: SetQueueAttributesInput(
            attributes: [
                "Policy":
                """
                {
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {
                                "Service": "sns.amazonaws.com"
                            },
                            "Action": "sqs:SendMessage",
                            "Resource": "\(queueArn)",
                            "Condition": {
                                "ArnEquals": {
                                    "aws:SourceArn": "\(topicArn)"
                                }
                            }
                        }
                    ]
                }
                """
            ]
        )
    )
}

```

```

        }
    ]
}
"""

    ],
    queueUrl: queueUrl
)
)
}

/// Receive the available messages on a queue, outputting them to the
/// screen. Returns a dictionary you pass to DeleteMessageBatch to delete
/// all the received messages.
///
/// - Parameters:
///   - sqsClient: The Amazon SQS client to use.
///   - queueUrl: The SQS queue on which to receive messages.
///
/// - Throws: Errors from `SQSClient.receiveMessage()`
///
/// - Returns: An array of SQSClientTypes.DeleteMessageBatchRequestEntry
///   items, each describing one received message in the format needed to
///   delete it.
func receiveAndListMessages(sqsClient: SQSClient, queueUrl: String) async
throws
    ->
[SQSClientTypes.DeleteMessageBatchRequestEntry] {
    let output = try await sqsClient.receiveMessage(
        input: ReceiveMessageInput(
            maxNumberOfMessages: 10,
            queueUrl: queueUrl
        )
    )

    guard let messages = output.messages else {
        print("No messages received.")
        return []
    }

    var deleteList: [SQSClientTypes.DeleteMessageBatchRequestEntry] = []

    // Print out all the messages that were received, including their
    // attributes, if any.

```

```

    for message in messages {
        print("Message ID:      \(message.messageId ?? "<unknown>")")
        print("Receipt handle: \(message.receiptHandle ?? "<unknown>")")
        print("Message JSON:    \(message.body ?? "<body missing>")")

        if message.receiptHandle != nil {
            deleteList.append(
                SQSClientTypes.DeleteMessageBatchRequestEntry(
                    id: message.messageId,
                    receiptHandle: message.receiptHandle
                )
            )
        }
    }

    return deleteList
}

/// Delete all the messages in the specified list.
///
/// - Parameters:
///   - sqsClient: The Amazon SQS client to use.
///   - queueUrl: The SQS queue to delete messages from.
///   - deleteList: A list of `DeleteMessageBatchRequestEntry` objects
///     describing the messages to delete.
///
/// - Throws: Errors from `SQSClient.deleteMessageBatch()`.
func deleteMessageList(sqsClient: SQSClient, queueUrl: String,
                      deleteList:
[SQSClientTypes.DeleteMessageBatchRequestEntry]) async throws {
    let output = try await sqsClient.deleteMessageBatch(
        input: DeleteMessageBatchInput(entries: deleteList, queueUrl:
queueUrl)
    )

    if let failed = output.failed {
        print("\(failed.count) errors occurred deleting messages from the
queue.")
        for message in failed {
            print("---> Failed to delete message \(message.id ?? "<unknown
ID>") with error: \(message.code ?? "<unknown>") (\(message.message ?? "..."))")
        }
    }
}

```

```

    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let rowOfStars = String(repeating: "*", count: 75)

        print("""
            \(\rowOfStars)
            Welcome to the cross-service messaging with topics and queues
example.

            In this workflow, you'll create an SNS topic, then create two SQS
            queues which will be subscribed to that topic.

            You can specify several options for configuring the topic, as well
as
            the queue subscriptions. You can then post messages to the topic
and
            receive the results on the queues.
            \(\rowOfStars)\n
            """)
        )

        // 0. Create SNS and SQS clients.

        let snsConfig = try await SNSClient.SNSClientConfiguration(region:
region)
        let snsClient = SNSClient(config: snsConfig)

        let sqsConfig = try await SQSClient.SQSClientConfiguration(region:
region)
        let sqsClient = SQSClient(config: sqsConfig)

        // 1. Ask the user whether to create a FIFO topic. If so, ask whether
        //     to use content-based deduplication instead of requiring a
        //     deduplication ID.

        let isFIFO = yesNoRequest(prompt: "Do you want to create a FIFO topic (Y/
N)? ")
        var isContentBasedDeduplication = false

        if isFIFO {
            print("""
                \(\rowOfStars)
                Because you've chosen to create a FIFO topic, deduplication is

```

```

supported.

Deduplication IDs are either set in the message or are
automatically
generated from the content using a hash function.

If a message is successfully published to an SNS FIFO topic,
any
message published and found to have the same deduplication ID
(within a 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.
"""
)

isContentBasedDeduplication = yesNoRequest(
    prompt: "Use content-based deduplication instead of entering a
deduplication ID (Y/N)? ")
print(rowOfStars)
}

var topicName = stringRequest(prompt: "Enter the name of the topic to
create: ")

// 2. Create the topic. Append ".fifo" to the name if FIFO was
// requested, and set the "FifoTopic" attribute to "true" if so as
// well. Set the "ContentBasedDeduplication" attribute to "true" if
// content-based deduplication was requested.

if isFIFO {
    topicName += ".fifo"
}

print("Topic name: \(topicName)")

var attributes = [
    "FifoTopic": (isFIFO ? "true" : "false")
]

// If it's a FIFO topic with content-based deduplication, set the
// "ContentBasedDeduplication" attribute.

```

```

    if isContentBasedDeduplication {
        attributes["ContentBasedDeduplication"] = "true"
    }

    // Create the topic and retrieve the ARN.

    let output = try await snsClient.createTopic(
        input: CreateTopicInput(
            attributes: attributes,
            name: topicName
        )
    )

    guard let topicArn = output.topicArn else {
        print("No topic ARN returned!")
        return
    }

    print("""
        Topic '\(topicName)' has been created with the
        topic ARN '\(topicArn)'.
    """)

    )

    print(rowOfStars)

    // 3. Create an SQS queue. Append ".fifo" to the name if one of the
    //     FIFO topic configurations was chosen, and set "FifoQueue" to
    //     "true" if the topic is FIFO.

    print("""
        Next, you will create two SQS queues that will be subscribed
        to the topic you just created.\n
    """)

    )

    let q1Url = try await createQueue(prompt: "Enter the name of the first
queue: ",
                                     sqsClient: sqsClient, isFIFO: isFIFO)

    guard let q1Url else {
        print("Unable to create queue 1!")
        return
    }

```

```
// 4. Get the SQS queue's ARN attribute using `GetQueueAttributes`.

let q1Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q1Url)

guard let q1Arn else {
    print("Unable to get ARN of queue 1!")
    return
}
print("Got queue 1 ARN: \(q1Arn)")

// 5. Attach an AWS IAM policy to the queue using
//     `SetQueueAttributes`.

try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q1Url,
                        queueArn: q1Arn, topicArn: topicArn)

// 6. Subscribe the SQS queue to the SNS topic. Set the topic ARN in
//     the request. Set the protocol to "sqs". Set the queue ARN to the
//     ARN just received in step 5. For FIFO topics, give the option to
//     apply a filter. A filter allows only matching messages to enter
//     the queue.

var q1Attributes: [String:String]? = nil

if isFIFO {
    print(
        """
        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.
        """
    )

    let subPrompt = """
        Would you like to filter messages for the first queue's
subscription to the
        topic \(topicName) (Y/N)?
    """
}
```

```

        """"
        if (yesNoRequest(prompt: subPrompt)) {
            q1Attributes = buildFilterAttributes()
        }
    }

    let sub1Output = try await snsClient.subscribe(
        input: SubscribeInput(
            attributes: q1Attributes,
            endpoint: q1Arn,
            protocol: "sqs",
            topicArn: topicArn
        )
    )

    guard let q1SubscriptionArn = sub1Output.subscriptionArn else {
        print("Invalid subscription ARN returned for queue 1!")
        return
    }

    // 7. Repeat steps 3-6 for the second queue.

    let q2Url = try await createQueue(prompt: "Enter the name of the second
queue: ",
                                     sqsClient: sqsClient, isFIFO: isFIFO)

    guard let q2Url else {
        print("Unable to create queue 2!")
        return
    }

    let q2Arn = try await getQueueARN(sqsClient: sqsClient, queueUrl: q2Url)

    guard let q2Arn else {
        print("Unable to get ARN of queue 2!")
        return
    }
    print("Got queue 2 ARN: \(q2Arn)")

    try await setQueuePolicy(sqsClient: sqsClient, queueUrl: q2Url,
                             queueArn: q2Arn, topicArn: topicArn)

    var q2Attributes: [String:String]? = nil

```



```

        if isFIFO {
            let subPrompt = ""
            Would you like to filter messages for the second queue's
subscription to the
            topic \((topicName) (Y/N)?
            ""
            if (yesNoRequest(prompt: subPrompt)) {
                q2Attributes = buildFilterAttributes()
            }
        }

        let sub2Output = try await snsClient.subscribe(
            input: SubscribeInput(
                attributes: q2Attributes,
                endpoint: q2Arn,
                protocol: "sqs",
                topicArn: topicArn
            )
        )

        guard let q2SubscriptionArn = sub2Output.subscriptionArn else {
            print("Invalid subscription ARN returned for queue 1!")
            return
        }

        // 8. Let the user publish messages to the topic, asking for a message
        //     body for each message. Handle the types of topic correctly (SEE
        //     MVP INFORMATION AND FIX THESE COMMENTS!!!

        print("\n\((rowOfStars)\n")

        var first = true

        repeat {
            var publishInput = PublishInput(
                topicArn: topicArn
            )

            publishInput.message = stringRequest(prompt: "Enter message text to
publish: ")

            // If using a FIFO topic, a message group ID must be set on the
            // message.

```

```

        if isFIFO {
            if first {
                print("""
                    Because you're using a FIFO topic, you must set a message
                    group ID. All messages within the same group will be
                    received in the same order in which they were published.

\n
                    """)
            }
            publishInput.messageGroupId = stringRequest(prompt: "Enter a
message group ID for this message: ")

            if !isContentBasedDeduplication {
                if first {
                    print("""
                        Because you're not using content-based
deduplication, you
                        must enter a deduplication ID. If other messages
with the
                        same deduplication ID are published within the same
deduplication interval, they will not be delivered.
                    """)
                }
                publishInput.messageDeduplicationId = stringRequest(prompt:
"Enter a deduplication ID for this message: ")
            }
        }

        // Allow the user to add a value for the "tone" attribute if they
        // wish to do so.

        var messageAttributes: [String: SNSClientTypes.MessageAttributeValue]
        = [:]

        let attrValSelection = menuRequest(prompt: "Choose a tone to apply to
this message.", options: attrValues)

        if attrValSelection != 0 {
            let val = SNSClientTypes.MessageAttributeValue(dataType:
"String", stringValue: attrValues[attrValSelection])
            messageAttributes["tone"] = val
        }

```

```
        publishInput.messageAttributes = messageAttributes

        // Publish the message and display its ID.

        let publishOutput = try await snsClient.publish(input: publishInput)

        guard let messageId = publishOutput.messageId else {
            print("Unable to get the published message's ID!")
            return
        }

        print("Message published with ID \(messageId).")
        first = false

        // 9. Repeat step 8 until the user says they don't want to post
        //     another.

    } while (yesNoRequest(prompt: "Post another message (Y/N)? "))

    // 10. Display a list of the messages in each queue by using
    //     `ReceiveMessage`. Show at least the body and the attributes.

    print(rowOfStars)
    print("Contents of queue 1:")
    let q1DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q1Url)
    print("\n\nContents of queue 2:")
    let q2DeleteList = try await receiveAndListMessages(sqsClient: sqsClient,
queueUrl: q2Url)
    print(rowOfStars)

    returnRequest(prompt: "\nPress return to clean up: ")

    // 11. Delete the received messages using `DeleteMessageBatch`.

    print("Deleting the messages from queue 1...")
    try await deleteMessageList(sqsClient: sqsClient, queueUrl: q1Url,
deleteList: q1DeleteList)
    print("\nDeleting the messages from queue 2...")
    try await deleteMessageList(sqsClient: sqsClient, queueUrl: q2Url,
deleteList: q2DeleteList)

    // 12. Unsubscribe and delete both queues.
```

```

        print("\nUnsubscribing from queue 1...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q1SubscriptionArn)
        )

        print("Unsubscribing from queue 2...")
        _ = try await snsClient.unsubscribe(
            input: UnsubscribeInput(subscriptionArn: q2SubscriptionArn)
        )

        print("Deleting queue 1...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q1Url)
        )

        print("Deleting queue 2...")
        _ = try await sqsClient.deleteQueue(
            input: DeleteQueueInput(queueUrl: q2Url)
        )

        // 13. Delete the topic.

        print("Deleting the SNS topic...")
        _ = try await snsClient.deleteTopic(
            input: DeleteTopicInput(topicArn: topicArn)
        )
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API SDK AWS per Swift.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Pubblicare](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo di Gateway API per invocare una funzione Lambda

I seguenti esempi di codice mostrano come creare una AWS Lambda funzione richiamata da Amazon API Gateway.

Java

SDK per Java 2.x

Mostra come creare una AWS Lambda funzione utilizzando l'API runtime Lambda Java. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. In questo esempio viene illustrato come creare una funzione Lambda invocata da Gateway Amazon API che analizza una tabella Amazon DynamoDB per le ricorrenze di lavoro e utilizza Amazon Simple Notification Service (Amazon SNS) per inviare un messaggio di testo ai dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

Mostra come creare una AWS Lambda funzione utilizzando l'API di JavaScript runtime Lambda. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. In questo esempio viene illustrato come creare una funzione Lambda invocata da Gateway Amazon API che analizza una tabella Amazon DynamoDB per le ricorrenze di lavoro e utilizza Amazon Simple Notification Service (Amazon SNS) per inviare un messaggio di testo ai dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Questo esempio è anche disponibile nella [Guida per lo sviluppatore di AWS SDK per JavaScript v3](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon SNS

Python

SDK per Python (Boto3)

L'esempio mostra come creare e utilizzare una REST API di Gateway Amazon API destinata a una funzione AWS Lambda . Il gestore Lambda dimostra come definire route in base ai

metodi HTTP, come ottenere dati dalla stringa, dall'intestazione e dal corpo della query e come restituire una risposta JSON.

- Distribuire una funzione Lambda.
- Creare una REST API di Gateway API.
- Creare una risorsa REST destinata alla funzione Lambda.
- Concedere l'autorizzazione affinché Gateway API invochi la funzione Lambda.
- Utilizza il pacchetto Richieste per inviare le richieste alla REST API.
- Eliminare tutte le risorse create durante la demo.

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

- Gateway API
- DynamoDB
- Lambda
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzo degli eventi pianificati per invocare una funzione Lambda

I seguenti esempi di codice mostrano come creare una AWS Lambda funzione richiamata da un evento EventBridge pianificato di Amazon.

Java

SDK per Java 2.x

Mostra come creare un evento EventBridge pianificato da Amazon che richiami una AWS Lambda funzione. Configura EventBridge per utilizzare un'espressione cron per pianificare quando viene richiamata la funzione Lambda. In questo esempio, viene creata una funzione Lambda utilizzando l'API di runtime Lambda Java. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. Questo esempio dimostra come creare un'app

che invia un messaggio di testo via mobile ai tuoi dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

Mostra come creare un evento EventBridge pianificato da Amazon che richiami una AWS Lambda funzione. Configura EventBridge per utilizzare un'espressione cron per pianificare quando viene richiamata la funzione Lambda. In questo esempio, crei una funzione Lambda utilizzando l'API JavaScript Lambda runtime. Questo esempio richiama diversi AWS servizi per eseguire un caso d'uso specifico. Questo esempio dimostra come creare un'app che invia un messaggio di testo via mobile ai tuoi dipendenti che si congratula con loro alla data dell'anniversario di un anno.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Questo esempio è anche disponibile nella [Guida per lo sviluppatore di AWS SDK per JavaScript v3](#).

Servizi utilizzati in questo esempio

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK per Python (Boto3)

Questo esempio mostra come registrare una AWS Lambda funzione come destinazione di un EventBridge evento Amazon pianificato. Il gestore Lambda scrive un messaggio intuitivo e i dati completi dell'evento su Amazon CloudWatch Logs per recuperarli in un secondo momento.

- Distribuzione di una funzione Lambda.
- Crea un evento EventBridge pianificato e rende la funzione Lambda la destinazione.
- Concede il permesso di EventBridge invocare la funzione Lambda.
- Stampa i dati più recenti dai CloudWatch registri per mostrare il risultato delle chiamate pianificate.
- Elimina tutte le risorse create durante la demo.

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

- CloudWatch Registri
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Utilizzo di Amazon SNS con un SDK AWS](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi serverless per Amazon SNS

I seguenti esempi di codice mostrano come usare Amazon SNS con. AWS SDKs

Esempi

- [Richiamo di una funzione Lambda da un trigger Amazon SNS](#)

Richiamo di una funzione Lambda da un trigger Amazon SNS

I seguenti esempi di codice mostrano come implementare una funzione Lambda che riceve un evento attivato dal ricevimento di messaggi da un argomento SNS. La funzione recupera i messaggi dal parametro dell'evento e registra il contenuto di ogni messaggio.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }
}
```

```
private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
ILambdaContext context)
{
    try
    {
        context.Logger.LogInformation($"Processed record
{record.Sns.Message}");

        // TODO: Do interesting work based on the new message
        await Task.CompletedTask;
    }
    catch (Exception e)
    {
        //You can use Dead Letter Queue to handle failures. By configuring a
        Lambda DLQ.
        context.Logger.LogError($"An error occurred");
        throw;
    }
}
```

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-lambda-go/events"
```

```

    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
    for _, record := range snsEvent.Records {
        processMessage(record)
    }
    fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
    message := record.SNS.Message
    fmt.Printf("Processed message: %s\n", message)
    // TODO: Process your record here
}

func main() {
    lambda.Start(handler)
}

```

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite Java.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;

```

```
import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Consumo di un evento SNS con JavaScript Lambda utilizzando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Consumo di un evento SNS con TypeScript Lambda utilizzando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```

): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}

```

PHP

SDK per PHP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite PHP.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/docs/runtimes/function

Another approach would be to create a custom runtime.

```

```
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-lambda-custom-runtime-for-php-a-practical-example/  
*/  
  
// Additional composer packages may be required when using Bref or any other PHP  
// functions runtime.  
// require __DIR__ . '/vendor/autoload.php';  
  
use Bref\Context\Context;  
use Bref\Event\Sns\SnsEvent;  
use Bref\Event\Sns\SnsHandler;  
  
class Handler extends SnsHandler  
{  
    public function handleSns(SnsEvent $event, Context $context): void  
    {  
        foreach ($event->getRecords() as $record) {  
            $message = $record->getMessage();  
  
            // TODO: Implement your custom processing logic here  
            // Any exception thrown will be logged and the invocation will be  
            marked as failed  
  
            echo "Processed Message: $message" . PHP_EOL;  
        }  
    }  
}  
  
return new Handler();
```

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite Python.


```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK per Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
    event['Records'].map { |record| process_message(record) }
end

def process_message(record)
    message = record['Sns']['Message']
    puts("Processing message: #{message}")
    rescue StandardError => e
        puts("Error processing message: #{e}")
```

```
    raise
end
```

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento SNS con Lambda tramite Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
// aws_lambda_events = { version = "0.10.0", default-features = false, features
//   = ["sns"] }
// lambda_runtime = "0.8.1"
// tokio = { version = "1", features = ["macros"] }
// tracing = { version = "0.1", features = ["log"] }
// tracing-subscriber = { version = "0.3", default-features = false, features =
//   ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for record in event.payload.records {
        process_record(&record)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);
}
```

```
    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Utilizzo di Amazon SNS con un SDK AWS](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Sicurezza di Amazon SNS

Il modello di [responsabilità AWS condivisa modello](#) si applica alla protezione dei dati in. Come descritto in questo modello, AWS è responsabile della protezione dell'infrastruttura globale che gestisce tutti i Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Questo contenuto include le attività di configurazione e gestione della sicurezza per AWS i servizi utilizzati. Per maggiori informazioni sulla privacy dei dati, consulta le [Domande frequenti sulla privacy dei dati](#). Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al [AWS Modello di responsabilità condivisa e GDPR](#) nel AWS Blog sulla sicurezza.

Ai fini della protezione dei dati, ti consigliamo di proteggere Account AWS le credenziali e 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.
- SSL/TLS Da utilizzare per comunicare con AWS le risorse. È consigliabile TLS 1.2 o versioni successive.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza soluzioni di AWS crittografia, insieme a tutti i controlli di sicurezza predefiniti all'interno 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 moduli crittografici convalidati FIPS 140-2 per l'accesso AWS tramite un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il [Federal Information Processing Standard \(FIPS\) 140-2](#).
- Protezione dei dati dei messaggi
 - La protezione dei dati dei messaggi è una nuova importante funzionalità di Amazon SNS
 - Usa la funzionalità di protezione dei dati dei messaggi per analizzare i messaggi alla ricerca di informazioni riservate o sensibili
 - Puoi garantire il controllo dei messaggi per tutti i contenuti che passano attraverso l'argomento
 - Puoi controllare l'accesso ai contenuti dei messaggi pubblicati sull'argomento e dei messaggi recapitati dall'argomento

Important

Ti suggeriamo vivamente di non inserire mai informazioni identificative sensibili, ad esempio i numeri di account dei clienti, in campi a formato libero, ad esempio un campo Nome. Ciò include quando lavori con Amazon SNS o altri Amazon Web Services utilizzando la console, l'API o. AWS CLI 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. Quando fornisci un URL a un server esterno, non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta a tale server.

Crittografia dei dati Amazon SNS

La protezione dei dati ha lo scopo di proteggere i dati sia in transito (durante la trasmissione verso e da Amazon SNS), sia quando sono inattivi (ovvero quando sono archiviati su disco nei data center Amazon SNS). È possibile proteggere i dati in transito tramite il protocollo Secure Sockets Layer (SSL) o tramite la crittografia lato client. Per impostazione predefinita, Amazon SNS archivia messaggi e file utilizzando la crittografia del disco. Puoi proteggere i dati inattivi richiedendo ad Amazon SNS di crittografare i tuoi messaggi prima di salvarli nel file system crittografato dei suoi data center. Amazon SNS consiglia di utilizzare SSE per una crittografia ottimizzata dei dati.

Protezione dei dati di Amazon SNS con crittografia lato server

La crittografia lato server (SSE) consente di archiviare dati sensibili in argomenti crittografati proteggendo il contenuto dei messaggi negli argomenti di Amazon SNS utilizzando chiavi gestite in (). AWS Key Management Service AWS KMS

SSE esegue la crittografia dei messaggi non appena vengono ricevuti da Amazon SNS. I messaggi vengono archiviati in forma crittografata e decrittografati solo quando vengono inviati.

- Per informazioni sulla gestione dell'SSE utilizzando Console di gestione AWS o il AWS SDK per Java (impostando l'`KmsMasterKeyId`attributo utilizzando le azioni [CreateTopic](#) e [SetTopicAttributes](#) API), vedere. [Configurazione della crittografia tematica di Amazon SNS con crittografia lato server](#)
- Per informazioni sulla creazione di argomenti crittografati utilizzando CloudFormation (impostando la `KmsMasterKeyId` proprietà utilizzando la [AWS::SNS::Topic](#) risorsa), consultate la Guida per l'AWS CloudFormation utente.

⚠ Important

Tutte le richieste agli argomenti con la funzione SSE abilitata devono utilizzare HTTPS e [Signature Version 4](#).

Per informazioni sulla compatibilità di altri servizi con argomenti crittografati, consulta la documentazione del servizio.

Amazon SNS supporta solo chiavi KMS di crittografia simmetrica. Non è possibile utilizzare nessun altro tipo di chiave KMS per crittografare le risorse del servizio. Per informazioni su come determinare se una chiave KMS è simmetrica o asimmetrica, consultare [Identifying asymmetric KMS keys](#) (Individuazione di chiavi KMS asimmetriche).

AWS KMS combina hardware e software sicuri e 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 chiavi KMS AWS gestite per Amazon SNS, 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.

Ambito della crittografia

SSE esegue la crittografia del corpo di un messaggio in un argomento Amazon SNS.

SSE non esegue la crittografia di quanto segue:

- Metadata dell'argomento (nome e attributo dell'argomento)
- Metadati del messaggio (oggetto, ID messaggio, time stamp e attributo)
- Policy di protezione dei dati
- Parametri per argomento

 Note

- Un messaggio viene crittografato solo se inviato dopo che la crittografia di una coda viene abilitata. Amazon SNS non crittografa i messaggi in backlog.
- Tutti i messaggi crittografati restano crittografati anche se la crittografia del relativo argomento è disabilitata.

Termini chiave

I seguenti termini chiave possono aiutarti a comprendere meglio le funzionalità di SSE . Per una descrizione dettagliata, consulta la [Documentazione di riferimento per l'API Amazon Simple Notification Service](#).

Chiave di dati

La chiave di crittografia dei dati (DEK) responsabile della crittografia dei contenuti dei messaggi Amazon SNS.

Per ulteriori informazioni, consulta [Chiave dati](#) nella AWS Key Management Service Guida per sviluppatori e [Pacchetto Crittografia](#) nella AWS Encryption SDK Guida per sviluppatori.

AWS KMS key ID

L'alias, l'alias ARN, l'ID chiave o l'ARN della chiave di un account personalizzato AWS KMS key, nel tuo account o AWS KMS in un altro account. Sebbene l'alias di AWS managed AWS KMS for Amazon SNS sia `alias/aws/sns` sempre, l'alias di un account AWS KMS personalizzato può, ad esempio, essere. `alias/MyAlias` Puoi utilizzare queste chiavi AWS KMS per proteggere i messaggi negli argomenti Amazon SNS.

 Note

Ricorda quanto segue:

- La prima volta che usi Console di gestione AWS per specificare il KMS AWS gestito per Amazon SNS per un argomento AWS KMS , crea AWS il KMS gestito per Amazon SNS.
- In alternativa, la prima volta che utilizzi l'Publishazione su un argomento con SSE abilitato, AWS KMS crea il KMS AWS gestito per Amazon SNS.

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 keys sezione della AWS KMS console o l'azione. [CreateKey](#) AWS KMS Per ulteriori informazioni, consultare [AWS KMS keys](#) e [Creating Keys](#) (Creazione di chiavi) nella Guida per gli sviluppatori di AWS Key Management Service . Per altri esempi di AWS KMS identificatori, consulta [KeyId](#) l'AWS Key Management Service API Reference. Per informazioni sulla ricerca degli AWS KMS identificatori, consulta [Find the Key ID and ARN](#) nella AWS Key Management Service Developer Guide.

 Important

Sono previsti costi aggiuntivi per l'utilizzo. AWS KMS Per ulteriori informazioni, consulta [Stima dei costi AWS KMS](#) e [Prezzi di AWS Key Management Service](#).

Gestione delle chiavi e dei costi di crittografia di Amazon SNS

Nelle sezioni seguenti vengono fornite informazioni sull'utilizzo delle chiavi gestite in AWS Key Management Service (AWS KMS).

 Note

Amazon SNS supporta solo chiavi KMS di crittografia simmetrica. Non è possibile utilizzare nessun altro tipo di chiave KMS per crittografare le risorse del servizio. Per informazioni su come determinare se una chiave KMS è simmetrica o asimmetrica, consultare [Identifying asymmetric KMS keys](#) (Individuazione di chiavi KMS asimmetriche).

Stima dei costi AWS KMS

Per prevedere i costi e comprendere meglio la tua AWS bolletta, potresti voler sapere con quale frequenza Amazon SNS utilizza la tua. AWS KMS key

 Note

Anche se la seguente formula può darti un'idea molto precisa dei costi previsti, i costi effettivi potrebbero essere più elevati a causa della natura diffusa di Amazon SNS.

Per calcolare il numero di richieste API (R) per argomento, usa la formula seguente:

$$R = B / D * (2 * P)$$

B è il periodo di fatturazione (in secondi).

D è il periodo di riutilizzo della chiave dati (in secondi—Amazon SNS riutilizza una chiave dati per un massimo di 5 minuti).

P è il numero di [principali](#) per la pubblicazione che effettuano invii all'argomento Amazon SNS.

Di seguito vengono riportati esempi di calcolo. Per informazioni dettagliate sui prezzi, consulta [Prezzi di AWS Key Management Service](#).

Esempio 1: calcolo del numero di chiamate AWS KMS API per 1 editore e 1 argomento

Questo esempio assume quanto segue:

- Il periodo di fatturazione è compreso tra il 1° e il 31 gennaio (2.678.400 secondi).
- Il periodo di riutilizzo della chiave di dati è di 5 minuti (300 secondi).
- C'è 1 argomento.
- C'è un principale per la pubblicazione.

$$2,678,400 / 300 * (2 * 1) = 17,856$$

Esempio 2: calcolo del numero di chiamate AWS KMS API 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 poter utilizzare SSE, è necessario configurare AWS KMS key le politiche per consentire la crittografia degli argomenti e la crittografia e la decrittografia dei messaggi. Per esempi e ulteriori informazioni sulle autorizzazioni di AWS KMS , consulta [AWS KMS Autorizzazioni API: Documentazione su operazioni e risorse](#) nella AWS Key Management Service Guida per sviluppatori. Per dettagli su come configurare un argomento Amazon SNS con crittografia lato server, consulta [Informazioni aggiuntive](#).

Note

È possibile gestire le autorizzazioni per le chiavi KMS utilizzando le policy IAM. Per ulteriori informazioni, consulta [Using IAM Policies with AWS KMS](#).
Sebbene sia possibile configurare le autorizzazioni globali per l'invio e la ricezione da Amazon SNS AWS KMS , è necessario nominare esplicitamente l' KMSs ARN completo di in regioni Resource specifiche nella sezione di una policy IAM.

È inoltre necessario assicurarsi che le politiche chiave di consentano le AWS KMS key autorizzazioni necessarie. Per eseguire questa operazione, denomina i principali che producono e utilizzano messaggi crittografati in Amazon SNS come utenti nella policy della chiave KMS.

In alternativa, puoi specificare AWS KMS le azioni richieste e l'ARN KMS in una policy IAM assegnata ai principali che pubblicano e sottoscrivono per ricevere messaggi crittografati in Amazon SNS. Per ulteriori informazioni, consulta [Gestione dell'accesso a AWS KMS](#) nella Guida per sviluppatori di AWS Key Management Service .

Se stai selezionando una chiave gestita dal cliente per l'argomento Amazon SNS e stai utilizzando alias per controllare l'accesso alle chiavi KMS utilizzando le policy IAM o le policy delle chiavi KMS con la chiave di condizione `kms:ResourceAliases`, assicurati che alla chiave gestita dal cliente selezionata sia associato anche un alias. Per ulteriori informazioni sull'uso degli alias per controllare l'accesso alle chiavi KMS, consulta [Using aliases to control access to KMS keys](#) (Utilizzo degli alias per controllare l'accesso alle chiavi KMS) nella Guida per gli sviluppatori di AWS Key Management Service .

Consentire a un utente di inviare messaggi a un argomento con SEE

L'editore deve avere le autorizzazioni `kms:GenerateDataKey*` e `kms:Decrypt` per la chiave AWS KMS key.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Abilita la compatibilità tra le fonti di eventi provenienti dai AWS servizi e gli argomenti crittografati

Diversi AWS servizi pubblicano eventi su argomenti di Amazon SNS. Per consentire a queste origini eventi di utilizzare argomenti crittografati, devi eseguire la seguente procedura.

1. Utilizzo di una chiave gestita dal cliente. Per ulteriori informazioni, consulta [Creazione di chiavi](#) nella Guida per gli sviluppatori di AWS Key Management Service .
2. Per consentire al AWS servizio di disporre delle `kms:Decrypt` autorizzazioni `kms:GenerateDataKey*` and, aggiungi la seguente dichiarazione alla politica KMS.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Origine eventi	Principale del servizio
Amazon CloudWatch	<code>cloudwatch.amazonaws.com</code>
CloudWatch Eventi Amazon	<code>events.amazonaws.com</code>
AWS CodeCommit	<code>codecommit.amazonaws.com</code>
AWS Database Migration Service	<code>dms.amazonaws.com</code>
AWS Directory Service	<code>ds.amazonaws.com</code>
Amazon DynamoDB	<code>dynamodb.amazonaws.com</code>
Amazon Inspector	<code>inspector.amazonaws.com</code>
Amazon Redshift	<code>redshift.amazonaws.com</code>
Amazon RDS	<code>events.rds.amazonaws.com</code>
Amazon Glacier	<code>glacier.amazonaws.com</code>
Amazon Simple Email Service	<code>ses.amazonaws.com</code>
Amazon Simple Storage Service	<code>s3.amazonaws.com</code>
AWS Snowball Edge	<code>importexport.amazonaws.com</code>
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager è costituito da due principi di servizio: <code>ssm-incidents.amazonaws.com</code> ; <code>ssm-contacts.amazonaws.com</code>

Note

Alcune fonti di eventi Amazon SNS richiedono di fornire un ruolo IAM (anziché il responsabile del servizio) nella AWS KMS key policy:

- [Amazon EC2 Auto Scaling](#)

- [Amazon Elastic Transcoder](#)
- [AWS CodePipeline](#)
- [AWS Config](#)
- [AWS Elastic Beanstalk](#)
- [AWS IoT](#)
- [EC2 Image Builder](#)

3. Aggiungi le chiavi di condizione `aws:SourceAccount` e `aws:SourceArn` per la policy delle risorse KMS per proteggere ulteriormente la chiave KMS da attacchi [confused deputy](#). Fare riferimento all'elenco della documentazione specifica del servizio (sopra) per i dettagli esatti in ciascun caso.

 Important

L'aggiunta di `aws:SourceAccount`, `aws:SourceArn`, e `aws:SourceOrgID` a una AWS KMS policy 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. [Abilitare SSE per il proprio argomento](#) utilizzando la chiave KMS.
5. Fornire l'ARN dell'argomento crittografato per l'origine eventi.

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.

Codice di stato HTTP: 400

KMSDisabledEccezione

La richiesta è stata rifiutata perché la chiave KMS specificata non è abilitata.

Codice di stato HTTP: 400

KMSInvalidStateException

La richiesta è stata rifiutata perché lo stato della risorsa specificata non è valido per questa richiesta. Per ulteriori informazioni, consulta [Key states of AWS KMS keys](#) (Stati chiave nelle chiavi AWS KMS keys) nella Guida per gli sviluppatori di AWS Key Management Service .

Codice di stato HTTP: 400

KMSNotFoundException

La richiesta è stata rifiutata perché non è possibile trovare l'entità o la risorsa specificata.

Codice di stato HTTP: 400

KMSOptInRequired

L'ID della chiave di AWS accesso richiede un abbonamento al servizio.

Codice di stato HTTP: 403

KMSThrottlingEccezione

La richiesta è stata negata a causa del throttling della richiesta. Per ulteriori informazioni sulla limitazione della larghezza di banda della rete consulta [Quote](#) nella Guida per gli sviluppatori di AWS Key Management Service .

Codice di stato HTTP: 400

Configurazione della crittografia tematica di Amazon SNS con crittografia lato server

Amazon SNS supporta la crittografia lato server (SSE) per proteggere il contenuto dei messaggi utilizzando (). AWS Key Management Service AWS KMS Segui le istruzioni riportate di seguito per abilitare SSE utilizzando la console Amazon SNS o CDK.

Opzione 1: abilita la crittografia utilizzando Console di gestione AWS

1. Accedi alla [console Amazon SNS](#).
2. Vai alla pagina Argomenti, seleziona l'argomento e scegli Modifica.
3. Espandere la sezione Encryption (Crittografia) e procedere come segue:
 - Attiva la crittografia su Abilita.
 - Seleziona la SNSchiave AWS gestita (alias/aws/sns) come chiave di crittografia. Questa opzione è selezionata per impostazione predefinita.
4. Scegli Save changes (Salva modifiche).

Note

- Chiave gestita da AWS Viene creato automaticamente se non esiste già.
- Se non vedi la chiave o non disponi di autorizzazioni sufficienti, chiedi all'amministratore `kms:ListAliases` e `kms:DescribeKey`.

Opzione 2: abilita la crittografia utilizzando AWS CDK

Per utilizzare la SNS chiave AWS gestita nella tua applicazione CDK, aggiungi il seguente frammento:

```
import software.amazon.awscdk.services.sns.*;
import software.amazon.awscdk.services.kms.*;
import software.amazon.awscdk.core.*;
```

```
public class SnsEncryptionExample extends Stack {
    public SnsEncryptionExample(final Construct scope, final String id) {
        super(scope, id);

        // Define the managed SNS key
        IKey snsKey = Alias.fromAliasName(this, "helloKey", "alias/aws/sns");

        // Create the SNS Topic with encryption enabled
        Topic.Builder.create(this, "MyEncryptedTopic")
            .masterKey(snsKey)
            .build();
    }
}
```

Informazioni aggiuntive

- Chiave KMS personalizzata: puoi specificare una chiave personalizzata, se necessario. Nella console Amazon SNS, seleziona la tua chiave KMS personalizzata dall'elenco o inserisci l'ARN.
- Autorizzazioni per chiavi KMS personalizzate: se utilizzi una chiave KMS personalizzata, includi quanto segue nella policy delle chiavi per consentire ad Amazon SNS di crittografare e decrittografare i messaggi:

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "sns.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

L'attivazione di SSE non modifica il modo in cui gli abbonati utilizzano i messaggi. AWS gestisce la crittografia e la decrittografia in modo trasparente. I messaggi rimangono crittografati anche quando sono inattivi e vengono decrittografati automaticamente prima della consegna agli abbonati. Per una sicurezza ottimale, AWS consiglia di abilitare HTTPS per tutti gli endpoint per garantire la trasmissione sicura dei messaggi.

Configurazione della crittografia degli argomenti di Amazon SNS con abbonamento crittografato alla coda Amazon SQS

Puoi abilitare la crittografia SSE (crittografia lato server) per un argomento per proteggerne i dati. Per consentire ad Amazon SNS di inviare messaggi a code crittografate Amazon SQS, la chiave gestita dal cliente associata alla coda Amazon SQS deve avere una dichiarazione di policy che conceda ad Amazon SNS l'accesso del principale del servizio alle operazioni `GenerateDataKey` e `Decrypt` dell'API AWS KMS. Per ulteriori informazioni sull'utilizzo di SSE, consulta [Protezione dei dati di Amazon SNS con crittografia lato server](#).

Questo argomento spiega come abilitare SSE per un argomento Amazon SNS con un abbonamento crittografato ad Amazon SQS queue utilizzando il Console di gestione AWS.

Fase 1: creazione di una chiave KMS personalizzata

1. Accedere alla [console AWS KMS](#) con un utente che dispone di almeno di una policy `AWSKeyManagementServicePowerUser`.
2. Scegli `Create a key` (Crea una chiave).
3. Per creare una chiave KMS di crittografia simmetrica, in `Key type` (Tipo di chiave) scegli `Symmetric` (Simmetrica).

Per informazioni su come creare una chiave KMS asimmetrica nella console AWS KMS, consultare [Creating asymmetric KMS keys \(console\)](#) (Creazione di chiavi KMS asimmetriche (console)).

4. In `Utilizzo della chiave`, l'opzione `Crittografa e decrittografa` è selezionata per impostazione predefinita.

Per informazioni su come creare le chiavi KMS che generano e verificano i codici MAC, consultare [Creating HMAC KMS keys](#) (Creazione di chiavi KMS HMAC).

Per informazioni sulle Opzioni avanzate, consultare [Special-purpose keys](#) (Chiavi per uso speciale).

5. Scegli Next (Successivo).
6. Digita un alias per la chiave KMS. 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, consultare [ABAC for AWS KMS](#) (ABAC per KMS) e [Using aliases to control access to KMS keys](#) (Utilizzo degli alias per controllare l'accesso alle chiavi KMS).

Un alias è un nome visualizzato che può essere utilizzato per identificare la chiave KMS. È consigliabile scegliere un alias che indica il tipo di dati che desideri proteggere o l'applicazione che desideri utilizzare con la chiave KMS.

Gli alias sono obbligatori quando si crea una chiave KMS nella Console di gestione AWS. Sono opzionali quando si utilizza l'[CreateKey](#) operazione.

7. (Facoltativo) Digita una descrizione per la chiave KMS.

Puoi aggiungere una descrizione ora o aggiornarla in qualsiasi momento, a meno che lo [stato della chiave](#) non sia Pending Deletion o Pending Replica Deletion. Per aggiungere, modificare o eliminare la descrizione di una chiave gestita dal cliente esistente, [modifica la descrizione](#) in Console di gestione AWS o utilizza l'[UpdateKeyDescription](#) operazione.

8. (Facoltativo) Digitare una chiave di tag e un valore di tag facoltativo. Per aggiungere più di un tag alla chiave KMS scegli Add tag (Aggiungi tag).

 Note

L'applicazione o l'eliminazione di un tag chiave KMS può consentire o negare l'autorizzazione alla chiave KMS. Per i dettagli, consultare [ABAC for AWS KMS](#) (ABAC

per KMS) e [Using tags to control access to KMS keys](#) (Utilizzo dei tag per controllare l'accesso alle chiavi KMS).

Quando aggiungi 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 chiave KMS. Per informazioni sull'assegnazione di tag alle chiavi KMS, consultare [Tagging keys](#) (Assegnazione di tag alle chiavi) e [ABAC for AWS KMS](#) (ABAC per KMS).

9. Scegli Next (Successivo).
10. Seleziona i ruoli e gli utenti IAM che possono gestire la chiave KMS.

Note

Questa politica chiave offre il Account AWS pieno controllo di questa chiave KMS. Consente agli amministratori dell'account di utilizzare policy IAM per concedere ad altri principali l'autorizzazione per la gestione della chiave KMS. Per informazioni dettagliate, consultare [Default key policy](#) (Policy della chiave predefinita).

Le best practice di IAM disincentivano l'uso di utenti IAM con credenziali a lungo termine. Quando possibile, utilizza i ruoli IAM che forniscono credenziali temporanee. Per i dettagli, consultare [Best practice per la sicurezza in IAM](#) nella Guida per l'utente IAM.

11. (Facoltativo) Per impedire ai ruoli e agli utenti IAM selezionati di eliminare questa chiave KMS, nella sezione Eliminazione chiave nella parte inferiore della pagina, deseleziona la casella di controllo Consenti agli amministratori delle chiavi di eliminare questa chiave.
12. Scegli Next (Successivo).
13. Selezionare i ruoli e gli utenti IAM che possono utilizzare la chiave nelle [operazioni di crittografia](#). 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.

Fase 2: creazione di un argomento Amazon SNS crittografato

1. Accedi alla [console Amazon SNS](#).
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, vengono visualizzati la Descrizione, l'account e l'ARN della chiave gestita dal cliente.

Important

Se non si è il proprietario della chiave gestita dal cliente, oppure se si effettua l'accesso con un account che non dispone delle autorizzazioni `kms:ListAliases` e `kms:DescribeKey`, non è possibile visualizzare le informazioni sulla chiave gestita dal cliente sulla console Amazon SNS.

Richiedere al proprietario della chiave gestita dal cliente di concedere queste autorizzazioni. Per esempi e ulteriori informazioni consulta [AWS KMS Autorizzazioni API: e Documentazione su operazioni e risorse](#) nella AWS Key Management Service Guida per sviluppatori.

- c. Per la chiave gestita dal cliente, scegli quella MyCustomKey [che hai creato in precedenza](#), quindi scegli Abilita la crittografia lato server.
6. Scegli Save changes (Salva modifiche).

SSE è abilitato per l'argomento e viene MyTopic visualizzata la pagina.

Lo stato Crittografia, Account AWS, Chiave gestita dal cliente, ARN della chiave gestita dal cliente e Descrizione dell'argomento sono visualizzati nella scheda Crittografia.

Il nuovo argomento crittografato viene visualizzato nell'elenco degli argomenti.

Fase 3: creazione e abbonamento a code Amazon SQS crittografate

1. Accedere alla [console Amazon SQS](#).
2. Scegli Create New Queue (Crea nuova coda).
3. Nella pagina Create New Queue (Crea nuova coda), procedi come indicato di seguito:
 - a. Immetti Queue Name (Nome coda) (ad esempio, MyEncryptedQueue1).
 - b. Scegli Standard Queue (Coda standard), quindi Configure Queue (Configura coda).
 - c. Scegli Use SSE (Usa SSE).
 - d. Infatti AWS KMS key, scegli [quello MyCustomKey che 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. Nella console Amazon SQS, seleziona MyEncryptedQueue1 e MyEncryptedQueue2, quindi scegli Operazioni coda, Sottoscrivi code ad argomento SNS.
6. Nella finestra di dialogo Sottoscrivi 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 [console Amazon SNS](#).
2. Nel pannello di navigazione, scegliere Argomenti.
3. Dall'elenco degli argomenti, scegli MyEncryptedTopic quindi seleziona Pubblica messaggio.
4. Nella pagina Publish a message (Pubblica un messaggio), procedi come indicato di seguito:
 - a. (Facoltativo) Nella sezione Message details (Dettagli messaggio), immettere il valore per Subject (Oggetto) (ad esempio, Testing message publishing).
 - b. Nella sezione Message body (Corpo messaggio), immettere il corpo del messaggio (ad esempio My message body is encrypted at rest.).
 - c. Seleziona Publish message (Pubblica messaggio).

Il messaggio viene pubblicato nelle code crittografate sottoscritte.

Fase 5: verifica della consegna del messaggio

1. Accedere alla [console Amazon SQS](#).
2. Dall'elenco delle code, scegli MyEncryptedQueue1, quindi 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 traffico Amazon SNS con endpoint VPC

Un endpoint Virtual Private Cloud (VPC) per Amazon SNS è un'entità logica all'interno di un VPC che consente la connettività solo ad Amazon SNS. Il VPC instrada le richieste ad Amazon SNS e le risposte al VPC. Nelle sezioni seguenti vengono fornite informazioni sull'utilizzo degli endpoint VPC e sulla creazione delle policy di endpoint VPC.

Se utilizzi Amazon Virtual Private Cloud (Amazon VPC) per ospitare AWS le tue risorse, puoi stabilire una connessione privata tra il tuo VPC e Amazon SNS. Con questa connessione, puoi pubblicare messaggi nei tuoi argomenti Amazon SNS senza inviarli tramite l'Internet pubblico.

Amazon VPC è un AWS servizio che puoi utilizzare per avviare AWS risorse in una rete virtuale definita dall'utente. Con un VPC, detieni il controllo delle impostazioni della rete, come l'intervallo di indirizzi IP, le sottoreti, le tabelle di routing e i gateway di rete. Per collegare il tuo VPC a Amazon SNS, definisci un endpoint VPC dell'interfaccia. Questo tipo di endpoint ti consente di connettere il tuo VPC AWS ai servizi. L'endpoint offre una connettività scalabile e affidabile a Amazon SNS senza richiedere un Internet gateway, un'istanza NAT (Network Address Translation) o una connessione VPN. Per ulteriori informazioni, consulta [Accedere a un endpoint VPC Servizio AWS con interfaccia nella Amazon VPC](#) User Guide.

Le informazioni contenute in questa sezione sono per gli utenti di Amazon VPC. Per ulteriori informazioni e per iniziare a creare un VPC, consulta [Plan your VPC nella Amazon VPC](#) User Guide.

Note

Gli endpoint VPC non consentono di sottoscrivere un argomento Amazon SNS a un indirizzo IP privato.

Creazione di un endpoint Amazon VPC per Amazon SNS

Per pubblicare messaggi negli argomenti Amazon SNS da Amazon VPC, crea un endpoint VPC di interfaccia. Quindi, puoi pubblicare messaggi agli argomenti mantenendo il traffico all'interno della rete gestita con VPC.

Utilizza le informazioni riportate di seguito per creare l'endpoint e testare la connessione tra VPC e Amazon SNS. In alternativa, per una procedura guidata che consente di iniziare da zero, consulta [Pubblicazione di un messaggio Amazon SNS da Amazon VPC](#).

Creazione dell'endpoint

Puoi creare un endpoint Amazon SNS nel tuo VPC utilizzando un AWS SDK Console di gestione AWS AWS CLI, l'API Amazon SNS oppure AWS CloudFormation

Per informazioni sulla creazione e sulla configurazione di un endpoint utilizzando la console Amazon VPC o il AWS CLI, consulta la sezione [Creazione di un endpoint di interfaccia](#) nella Guida per l'utente di Amazon VPC.

Important

Puoi utilizzare Amazon Virtual Private Cloud solo con endpoint Amazon SNS HTTPS.

Quando si crea un endpoint, occorre specificare Amazon SNS come servizio a cui desideri che il tuo VPC si connetta. Nella console Amazon VPC, i nomi dei servizi variano in base alla regione. Ad esempio, se si sceglie Stati Uniti orientali (Virginia settentrionale), il nome del servizio è `com.amazonaws.us-east-1.sns`.

Quando configuri Amazon SNS per inviare messaggi da Amazon VPC, devi abilitare il DNS privato e specificare gli endpoint nel formato `sns.us-east-2.amazonaws.com`.

Il DNS privato non supporta endpoint precedenti come `queue.amazonaws.com` o `us-east-2.queue.amazonaws.com`.

Per informazioni sulla creazione e configurazione di un endpoint utilizzando AWS CloudFormation, consulta la risorsa nella Guida per l'utente. [AWS::EC2::VPCEndpoint](#) AWS CloudFormation

Verifica la connessione tra il tuo VPC e Amazon SNS

Dopo aver creato un endpoint per Amazon SNS, puoi pubblicare i messaggi dal VPC agli argomenti Amazon SNS. Per testare questa connessione, procedi come indicato di seguito:

1. Connettiti a un' EC2 istanza Amazon che risiede nel tuo VPC. Per informazioni sulla connessione, consulta [Connect to Your Linux Instance](#) o [Connecting to Your Windows Instance](#) nella EC2 documentazione di Amazon.

Ad esempio, per connettersi a un'istanza Linux utilizzando un client SSH, esegui il comando seguente da un terminale:

```
$ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
```

Dove:

- `ec2-key-pair.pem` è il file che contiene la coppia di chiavi 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.
2. Dalla tua istanza, utilizza il comando Amazon SNS [publish](#) con la AWS CLI. È possibile inviare un messaggio semplice a un argomento con il comando seguente:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```


Dove:

- `aws-region` è AWS la regione in cui si trova l'argomento.
- `sns-topic-arn` è l'Amazon Resource Name (ARN) dell'argomento. Per ottenere l'ARN dalla [console Amazon SNS](#): scegli Argomenti, quindi trova l'argomento e il valore nella colonna ARN.

Se il messaggio è stato ricevuto da Amazon SNS, il terminale stamperà un ID messaggio, come il seguente:

```
{
  "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Creazione di una policy per endpoint VPC di Amazon per Amazon SNS

È possibile creare una policy per gli endpoint VPC di Amazon per Amazon SNS per specificare quanto segue:

- Il principale che può eseguire azioni.
- Le azioni che possono essere eseguite.
- Le risorse sui cui si possono eseguire operazioni.

Per ulteriori informazioni, consultare [Controllo degli accessi ai servizi con endpoint VPC](#) nella Guida per l'utente di Amazon VPC.

Il seguente esempio di policy di endpoint VPC specifica che l'utente IAM `MyUser` è autorizzato a pubblicare nell'argomento Amazon SNS `MyTopic`.

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

```
}]  
}
```

Non si può accedere a quanto segue:

- Altre operazioni API Amazon SNS, come `sns:Subscribe` e `sns:Unsubscribe`.
- Altri utenti IAM e regole che provano a utilizzare questo endpoint VPC.
- `MyUser` che pubblica in un altro argomento Amazon SNS.

Note

L'utente IAM può ancora utilizzare altre operazioni API Amazon SNS dall'esterno del VPC.

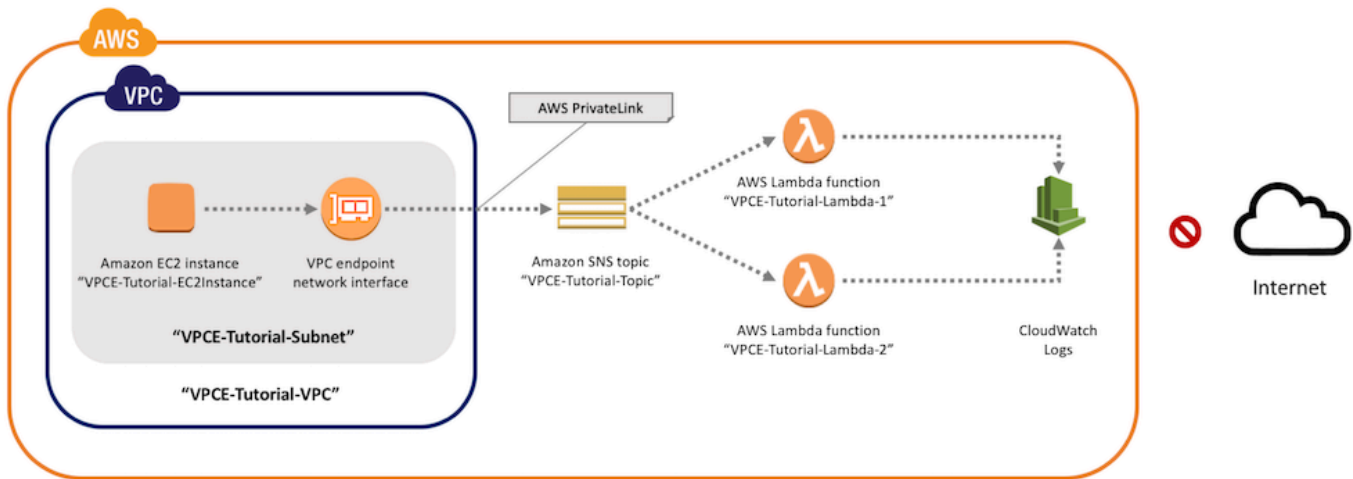
Pubblicazione di un messaggio Amazon SNS da Amazon VPC

In questa sezione viene descritto come pubblicare in un argomento Amazon SNS mantenendo i messaggi protetti in una rete privata. Puoi pubblicare un messaggio da un'istanza Amazon EC2 ospitata in Amazon Virtual Private Cloud (Amazon VPC). Il messaggio rimane all'interno della AWS rete senza viaggiare sulla rete Internet pubblica. Per pubblicare messaggi privatamente da un VPC, è possibile migliorare la sicurezza del traffico tra le applicazioni e Amazon SNS. Questa sicurezza è importante quando si pubblicano informazioni personali identificabili (PII) sui tuoi clienti oppure quando l'applicazione è soggetta a normative di mercato. Ad esempio, la pubblicazione in privato è utile se si tratta di un sistema sanitario che deve rispettare l'Health Insurance Portability and Accountability Act (HIPAA) o di un sistema finanziario che deve essere conforme al Payment Card Industry Data Security Standard (PCI DSS).

La procedura generale da seguire è riportata di seguito:

- Usa un AWS CloudFormation modello per creare automaticamente una rete privata temporanea nel tuo Account AWS.
- Creare un endpoint VPC che collega VPC con Amazon SNS.
- Accedere a un'istanza Amazon EC2 e pubblicare un messaggio in privato a un argomento Amazon SNS.
- Verificare che il messaggio sia stato consegnato correttamente.
- Elimina le risorse che hai creato durante questo processo in modo che non rimangano nel tuo Account AWS.

Il diagramma seguente illustra la rete privata che crei nel tuo AWS account durante il completamento di questi passaggi:



Questa rete è costituita da un VPC che contiene un'istanza Amazon EC2. L'istanza si collega a Amazon SNS; attraverso un endpoint VPC di interfaccia. Questo tipo di endpoint si connette a servizi alimentati da AWS PrivateLink. Con questa connessione stabilita, puoi accedere all'istanza Amazon EC2 e pubblicare messaggi all'argomento Amazon SNS, anche se la rete è disconnessa dalla rete Internet pubblica. L'argomento 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](#)
- [Fase 1: Creazione di una coppia di chiavi di Amazon EC2](#)
- [Fase 2: Creare le risorse AWS](#)
- [Fase 3: Conferma che l'istanza Amazon EC2; non dispone di accesso Internet](#)
- [Fase 4: Creazione di un endpoint Amazon VPC per Amazon SNS](#)
- [Fase 5: Pubblicazione di un messaggio nel proprio argomento Amazon SNS](#)
- [Fase 6: verifica delle consegne del messaggio](#)
- [Fase 7: pulire](#)
- [Risorse correlate](#)

Prima di iniziare

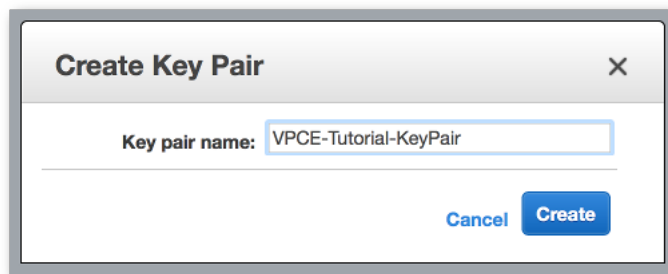
Prima di iniziare è necessario un account Amazon Web Services (AWS). Quando ti registri, il tuo account viene automaticamente registrato per tutti i servizi in AWS, inclusi Amazon SNS e Amazon VPC. Se non hai già creato un account, vai a <https://aws.amazon.com/>, quindi scegli Crea un account gratuito.

Fase 1: Creazione di una coppia di chiavi di Amazon EC2

Una coppia di chiavi viene utilizzata per effettuare l'accesso a un'istanza Amazon EC2. È costituita da una chiave pubblica utilizzata per crittografare le informazioni di accesso e una chiave privata utilizzata per decifrarle. Quando crei una coppia di chiavi, viene scaricata una copia della chiave privata. Successivamente, una coppia di chiavi viene utilizzata per effettuare l'accesso a un'istanza Amazon EC2. Per effettuare l'accesso, è necessario specificare il nome della coppia di chiavi e fornire la chiave privata.

Per creare la coppia di chiavi

1. Accedi Console di gestione AWS e apri la console Amazon EC2 all'indirizzo. <https://console.aws.amazon.com/ec2/>
2. Nel menu di navigazione a sinistra, individua la sezione Network & Security (Rete e sicurezza). Quindi scegli Key Pairs (Coppie di chiavi).
3. Scegli Crea coppia di chiavi.
4. Nella finestra Create Key Pair (Crea coppia di chiavi), per Key pair name (Nome coppia di chiavi), digita **VPCE-Tutorial-KeyPair**. Quindi scegli Create (Crea).



5. Il file della chiave privata viene automaticamente scaricato dal browser. Salvalo in un posto sicuro. Amazon EC2 assegna al file un'estensione .pem.
6. (Facoltativo) Se usi un client SSH su un computer Mac o Linux per connetterti alla tua istanza, usa il comando `chmod` per impostare le autorizzazioni del file della chiave privata in modo da essere l'unico a poterlo leggere:

- a. Apri un terminale e vai alla directory che contiene la chiave privata:

```
$ cd /filepath_to_private_key/
```

- b. Imposta le autorizzazioni utilizzando il comando seguente:

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

Fase 2: Creare le risorse AWS

Per configurare l'infrastruttura, si utilizza un CloudFormation modello. Un modello è un file che funge come schema per la creazione di risorse AWS , come istanze Amazon EC2 e argomenti Amazon SNS. Il modello per questo processo è disponibile GitHub e può essere scaricato.

Fornisci il modello e fornisci CloudFormation le risorse di cui hai bisogno come pila del tuo Account AWS. CloudFormation Uno stack è una raccolta di risorse che puoi gestire come un'unità singola. Una volta completati questi passaggi, puoi 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:

- Un VPC e le risorse di rete associate, tra cui una sottorete, un gruppo di sicurezza, un gateway Internet e una tabella di routing.
- Un'istanza Amazon EC2; avviata nella sottorete nel VPC.
- Argomento Amazon SNS
- Due AWS Lambda funzioni. Queste funzioni ricevono messaggi pubblicati sull'argomento Amazon SNS e registrano gli eventi in CloudWatch Logs.
- CloudWatch Metriche e log di Amazon.
- Un ruolo IAM che consente all'istanza Amazon EC2 di utilizzare Amazon SNS e un ruolo IAM che consente alle funzioni Lambda di scrivere nei log. CloudWatch

Per creare le risorse AWS

1. Scarica il [file modello](#) dal GitHub sito Web.
2. Accedi alla [console 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**

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name

Parameters

KeyName Name of an existing EC2 KeyPair to enable SSH access to the instance

SSHLocation The IP address range that can be used to SSH to the EC2 instance

- 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 CloudFormation potrebbe creare risorse IAM con nomi personalizzati.
9. Scegli Create (Crea).

La CloudFormation console apre la pagina Stacks. Lo VPCE-Tutorial-Stack stato è CREATE_IN_PROGRESS. Dopo pochi minuti, quando il processo di creazione è stato completato, lo stato diventa CREATE_COMPLETE.

Create Stack

▼

Actions

Design template

Filter: Active

▼

By Stack Name

	Stack Name	Created Time	Status	Description
☒	VPCE-Tutorial-Stack	2018-05-18 16:38:06 UTC-0700	CREATE_COMPLETE	CloudFormation Template for SNS VPC Endpoints Tutorial

Tip

Scegli il pulsante Refresh (Aggiorna) per visualizzare l'ultimo stato dello stack.

Fase 3: Conferma che l'istanza Amazon EC2; non dispone di accesso Internet

L'istanza Amazon EC2 che è stata avviata nel VPC nella fase precedente non dispone di accesso a Internet. Non consente il traffico in uscita e non è in grado di pubblicare messaggi a Amazon SNS. Verifica accedendo all'istanza. Quindi, prova a connetterti a un endpoint pubblico e ad inviare un messaggio a Amazon SNS.

A questo punto della procedura il tentativo di pubblicazione ha esito negativo. In un secondo momento, dopo aver creato un endpoint VPC per Amazon SNS il tentativo di pubblicazione va a buon fine.

Eseguire la connessione all'istanza di Amazon EC2.

1. Apri la console Amazon EC2 all'indirizzo <https://console.aws.amazon.com/ec2/>.
2. Nel menu di navigazione a sinistra, individua la sezione Instances (Istanze). Quindi scegli Instances (Istanze).
3. Nell'elenco delle istanze, selezionare VPCE -. Tutorial-EC2Instance
4. Copia il nome host fornito nella colonna DNS pubblico.

Launch Instance

Connect

Actions

Filter by tags and attributes or search by keyword

2

⌵ ⌴

☐	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)
☒	VPCE-Tutorial-EC2Instance	i-03011ba034402c01b	t2.micro	us-east-1c	● running	✓ 2/2 checks ...	None	📍 ec2-34-204-147-55.compute-1.amazonaws.com

5. Apri un terminale. Dalla directory che contiene la coppia di key pair, connettiti all'istanza utilizzando il seguente comando, *instance-hostname* dov'è il nome host che hai copiato dalla console Amazon EC2:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

Per verificare che l'istanza non dispone di una connessione a Internet

- Nel terminale, tenta di connetterti a un endpoint pubblico, ad esempio amazon.com:

```
$ ping amazon.com
```

Poiché il tentativo di connessione ha esito negativo, è possibile annullare in qualsiasi momento (Ctrl+C in Windows o Comando+C su macOS).

Per verificare che l'istanza non dispone di una connessione a Amazon SNS

1. Accedi alla [console Amazon SNS](#).
2. Nel menu di navigazione a sinistra, scegli Topics (Argomenti).
3. Nella pagina Topics (Argomenti), copia l'Amazon Resource Name (ARN) per l'argomento VPCE-Tutorial-Topic.
4. Nel terminale, tenta di pubblicare un messaggio all'argomento:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Poiché il tentativo di pubblicazione ha esito negativo, è possibile annullare in qualsiasi momento.

Fase 4: Creazione di un endpoint Amazon VPC per Amazon SNS

Per collegare il tuo VPC a Amazon SNS, definisci un endpoint VPC dell'interfaccia. Dopo aver aggiunto l'endpoint, è possibile accedere all'istanza Amazon EC2 nel proprio VPC, quindi è possibile utilizzare l'API di Amazon SNS. È possibile pubblicare messaggi all'argomento che vengono pubblicati privatamente. Rimangono 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.

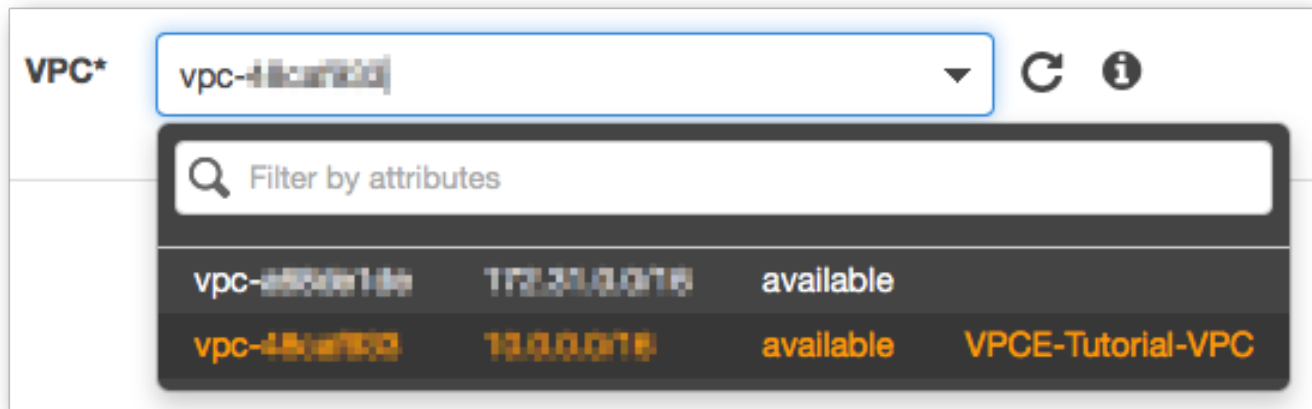
Per creare l'endpoint

1. Apri la console Amazon VPC all'indirizzo <https://console.aws.amazon.com/vpc/>.

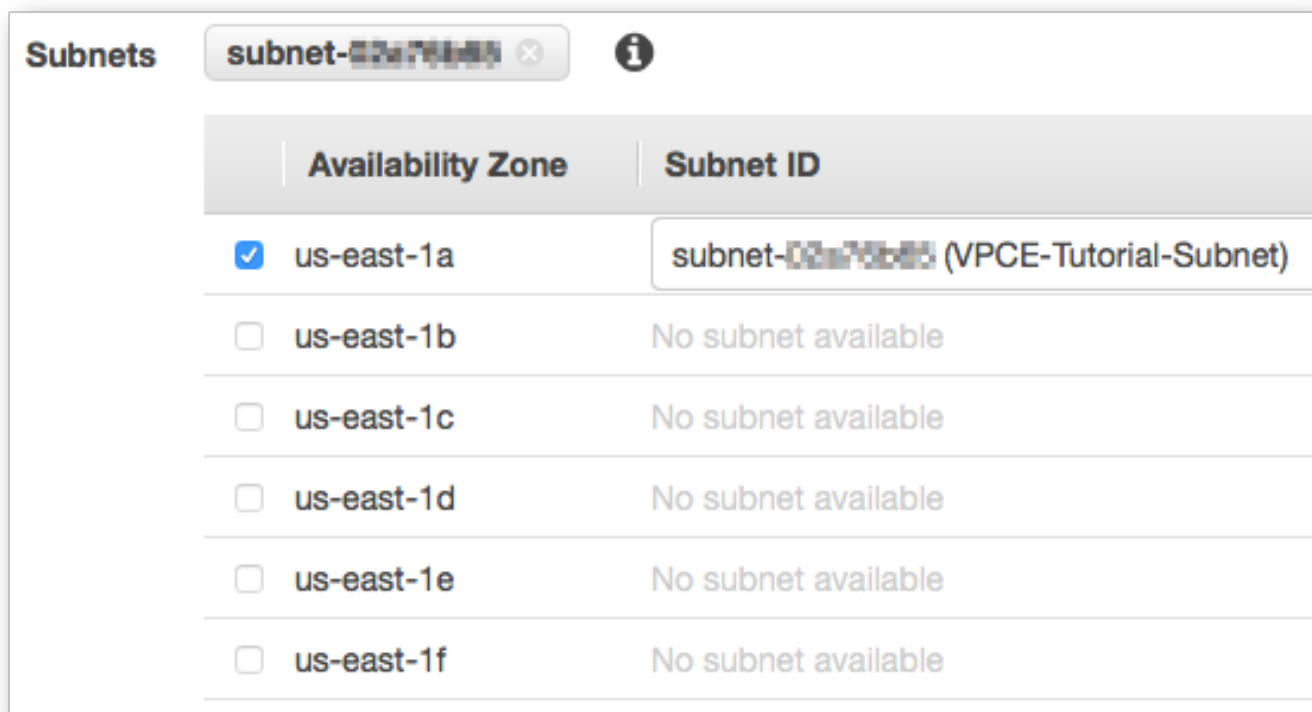
2. Nel menu di navigazione a sinistra, scegli Endpoints.
3. Scegliere Create Endpoint (Crea endpoint).
4. Nella pagina Crea endpoint, per Categoria di servizio lascia la scelta predefinita di AWS Servizi.
5. Per Nome servizio, scegli il nome del servizio per Amazon SNS;

I nomi dei servizi variano in base alla regione scelta. Ad esempio, se hai scelto Stati Uniti orientali (Virginia settentrionale), il nome del servizio è com.amazonaws. **us-east-1**.sns.

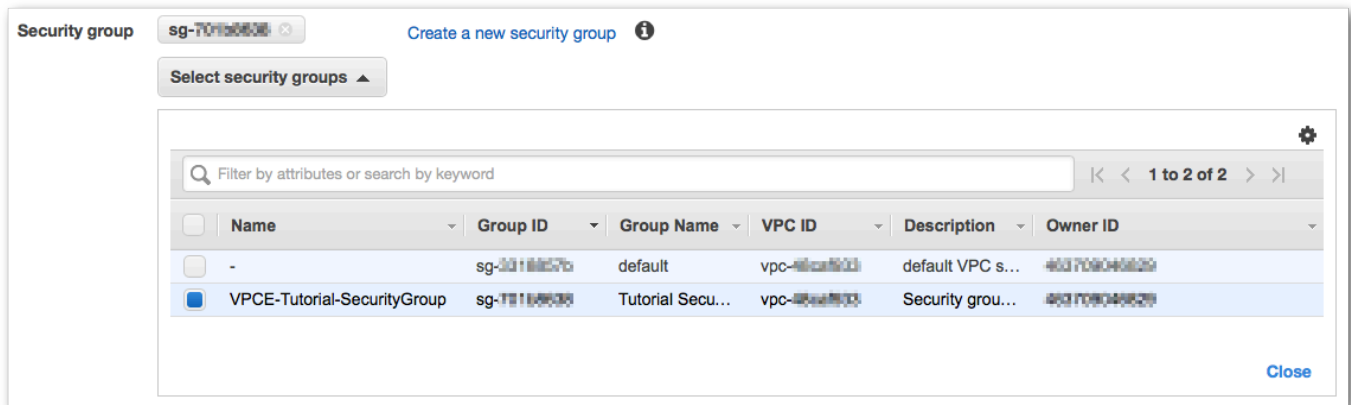
6. Per VPC, scegli il VPC che ha il nome VPCE-Tutorial-VPC.



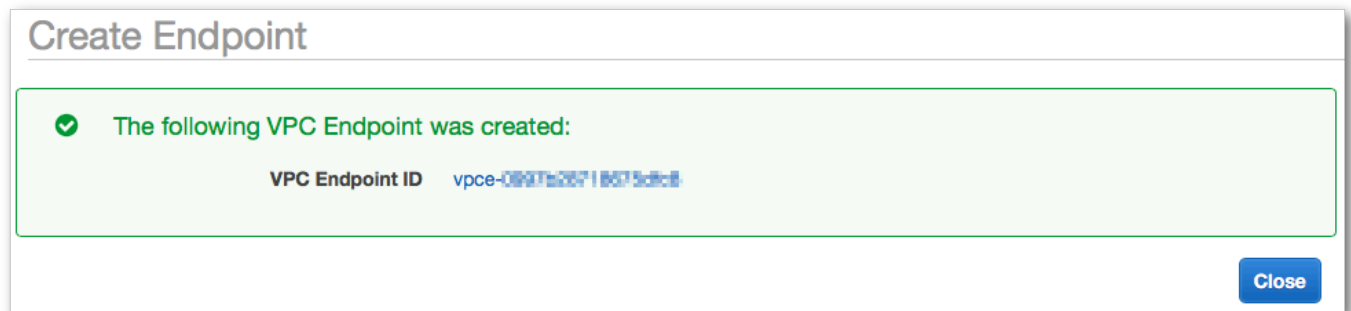
7. Per Subnets (Sottoreti), scegli la sottorete che ha VPCE-Tutorial-Subnet nell'ID della sottorete.



8. Per Enable Private DNS Name (Abilita nome DNS privato), seleziona Enable for this endpoint (Abilita per questo endpoint).
9. Per il gruppo di sicurezza, scegli Seleziona gruppo di sicurezza e scegli VPCE-Tutorial -. SecurityGroup

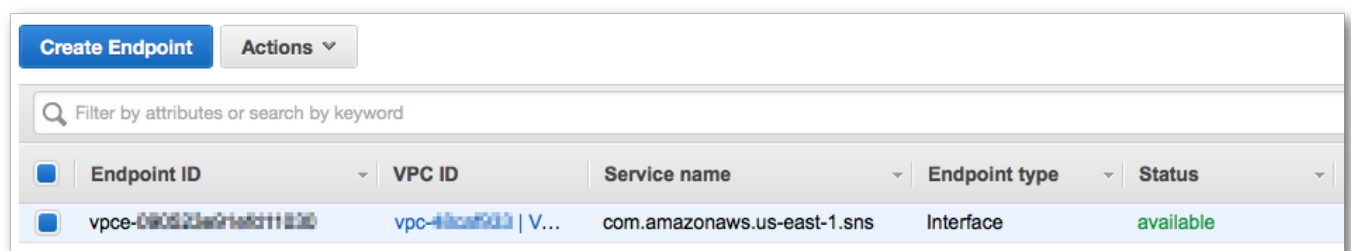


10. Seleziona Crea endpoint. La console Amazon VPC conferma che è stato creato un endpoint VPC.



11. Scegli Chiudi.

La console Amazon VPC apre la pagina Endpoints. Il nuovo endpoint ha lo stato pending (in sospeso). Dopo pochi minuti, quando il processo di creazione è stato completato, lo stato diventa available (disponibile).



Fase 5: Pubblicazione di un messaggio nel proprio argomento Amazon SNS

Ora che il VPC include un endpoint per Amazon SNS, può possibile accedere all'istanza Amazon EC2 e pubblicare messaggi nell'argomento.

Per pubblicare un messaggio

1. Se il terminale non è più connesso all'istanza Amazon EC2, effettuare nuovamente il collegamento:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Eseguire lo stesso comando eseguito precedentemente per pubblicare un messaggio nel proprio argomento Amazon SNS. Questa volta, il tentativo di pubblicazione va a buon fine e Amazon SNS; restituisce un ID messaggio:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

{
  "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

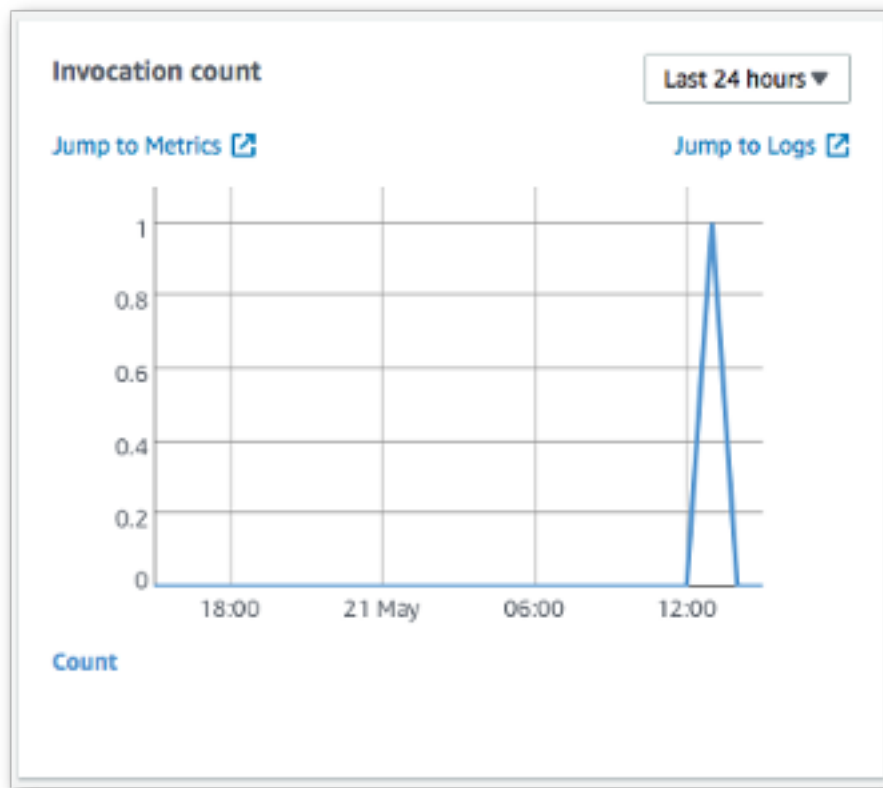
Fase 6: verifica delle consegne del messaggio

Quando l'argomento Amazon SNS riceve un messaggio, distribuisce il messaggio inviandolo alle due funzioni Lambda di sottoscrizione. Quando queste funzioni ricevono il messaggio, registrano l'evento CloudWatch nei registri. 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

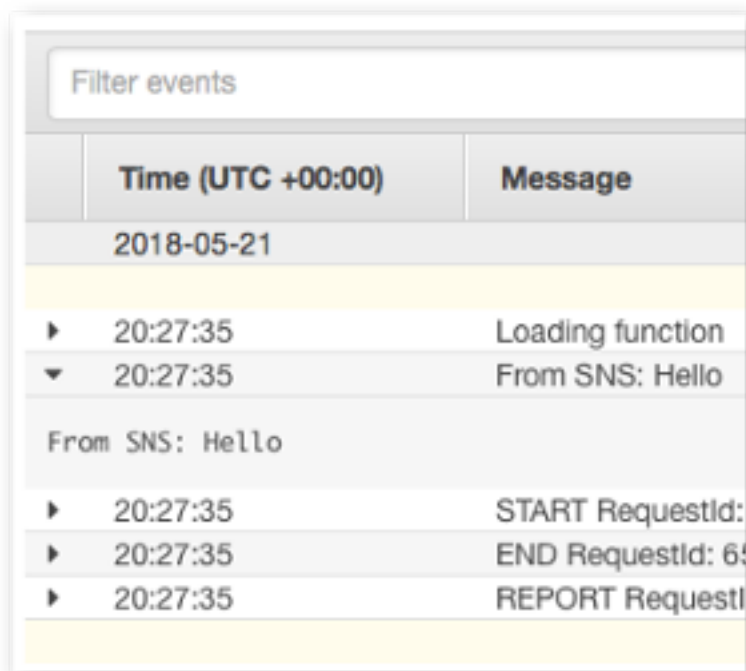
1. Apri la console all' AWS Lambda indirizzo. <https://console.aws.amazon.com/lambda/>
2. Nella pagina Functions (Funzioni), scegli VPCE-Tutorial-Lambda-1.
3. Selezionare Monitoring (Monitoraggio).
4. Controlla il grafico Invocation count (Conteggio invocazioni). Questo grafico mostra il numero di volte che la funzione Lambda; è stata eseguita.

Il conteggio di invocazioni corrisponde al numero di volte in cui è stato pubblicato un messaggio all'argomento.



Per verificare che i CloudWatch registri siano stati aggiornati

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.



Filter events	
Time (UTC +00:00)	Message
2018-05-21	
▶ 20:27:35	Loading function
▼ 20:27:35	From SNS: Hello
	From SNS: Hello
▶ 20:27:35	START RequestId:
▶ 20:27:35	END RequestId: 65
▶ 20:27:35	REPORT RequestId:

- 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 il gruppo di log/aws/lambda/VPCE-Tutorial-Lambda-2/.

Complimenti! Aggiungendo un endpoint per Amazon SNS a un VPC, è possibile pubblicare un messaggio in un argomento all'interno della rete gestita dal VPC. Il messaggio è stato pubblicato privatamente senza essere esposto sulla rete Internet pubblica.

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 a tuo carico. Account AWS

In primo luogo, elimina l'endpoint VPC utilizzando la console Amazon VPC. Quindi, elimina le altre risorse che hai creato eliminando lo stack nella console. CloudFormation Quando elimini uno stack, CloudFormation rimuove le risorse dello stack dal tuo. Account AWS

Per eliminare l'endpoint VPC

1. Apri la console Amazon VPC all'indirizzo <https://console.aws.amazon.com/vpc/>.
2. Nel menu di navigazione a sinistra, scegli Endpoints.
3. Seleziona l'endpoint creato.

4. Scegli Actions (Operazioni), quindi Delete Endpoint (Elimina endpoint).
5. Nella finestra Delete Endpoint (Elimina endpoint), scegli Yes, Delete (Sì, elimina).

Lo stato dell'endpoint diventa deleting (in eliminazione). Quando l'eliminazione viene completata, l'endpoint viene rimosso dalla pagina.

Per eliminare il tuo stack CloudFormation

1. Apri la CloudFormation console in <https://console.aws.amazon.com/cloudformazione>.
2. Selezionare lo stack VPCE-Tutorial-Stack.
3. Scegliere Actions (Operazioni), quindi Delete Stack (Elimina stack).
4. Nella finestra Delete Stack (Elimina stack), scegli Yes, Delete (Sì, elimina).

Lo stato dello stack diventa DELETE_IN_PROGRESS. Quando l'eliminazione viene completata, lo stack viene rimosso dalla pagina.

Risorse correlate

Per ulteriori informazioni, consulta le risorse seguenti.

- [AWS Blog sulla sicurezza: Protezione dei messaggi pubblicati su Amazon SNS con AWS PrivateLink](#)
- [What Is Amazon VPC?](#)
- [Endpoint VPC](#)
- [Che cos'è Amazon EC2?](#)
- [Concetti di CloudFormation](#)

Connect ad Amazon SNS utilizzando endpoint Dual-stack (e) IPv4 IPv6

Gli endpoint dual-stack supportano sia il traffico che il traffico. IPv4 IPv6 Quando effettui una richiesta a un endpoint dual-stack, l'URL dell'endpoint si risolve in un indirizzo o in un indirizzo. IPv4 IPv6 [Per ulteriori informazioni sugli endpoint dual-stack e FIPS, consulta la guida di riferimento SDK.](#)

Amazon SNS supporta endpoint dual-stack regionali, il che significa che devi specificare la AWS regione come parte del nome dell'endpoint. I nomi degli endpoint dual-stack utilizzano la seguente

convenzione di denominazione: `sns.Region.amazonaws.com` Ad esempio, il nome dell'endpoint dual-stack per la Regione eu-west-1 è `sns.eu-west-1.amazonaws.com`.

[Per l'elenco completo degli endpoint Amazon SNS, consulta la AWS Guida generale.](#)

Miglioramento della sicurezza di Amazon SNS con Message Data Protection

Important

La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta [Amazon SNS Message Data Protection Availability Change](#).

- Rispetto a quanto avviene per i dati a riposo, la [protezione dei dati dei messaggi](#) è una caratteristica di Amazon SNS utilizzata per definire regole e policy personalizzate per l'audit e il controllo del contenuto dei dati in transito.
- La protezione dei dati dei messaggi fornisce servizi di governance, conformità e audit per le applicazioni aziendali basate su messaggi, in modo che l'ingresso e l'uscita dei dati possano essere controllati dal proprietario dell'argomento di Amazon SNS e i flussi di contenuti possano essere tracciati e registrati.
- Puoi scrivere regole di governance basate sul payload per impedire l'ingresso dei contenuti non autorizzati del payload nei flussi di messaggi.
- Puoi concedere diverse autorizzazioni di accesso ai contenuti ai singoli iscritti e controllare l'intero processo del flusso dei contenuti.

Identity and Access Management in Amazon SNS

L'accesso 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 argomenti e messaggi di Amazon SNS. Le sezioni che seguono forniscono informazioni su come utilizzare [AWS Identity and Access Management \(IAM\)](#) e Amazon SNS per proteggere le proprie risorse attraverso il controllo degli accessi.

AWS Identity and Access Management (IAM) è un dispositivo Servizio AWS che aiuta un amministratore a controllare in modo sicuro l'accesso alle risorse. AWS Gli amministratori IAM controllano chi può essere autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) per utilizzare le risorse Amazon SNS. IAM è un software Servizio AWS che puoi utilizzare senza costi aggiuntivi.

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia in base al tuo ruolo:

- Utente del servizio: richiedi le autorizzazioni all'amministratore se non riesci ad accedere alle funzionalità (consulta [Risoluzione dei problemi di identità e accesso ad Amazon Simple Notification Service](#))
- Amministratore del servizio: determina l'accesso degli utenti e invia le richieste di autorizzazione (consulta [Come funziona Amazon SNS con IAM](#))
- Amministratore IAM: scrivi policy per gestire l'accesso (consulta [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. Devi autenticarti come utente Account AWS root, utente IAM o assumendo un ruolo IAM.

Puoi accedere come identità federata utilizzando credenziali provenienti da una fonte di identità come AWS IAM Identity Center (IAM Identity Center), autenticazione Single Sign-On o credenziali. Google/Facebook Per ulteriori informazioni sull'accesso, consulta [Come accedere all' Account AWS](#) nella Guida per l'utente di Accedi ad AWS .

Per l'accesso programmatico, AWS fornisce un SDK e una CLI per firmare crittograficamente le richieste. Per ulteriori informazioni, consulta [AWS Signature Version 4 per le richieste API](#) nella Guida per l'utente di IAM.

Account AWS utente root

Quando si crea un Account AWS, si inizia con un'identità di accesso denominata utente Account AWS root che ha accesso completo a tutte Servizi AWS le risorse. Consigliamo vivamente di non utilizzare l'utente root per le attività quotidiane. Per le attività che richiedono le credenziali dell'utente root, consulta [Attività che richiedono le credenziali dell'utente root](#) nella Guida per l'utente IAM.

Identità federata

Come procedura ottimale, richiedi agli utenti umani di utilizzare la federazione con un provider di identità per accedere Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory aziendale, del provider di identità Web o Directory Service che accede Servizi AWS utilizzando le credenziali di una fonte di identità. Le identità federate assumono ruoli che forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, si consiglia di utilizzare AWS IAM Identity Center. Per ulteriori informazioni, consulta [Che cos'è il Centro identità IAM?](#) nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un [utente IAM](#) è una identità che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ti consigliamo di utilizzare credenziali temporanee invece di utenti IAM con credenziali a lungo termine. Per ulteriori informazioni, consulta [Richiedere agli utenti umani di utilizzare la federazione con un provider di identità per accedere AWS utilizzando credenziali temporanee](#) nella Guida per l'utente IAM.

Un [gruppo IAM](#) specifica una raccolta di utenti IAM e semplifica la gestione delle autorizzazioni per gestire gruppi di utenti di grandi dimensioni. Per ulteriori informazioni, consulta [Casi d'uso per utenti IAM](#) nella Guida per l'utente di IAM.

Ruoli IAM

Un [ruolo IAM](#) è un'identità con autorizzazioni specifiche che fornisce credenziali temporanee. Puoi assumere un ruolo [passando da un ruolo utente a un ruolo IAM \(console\)](#) o chiamando un'operazione AWS CLI o AWS API. Per ulteriori informazioni, consulta [Metodi per assumere un ruolo](#) nella Guida per l'utente di IAM.

I ruoli IAM sono utili per l'accesso degli utenti federati, le autorizzazioni utente IAM temporanee, l'accesso multi-account, l'accesso multi-servizio e le applicazioni in esecuzione su Amazon EC2. Per maggiori informazioni, consultare [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Gestione dell'accesso tramite policy

Puoi controllare l'accesso AWS creando policy e associandole a AWS identità o risorse. Una policy definisce le autorizzazioni quando è associata a un'identità o a una risorsa. AWS valuta queste

politiche quando un preside effettua una richiesta. La maggior parte delle politiche viene archiviata AWS come documenti JSON. Per maggiori informazioni sui documenti delle policy JSON, consulta [Panoramica delle policy JSON](#) nella Guida per l'utente IAM.

Utilizzando le policy, gli amministratori specificano chi ha accesso a cosa definendo quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Un amministratore IAM crea le policy IAM e le aggiunge ai ruoli, che gli utenti possono quindi assumere. Le policy IAM definiscono le autorizzazioni indipendentemente dal metodo utilizzato per eseguirle.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile collegare a un'identità (utente, gruppo o ruolo). Tali policy controllano le operazioni autorizzate per l'identità, nonché le risorse e le condizioni in cui possono essere eseguite. Per informazioni su come creare una policy basata su identità, consultare [Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente](#) nella Guida per l'utente IAM.

Le policy basate su identità possono essere policy in linea (con embedding direttamente in una singola identità) o policy gestite (policy autonome collegate a più identità). Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta [Scegliere tra policy gestite e policy in linea](#) nella Guida per l'utente di IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Gli esempi includono le policy di trust dei ruoli IAM 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. In una policy basata sulle risorse è obbligatorio [specificare un'entità principale](#).

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite di IAM in una policy basata sulle risorse.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in AWS Organizations. Per ulteriori informazioni, consultare [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations.
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta [Politiche di controllo delle risorse \(RCPs\)](#) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare [Policy di sessione](#) nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta [Logica di valutazione delle policy](#) nella IAM User Guide.

Controllo accessi

Amazon SNS dispone di un proprio sistema di autorizzazioni basato sulle risorse che utilizza policy scritte nello stesso linguaggio utilizzato per le policy (IAM). AWS Identity and Access Management. Questo significa che puoi ottenere risultati simili con le policy Amazon SNS e IAM.

Note

È importante capire che tutti 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 ulteriori informazioni sull'utilizzo dell'accesso multi-account, consultare la sezione relativa all'[abilitazione dell'accesso multiaccount](#) nella Guida dell'utente IAM.

Casi d'uso del controllo degli accessi di Amazon SNS

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](#).
- Limitare le sottoscrizioni all'argomento solo per il protocollo HTTPS. Per ulteriori informazioni, consulta [Limitazione delle sottoscrizioni a HTTPS](#).
- Permettere ad Amazon SNS di pubblicare i messaggi nella coda Amazon SQS. Per ulteriori informazioni, consulta [Pubblicare messaggi in una coda Amazon SQS..](#)

Concetti chiave della politica di accesso di Amazon SNS

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.

Autorizzazione

L'autorizzazione è il concetto che permette o rifiuta un tipo di accesso a una particolare risorsa. Le autorizzazioni seguono essenzialmente questo modulo: "A ha/non ha l'autorizzazione di eseguire B in C se si applica D". Ad esempio, Jane (A) ha l'autorizzazione a pubblicare (B) in TopicA (C) se usa il protocollo HTTP (D). Ogni volta che Jane pubblica in TopicA, il servizio verifica se ha l'autorizzazione e se la richiesta soddisfa le condizioni stabilite nell'autorizzazione.

Dichiarazione

Una dichiarazione è la descrizione formale di una singola autorizzazione, scritta nel linguaggio della policy di accesso. Scrivi sempre una dichiarazione nell'ambito di un documento container più ampio denominato policy (vedi il prossimo concetto).

Policy

Una policy è un documento (scritto nella sintassi della/e policy di accesso) che funge da container per una o più dichiarazioni. Ad esempio, una policy potrebbe avere due dichiarazioni: una che afferma che Jane può iscriversi utilizzando il protocollo e-mail e un'altra che afferma che Bob non può pubblicare in Topic A. Come mostrato nella figura seguente, uno scenario equivalente potrebbe

avere due policy, una che afferma che Jane può iscriversi utilizzando il protocollo e-mail e un altro che afferma che Bob non può pubblicare su Topic A.



Solo i caratteri ASCII sono ammessi nei documenti delle policy. È possibile utilizzare `aws:SourceAccount` e `aws:SourceOwner` aggirare lo scenario in cui è necessario collegare altri AWS servizi ARNs che contengono caratteri non ASCII. vedi la differenza tra [aws:SourceAccount rispetto a aws:SourceOwner](#).

Emittente

L'emittente è la persona che scrive una policy per concedere le autorizzazioni per una risorsa. L'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 particolare intervallo CIDR)

La chiave è la caratteristica specifica che costituisce la base per la limitazione degli accessi. Ad esempio, la data e l'ora della richiesta.

Usa insieme condizioni e chiavi per esprimere la limitazione. Il modo più semplice per capire come si implementa effettivamente una limitazione è con un esempio: se vuoi limitare l'accesso a prima del 30 maggio 2010, utilizzi la condizione chiamata `DateLessThan`. Utilizzare la chiave chiamata `aws:CurrentTime` e impostarla sul valore `2010-05-30T00:00:00Z`. AWS definisce le condizioni e le chiavi che è possibile utilizzare. Il AWS servizio stesso (ad esempio, Amazon SQS o Amazon SNS) potrebbe anche definire chiavi specifiche del servizio. Per ulteriori informazioni, consulta [Autorizzazioni API Amazon SNS: riferimento a operazioni e risorse](#).

Richiedente

Il richiedente è la persona che invia una richiesta a un AWS servizio e chiede l'accesso a una particolare risorsa. Il richiedente invia una richiesta AWS che dice essenzialmente: «Mi permetterai di fare da B a C dove 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

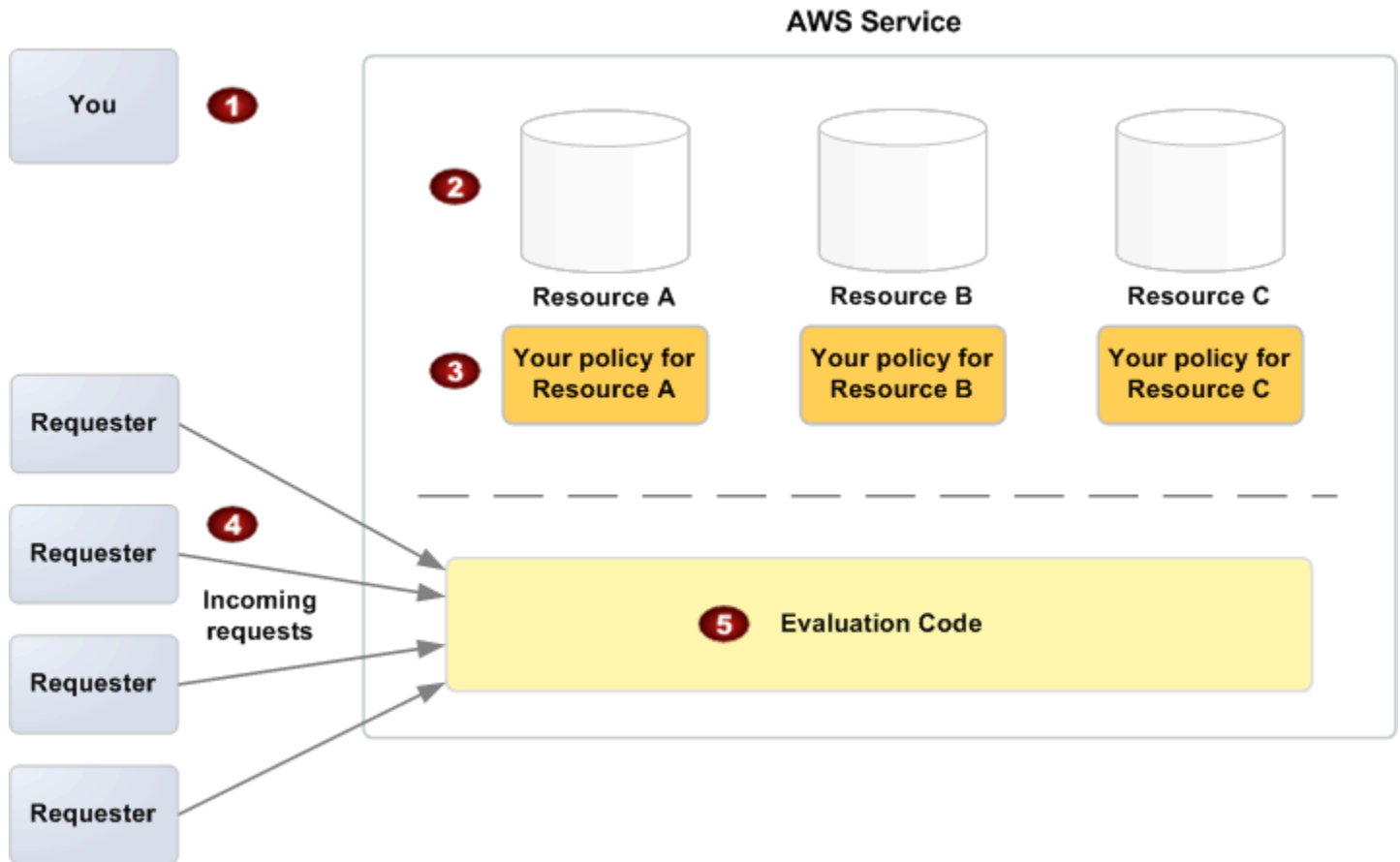
Un consenso viene restituito da una dichiarazione che ha effect=allow, presupponendo che tutte le condizioni dichiarate siano soddisfatte. Esempio: consenti le richieste se vengono ricevute prima delle 13:00 del 30 aprile 2010. Un consenso sovrascrive tutti i rifiuti per default, ma mai un rifiuto esplicito.

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 accessi di Amazon SNS

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 code Amazon SQS).

3 Le tue policy.

In genere hai una policy per risorsa, ma possono essere di più. Il AWS servizio stesso fornisce un'API che utilizzi per caricare e gestire le tue politiche.

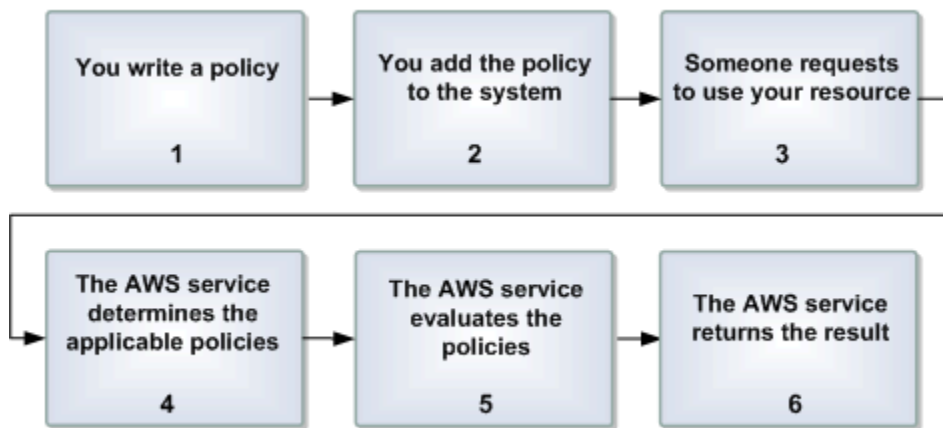
4 Richiedenti e relative richieste in arrivo al AWS servizio.

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 argomenti Amazon SNS.

- 2 Carichi la tua politica su. AWS

Il AWS servizio stesso fornisce un'API che utilizzi per caricare le tue politiche. Ad esempio, usare l'operazione di Amazon SNS `SetTopicAttributes` per caricare una policy per un particolare argomento Amazon SNS.

- 3 Qualcuno invia una richiesta per usare la tua risorsa.

Ad esempio, un utente invia una richiesta ad Amazon SNS; per usare uno dei tuoi argomenti.

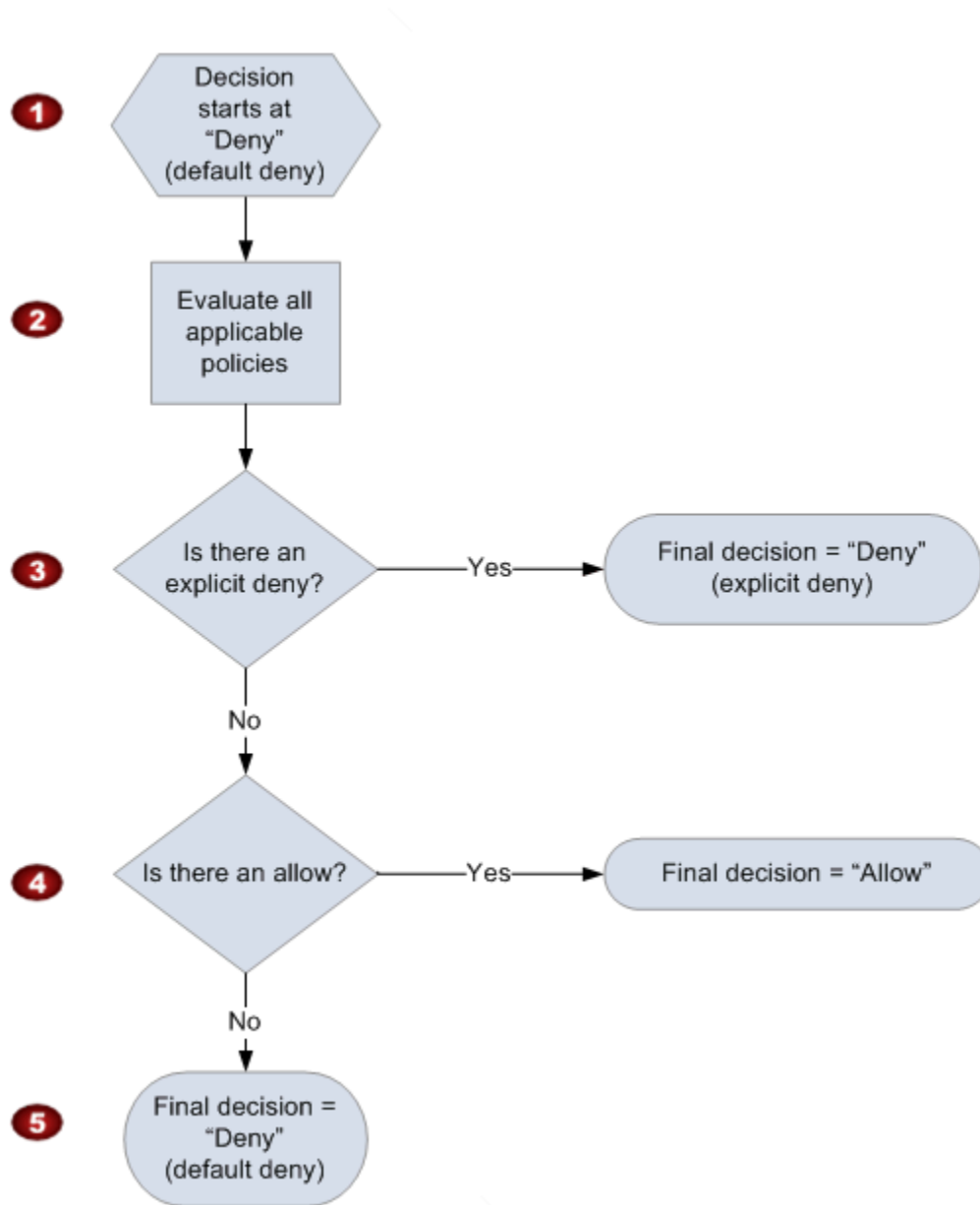
4	Il AWS servizio determina quali politiche sono applicabili alla richiesta. Ad esempio, Amazon SNS guarda tutte le policy Amazon SNS disponibili e determina quali sono applicabili (in base a cosa è la risorsa, chi è il richiedente e così via).
5	Il AWS servizio valuta le politiche. Ad esempio, Amazon SNS valuta le policy e determina se il richiedente è autorizzato o meno ad utilizzare il tuo argomento. Per ulteriori informazioni sulla logica della decisione, consulta Logica della valutazione.
6	Il 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.

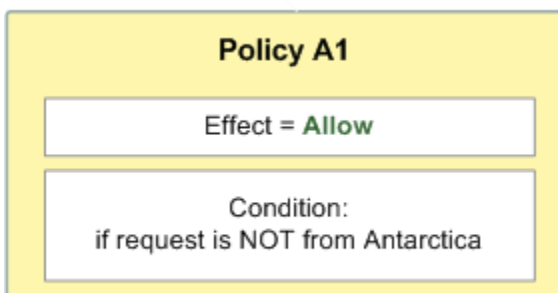
Se ne trova anche uno, il codice di applicazione restituisce una decisione di "consenso" e il processo è completato (il servizio continua a elaborare la richiesta).
- 5 Se non viene trovato alcun consenso, la decisione finale è il "rifiuto" e dal momento che non è stato trovato un rifiuto esplicito o 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 Amazon SNS, ma la policy sull'argomento non si riferisce Account AWS affatto a quella dell'utente, tale policy genera una negazione predefinita.

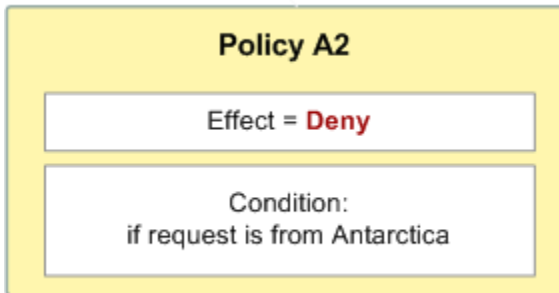
Una policy restituisce un rifiuto per default anche quando una condizione in una dichiarazione non viene soddisfatta. Se tutte le condizioni nella dichiarazione sono soddisfatte, la policy restituisce un consenso o un rifiuto esplicito, a seconda del valore dell'elemento Effetto nella policy. Le policy non specificano cosa fare se una condizione non viene soddisfatta e quindi il risultato di default in quel caso è un rifiuto per default.

Ad esempio, supponiamo che tu voglia impedire le richieste provenienti dall'Antartide. Scrivi una policy (chiamata Policy A1) che consente una richiesta solo se non proviene dall'Antartide. Il diagramma seguente illustra la policy.



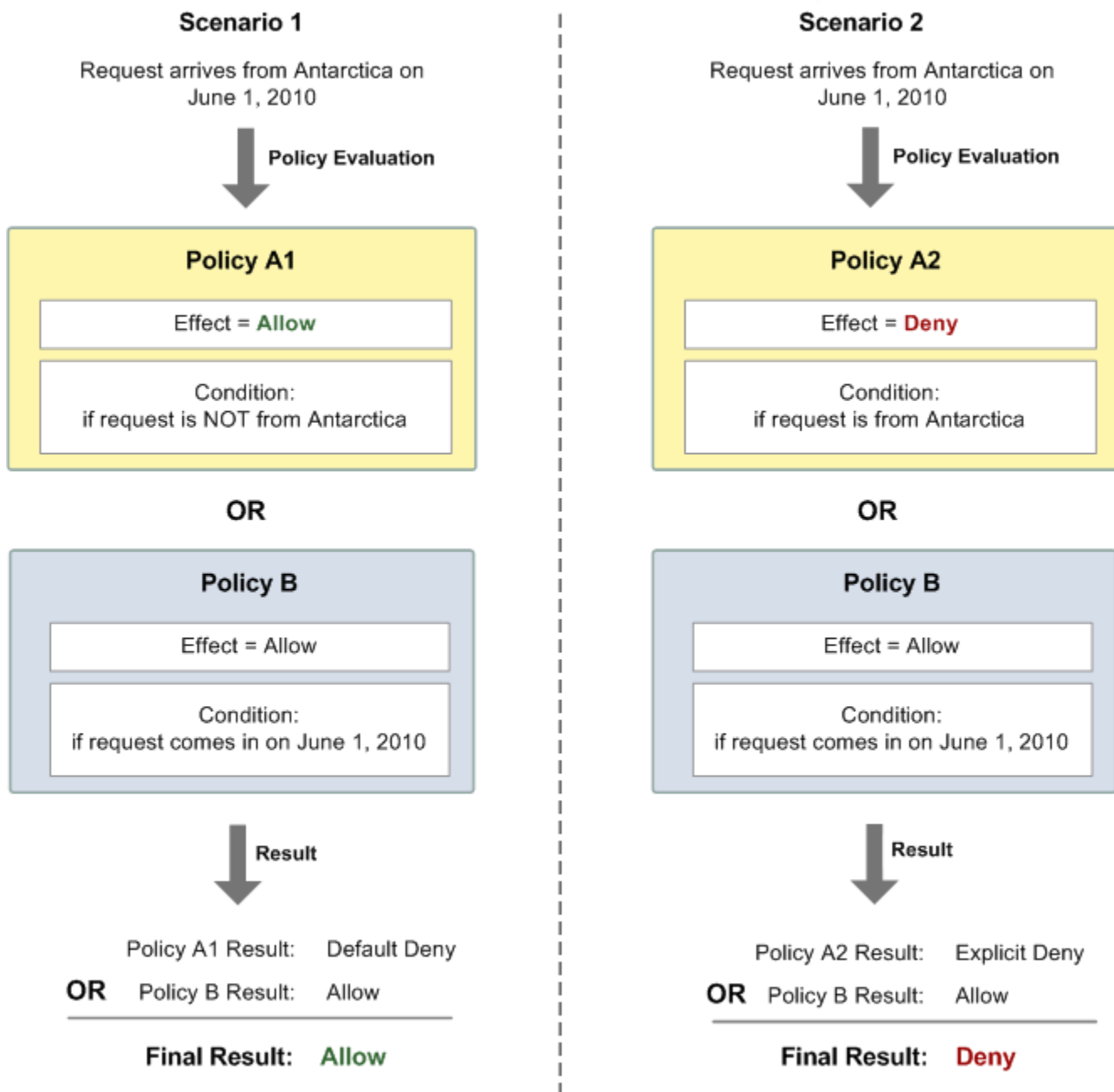
Se qualcuno invia una richiesta dagli Stati Uniti, la condizione è soddisfatta (la richiesta non proviene dall'Antartide). Pertanto, la richiesta è consentita. Tuttavia, se qualcuno invia una richiesta dall'Antartide, la condizione non viene soddisfatta e il risultato della policy è quindi un rifiuto per default.

Puoi trasformare il risultato in un rifiuto esplicito riscrivendo la policy (denominata Policy A2) come nel diagramma seguente. In questo caso, la policy rifiuta esplicitamente una richiesta se proviene dall'Antartide.



Se qualcuno invia una richiesta dall'Antartide, la condizione viene soddisfatta e il risultato della policy è quindi un rifiuto esplicito.

La differenza tra un rifiuto per default e un rifiuto esplicito è importante perché un rifiuto per default può essere sovrascritto da un consenso, mentre un rifiuto esplicito no. Ad esempio, supponiamo che ci sia un'altra policy che consente le richieste se arrivano il 1° giugno 2010. In che modo questa policy influisce sul risultato complessivo se abbinata alla policy che limita l'accesso dall'Antartide? Confronteremo il risultato complessivo quando abbiniamo la policy basata sulla data (che chiameremo policy B) con le precedenti policy A1 e A2. Lo scenario 1 abbina la policy A1 con la policy B e lo scenario 2 abbina la policy A2 con la policy B. La seguente figura e la discussione mostrano i risultati quando arriva una richiesta dall'Antartide il 1 giugno 2010.



Nello scenario 1, la policy A1 restituisce un rifiuto per default, come descritto in precedenza in questa sezione. La policy B restituisce un consenso perché la policy (per definizione) consente le richieste che arrivano il 1 giugno 2010. Il consenso dalla policy B sovrascrive il rifiuto per default dalla policy A1 e pertanto la richiesta è consentita.

Nello scenario 2, la policy A2 restituisce un rifiuto esplicito, come descritto in precedenza in questa sezione. Anche in questo caso, la policy B restituisce un consenso. Il rifiuto esplicito dalla policy A2 sovrascrive il consenso dalla policy B e pertanto la richiesta viene rifiutata.

Esempi di casi per il controllo degli accessi Amazon SNS

Questa sezione include alcuni esempi di casi d'uso tipici per il controllo accessi.

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'azione dell'API Amazon SNS. `AddPermission`

L'`AddPermission` consente di specificare un argomento, un elenco di Account AWS IDs, un elenco di azioni e un'etichetta. Amazon SNS genera quindi e aggiunge automaticamente una nuova dichiarazione di policy alla policy di controllo degli accessi dell'argomento. Non è necessario che tu scriva tu stesso la dichiarazione sulla policy: Amazon SNS la gestisce 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`, Amazon SNS genererà e inserirà la seguente dichiarazione di 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, `1111-2222-3333` avrà l'autorizzazione a pubblicare messaggi sull'argomento. Account AWS

Informazioni aggiuntive:

- Gestione personalizzata delle policy: sebbene `AddPermission` sia utile per concedere le autorizzazioni, spesso è utile gestire manualmente la policy di controllo degli accessi

dell'argomento per scenari più complessi, come l'aggiunta di condizioni o la concessione di autorizzazioni a ruoli o servizi IAM specifici. Puoi farlo utilizzando l'`SetTopicAttributesAPI` per aggiornare direttamente l'attributo `policy`.

- **Best practice di sicurezza:** fai attenzione quando concedi le autorizzazioni per assicurarti che solo le entità attendibili Account AWS abbiano accesso ai tuoi argomenti di Amazon SNS. Esamina e verifica regolarmente le politiche allegate ai tuoi argomenti per mantenere la sicurezza.
- **Limiti delle policy:** tieni presente che esistono limiti alla dimensione e alla complessità delle policy di Amazon SNS. Se devi aggiungere molte autorizzazioni o condizioni complesse, assicurati che la tua politica rimanga entro questi limiti.

Limitazione delle sottoscrizioni a HTTPS

Per limitare il protocollo di invio delle notifiche per il tuo argomento Amazon SNS a HTTPS, devi creare una policy personalizzata. L'`AddPermission` in Amazon SNS non consente di specificare restrizioni di protocollo quando si concede l'accesso al proprio argomento. Pertanto, devi scrivere manualmente una politica che applichi questa restrizione e quindi utilizzare l'`SetTopicAttributes` 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 si desidera concedere l'accesso e imporre la condizione che siano consentiti solo gli abbonamenti HTTPS. 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 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. Utilizza l'`SetTopicAttributes` azione nell'API Amazon SNS per applicare questa politica al tuo argomento. Imposta l'`Policy` attributo dell'argomento sulla policy JSON 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. La limitazione degli abbonamenti a HTTPS migliora la sicurezza delle notifiche garantendo 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.

Pubblicare messaggi in una coda Amazon SQS.

Per pubblicare messaggi dal tuo argomento Amazon SNS in una coda Amazon SQS, devi configurare le autorizzazioni corrette sulla coda Amazon SQS. Sebbene sia Amazon SNS che Amazon SQS AWS utilizzino il linguaggio delle policy di controllo degli accessi, è necessario impostare esplicitamente una policy sulla coda di Amazon SQS per consentire l'invio di messaggi dall'argomento Amazon SNS.

Puoi raggiungere questo obiettivo utilizzando l'`SetQueueAttributes` azione per applicare una policy personalizzata alla coda di Amazon SQS. A differenza di Amazon SNS, Amazon SQS

non supporta `AddPermission` l'azione per la creazione di dichiarazioni politiche con condizioni. Pertanto, è necessario scrivere la policy manualmente.

Di seguito è riportato un esempio di policy di Amazon SQS che concede ad Amazon SNS l'autorizzazione a inviare messaggi alla tua coda. Tieni presente che questa policy è associata alla coda Amazon SQS, non all'argomento Amazon SNS. Le azioni specificate sono azioni Amazon SQS e la risorsa è l'Amazon Resource Name (ARN) della coda. È possibile recuperare l'ARN della coda 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"
      }
    }
  }]
}
```

Questo criterio utilizza la `aws:SourceArn` condizione per limitare l'accesso alla coda SQS in base all'origine dei messaggi inviati. Ciò garantisce che solo i messaggi provenienti dall'argomento SNS specificato (in questo caso, `arn:aws:sns:us-east-2:444455556666:`) possano essere recapitati alla coda. `MyTopic`

Informazioni aggiuntive:

- Coda ARN. Assicurati di recuperare l'ARN corretto della tua coda Amazon SQS 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'argomento Amazon SNS per interagire con la coda di Amazon SQS e rivedi regolarmente le tue politiche per assicurarti che siano sicure up-to-date

- Politiche predefinite in Amazon SNS. Amazon SNS non concede automaticamente una policy predefinita che consenta ad altri account Servizi AWS di accedere agli argomenti appena creati. Per impostazione predefinita, gli argomenti di Amazon SNS vengono creati senza autorizzazioni, il che significa che sono privati e accessibili solo all'account che li ha creati. Per consentire l'accesso ad altri Servizi AWS account o responsabili, devi definire e allegare esplicitamente una politica di accesso all'argomento. Ciò è in linea con il principio del privilegio minimo, che garantisce che per impostazione predefinita non venga concesso alcun accesso involontario.
- Test e convalida. Dopo aver impostato la policy, verifica l'integrazione pubblicando messaggi sull'argomento Amazon SNS e verificando che vengano recapitati correttamente alla coda di Amazon SQS. Questo aiuta a confermare che la policy sia 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 argomento Amazon SNS, 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 relativo ad Amazon SNS.

Ecco come puoi configurarlo:

1. Scrivi la politica. La policy dovrebbe concedere al servizio Amazon S3 (s3.amazonaws.com) le autorizzazioni necessarie per la pubblicazione sul tuo argomento Amazon SNS. 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 Amazon SNS.
 - 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. Usa l'`SetTopicAttributes` operazione per impostare questa politica sul tuo argomento Amazon SNS. 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 argomento Amazon SNS in base all'account di origine.
 - Gateway Amazon API
 - Amazon CloudWatch
 - Amazon DevOps Guru
 - Amazon EventBridge
 - GameLift Server Amazon
 - Servizio di SMS e messaggi vocali Amazon Pinpoint
 - Amazon RDS

- Amazon Redshift
 - Amazon Glacier
 - Amazon SES
 - Amazon Simple Storage Service
 - AWS CodeCommit
 - Directory Service
 - AWS Lambda
 - Strumento di gestione degli incidenti AWS Systems Manager
- Test e convalida. Dopo aver applicato la policy, verifica la configurazione attivando un evento nel bucket Amazon S3 e confermando che la pubblicazione è avvenuta 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 policy tematiche di Amazon SNS 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 ad Amazon SES di pubblicare in 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 di aver effettuato l'accesso dall'account 111122223333, aperto Amazon SES e creato un'e-mail. Per pubblicare notifiche relative a questa e-mail in un argomento Amazon SNS proprietario dell'account 444455556666, bisogna creare una policy come la seguente. A tale scopo, è necessario fornire informazioni sull'entità principale (l'altro servizio) e sulla proprietà di ciascuna risorsa. La dichiarazione `Resource` fornisce l'argomento ARN, che include l'account ID del proprietario dell'argomento, 444455556666. La dichiarazione `"aws:SourceOwner": "111122223333"` specifica che il tuo account è proprietario dell'e-mail.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
```

```
"Effect": "Allow",
"Principal": {
  "Service": "ses.amazonaws.com"
},
"Action": "sns:Publish",
"Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
"Condition": {
  "StringEquals": {
    "aws:SourceOwner": "111122223333"
  }
}
}
```

Quando si pubblicano eventi su Amazon SNS, i seguenti servizi supportano `aws:SourceOwner`:

- Gateway Amazon API
- Amazon CloudWatch
- Amazon DevOps Guru
- GameLift Server Amazon
- Servizio di SMS e messaggi vocali Amazon Pinpoint
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- Directory Service
- AWS Lambda
- Strumento di gestione degli incidenti AWS Systems Manager

`aws:SourceAccount` rispetto a `aws:SourceOwner`

Important

`aws:SourceOwner` è obsoleto e i nuovi servizi possono integrarsi con Amazon SNS solo tramite `aws:SourceArn` e `aws:SourceAccount`. Amazon SNS mantiene ancora la

compatibilità con le versioni precedenti per i servizi esistenti che supportano attualmente `aws:SourceOwner`.

Le chiavi di condizione `aws:SourceAccount` e `aws:SourceOwner` sono impostate ciascuna da alcuni Servizi AWS quando pubblicano in un argomento Amazon SNS. Se supportato, il valore 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 ad Amazon SES di pubblicare in un argomento di proprietà di un altro account](#) come utilizza Amazon SES `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, controllare l'accesso e la sicurezza e condividere le risorse tra le tue sedi. 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 SNS l'argomento `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

Utilizza i seguenti passaggi per richiamare un argomento di Amazon SNS con CloudWatch un allarme tra diversi. Account AWS Questo esempio utilizza due account:

- L'account A viene utilizzato per creare l' CloudWatch allarme.
- L'account B viene utilizzato per creare un argomento SNS.

Crea un argomento SNS nell'account B

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione scegliere Argomenti, quindi Crea nuovo argomento.
3. Scegli Standard per il tipo di argomento, quindi crea un nome per l'argomento.
4. Scegli Crea argomento, quindi copia l'ARN dell'argomento.
5. Nel riquadro di navigazione scegliere Subscriptions (Sottoscrizioni), quindi selezionare Create subscription (Crea sottoscrizione).
6. Aggiungi l'ARN dell'argomento nella sezione Arn dell'argomento, scegli Email come protocollo, quindi inserisci un indirizzo e-mail.
7. Scegli Crea abbonamento, quindi controlla la tua e-mail per confermare l'iscrizione.

Crea un CloudWatch allarme nell'account A

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nel riquadro di navigazione, scegli Allarmi, quindi scegli Crea avvisi.
3. Se non hai ancora creato un allarme, creane uno adesso. Altrimenti, seleziona la metrica e fornisci i dettagli per la soglia e i parametri di confronto.

4. Da Configura azioni, in Notifiche, scegli Usa l'argomento ARN per notificare altri account, quindi inserisci l'argomento ARN dall'account B.
5. Crea un nome per l'avviso, quindi scegli Crea allarme.

Aggiorna la politica di accesso dell'argomento SNS nell'account B

1. Accedi alla [console Amazon SNS](#).
2. Nel riquadro di navigazione, scegli Argomenti, quindi seleziona l'argomento.
3. Scegli Modifica, quindi aggiungi quanto segue alla politica:

Note

Sostituisci i valori di esempio nella politica seguente con i tuoi.

JSON

```
{
  "Version": "2012-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SNS:GetTopicAttributes",
        "SNS:SetTopicAttributes",
        "SNS:AddPermission",
        "SNS:RemovePermission",
        "SNS:DeleteTopic",
        "SNS:Subscribe",
        "SNS:ListSubscriptionsByTopic",
        "SNS:Publish"
      ],
      "Resource": "arn:aws:cloudwatch:us-west-1:111122223333:alarm:",
      "Condition": {
```

```

        "ArnLike": {
            "aws:SourceArn": "arn:aws:cloudwatch:us-
west-1:111122223333:alarm:"
        }
    }
}
]
}

```

Prova l'allarme

Per testare l'allarme, modifica la soglia di allarme in base ai punti dati metrici o modifica manualmente lo stato dell'allarme. Quando si modifica la soglia o lo stato dell'allarme, si riceve una notifica via e-mail.

Soluzione alternativa per l'utilizzo di un argomento Amazon SNS locale e l'inoltro di messaggi

Utilizza i seguenti passaggi per abilitare le notifiche CloudWatch Amazon SNS for Alarms su più account:

1. Crea un [argomento Amazon SNS](#) nello stesso account dell'CloudWatchallarme (111122223333).
2. Sottoscrivi una [funzione Lambda](#) o una [EventBridge regola Amazon](#) a quell'argomento di Amazon SNS.
3. La funzione o la EventBridge regola Lambda può quindi pubblicare il messaggio sull'argomento Amazon SNS nell'account di destinazione (444455556666).

Limitare la pubblicazione a un argomento Amazon SNS solo da un endpoint VPC specifico

In questo caso, l'argomento nell'account 444455556666 può pubblicare solo dall'endpoint VPC con l'ID `vpce-1ab2c34d`.

```

{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {

```

```
"StringNotEquals": {  
  "aws:sourceVpce": "vpce-1ab2c34d"  
}  
}  
}]  
}
```

Come funziona Amazon SNS con IAM

Prima di utilizzare IAM per gestire l'accesso ad Amazon SNS, scopri quali funzioni IAM sono disponibili per l'uso con Amazon SNS.

Funzionalità di IAM utilizzabili con Amazon Simple Notification Service

Funzionalità IAM	Supporto per Amazon SNS
Policy basate sull'identità	Sì
Policy basate su risorse	Sì
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
ACLs	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	Sì
Ruoli collegati al servizio	No

Per avere una visione di alto livello di come Amazon SNS e AWS altri servizi funzionano con la maggior parte delle funzionalità IAM, [AWS consulta i servizi che funzionano con](#) IAM nella IAM User Guide.

AWS politiche gestite per Amazon Simple Notification Service

Una politica AWS gestita è una politica autonoma creata e amministrata da 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 Si consiglia pertanto di ridurre ulteriormente le autorizzazioni definendo [policy gestite dal cliente](#) specifiche per i propri casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle politiche gestite. 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 nel Servizio AWS viene lanciata una nuova o quando diventano disponibili nuove operazioni API per i servizi esistenti.

Per ulteriori informazioni, consultare [Policy gestite da AWS](#) nella Guida per l'utente di IAM.

AWS politica gestita: Amazon SNSFull Access

AmazonSNSFullAccess fornisce l'accesso completo ad Amazon SNS utilizzando la console di gestione AWS. Questa policy include anche le seguenti azioni di lettura e scrittura per AWS End User Messaging SMS le chiamate tramite Amazon SNS. Puoi allegare questa politica ai tuoi utenti, gruppi o ruoli.

Dettagli delle autorizzazioni

Le seguenti autorizzazioni si applicano solo quando si utilizza Amazon APIs SNS:

- `sns : *` — Concede le autorizzazioni complete per eseguire qualsiasi azione relativa ad Amazon SNS. Questo carattere jolly (*) indica che l'utente può eseguire tutte le azioni possibili di Amazon SNS.

- **sms-voice:DescribeVerifiedDestinationNumbers**— Consente di recuperare un elenco di numeri di telefono che sono stati verificati per l'invio di messaggi SMS all'interno di Account AWS
- **sms-voice:CreateVerifiedDestinationNumber**— Consente di verificare un nuovo numero di telefono da utilizzare con i servizi di messaggistica SMS di AWS
- **sms-voice:SendDestinationNumberVerificationCode**— Consente di inviare un codice di verifica a un numero di telefono in fase di verifica per la messaggistica SMS all'interno AWS.
- **sms-voice:SendTextMessage**— Consente di creare un nuovo messaggio di testo e inviarlo al numero di telefono del destinatario. **SendTextMessage** invia un messaggio SMS 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 messaggistica SMS di AWS
- **sms-voice:DescribeAccountAttributes**— Consente di recuperare informazioni dettagliate sugli attributi a livello di account relativi ai servizi di messaggistica SMS all'interno AWS
- **sms-voice:DescribeSpendLimits**— Consente di recuperare informazioni sui limiti di spesa associati ai servizi di messaggistica SMS all'interno di Account AWS
- **sms-voice:DescribePhoneNumbers**— Consente di recuperare informazioni dettagliate sui numeri di telefono associati ai servizi di messaggistica SMS all'interno di Account AWS
- **sms-voice:SetTextMessageSpendLimitOverride**— Consente di impostare o annullare il limite di spesa per i messaggi di testo SMS all'interno di Account AWS
- **sms-voice:DescribeOptedOutNumbers**— Consente di recuperare un elenco di numeri di telefono che hanno scelto di non ricevere SMS dal proprio account AWS
- **sms-voice>DeleteOptedOutNumber**— Consente di rimuovere un numero di telefono dall'elenco dei numeri esclusi all'interno del Account AWS

AmazonSNSFullAccessesempio di politica

JSON

```
{  
  "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, consulta [Amazon SNSFull Access](#) nel AWS Managed Policy Reference.

AWS politica gestita: Amazon SNSRead OnlyAccess

AmazonSNSReadOnlyAccessfornisce accesso in sola lettura ad Amazon SNS utilizzando. Console di gestione AWS Questa policy include anche le seguenti azioni di sola lettura per le chiamate tramite AWS End User Messaging SMS Amazon SNS. Puoi allegare questa policy ai tuoi utenti, gruppi e ruoli.

Dettagli delle autorizzazioni

Le seguenti autorizzazioni si applicano solo quando si utilizza Amazon APIs SNS:

- **sns:GetTopicAttributes**— Consente di recuperare gli attributi di un argomento Amazon SNS. Ciò include informazioni come l'ARN (Amazon Resource Name) dell'argomento, 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 le risorse Amazon SNS. Ciò include le autorizzazioni per elencare vari elementi relativi ad Amazon SNS, come:
 - **sns:ListTopics**— Consente di recuperare un elenco di tutti gli argomenti di Amazon SNS in Account AWS
 - **sns:ListSubscriptions**— Consente di recuperare un elenco di tutti gli abbonamenti agli argomenti di Amazon SNS.
 - **sns:ListSubscriptionsByTopic**— Consente di elencare tutti gli abbonamenti per un argomento specifico di Amazon SNS.
 - **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 messaggi SMS tramite Amazon SNS.
- **sns:GetEndpointAttributes**— Consente di recuperare gli attributi di un endpoint associato a un'applicazione della piattaforma Amazon SNS. 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 policy di protezione dei dati associata a un argomento di Amazon SNS.
- **sns:GetPlatformApplicationAttributes**— Consente di recuperare gli attributi di un'applicazione della piattaforma Amazon SNS. 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 impostazioni SMS predefinite per Account AWS

- `sns:GetSMSSandboxAccountStatus`— Consente di recuperare lo stato attuale della sandbox SMS per il tuo Account AWS
- `sns:GetSubscriptionAttributes`— Consente di recuperare gli attributi di un abbonamento specifico a un argomento di Amazon SNS.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Consente di visualizzare o recuperare un elenco di numeri di telefono che sono stati verificati per l'invio di messaggi SMS 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 SMS di AWS
- `sms-voice:DescribeSpendLimits`— Consente di visualizzare o recuperare informazioni sui limiti di spesa associati ai servizi di messaggistica SMS all'interno del Account AWS
- `sms-voice:DescribePhoneNumbers`— Consente di visualizzare o recuperare informazioni sui numeri di telefono utilizzati per i servizi di messaggistica SMS 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 messaggi SMS dal Account AWS

AmazonSNSReadOnlyAccessesempio di politica

JSON

```
{
  "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, consulta [Amazon SNSFull Access](#) nel AWS Managed Policy Reference.

Amazon SNS aggiorna le policy gestite AWS

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 feed RSS nella pagina della cronologia dei documenti di Amazon SNS.

Modifica	Descrizione	Data
AmazonSNSFullAccess : aggiornamento a una policy esistente	Amazon SNS ha aggiunto nuove autorizzazioni per consentire l'accesso completo ad Amazon SNS utilizzando Console di gestione AWS	24/09/2024

Modifica	Descrizione	Data
Amazon SNSRead OnlyAccess : aggiornamento a una politica esistente	Amazon SNS ha aggiunto nuove autorizzazioni per consentire l'accesso in sola lettura ad Amazon SNS utilizzando. Console di gestione AWS	24/09/2024
Amazon SNS ha iniziato a tracciare le modifiche	Amazon SNS ha iniziato a tracciare le modifiche per le sue politiche AWS gestite.	27/08/2024

Operazioni delle policy per Amazon SNS

Supporta le operazioni di policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso in una policy. Includere le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di operazioni di Amazon SNS, consulta [Risorse definite da Amazon SNS](#) nella documentazione di riferimento all'autorizzazione del servizio.

Le operazioni delle policy in Amazon SNS utilizzano il seguente prefisso prima dell'operazione:

```
sns
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [  
    "sns:action1",  
    "sns:action2"  
]
```

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta [Esempi di policy basate su identità per Amazon Simple Notification Service](#).

Risorse delle policy per Amazon SNS

Supporta le risorse relative alle policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Come best practice, specifica una risorsa utilizzando il suo [nome della risorsa Amazon \(ARN\)](#). Per le azioni che non supportano le autorizzazioni a livello di risorsa, si utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"

```

Per visualizzare un elenco dei tipi di risorse Amazon SNS e relativi ARNs, consulta [Actions Defined by Amazon Simple Notification Service nel Service](#) Authorization Reference. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ciascuna risorsa, consulta [Risorse definite da Amazon SNS](#).

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta [Esempi di policy basate su identità per Amazon Simple Notification Service](#).

Chiavi di condizione delle policy per Amazon SNS

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale entità principale può eseguire operazioni su quali risorse e in quali condizioni.

L'elemento `Condition` specifica quando le istruzioni vengono eseguite in base a criteri definiti. È possibile 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. Per visualizzare tutte le chiavi di condizione AWS globali, consulta le chiavi di [contesto delle condizioni AWS globali nella Guida](#) per l'utente IAM.

Per un elenco completo delle chiavi di condizione di Amazon SNS, consulta [Chiavi di condizione per Amazon SNS](#) nella documentazione di riferimento all'autorizzazione di servizio. Per informazioni su

operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consulta [Risorse definite da Amazon SNS](#).

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta [Esempi di policy basate su identità per Amazon Simple Notification Service](#).

ACLs in Amazon SNS

Supporti ACLs: 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. ACLs sono simili alle politiche basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Amazon SNS

Supporta ABAC (tag nelle policy): parzialmente

Il controllo degli accessi basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi, chiamati tag. Puoi allegare tag a entità e AWS risorse IAM, quindi progettare politiche ABAC per consentire le operazioni quando il tag del principale corrisponde al tag sulla risorsa.

Per controllare l'accesso basato su tag, fornire 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 Sì. Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per maggiori informazioni su ABAC, consulta [Definizione delle autorizzazioni con autorizzazione ABAC](#) nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta [Utilizzo del controllo degli accessi basato su attributi \(ABAC\)](#) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Amazon SNS

Supporta le credenziali temporanee: sì

Le credenziali temporanee forniscono l'accesso a breve termine alle AWS risorse e vengono create automaticamente quando si utilizza la federazione o si cambia ruolo. AWS consiglia di generare

dinamicamente credenziali temporanee anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta [Credenziali di sicurezza temporanee in IAM](#) e [Servizi AWS compatibili con IAM](#) nella Guida per l'utente IAM.

Autorizzazioni dei principali tra servizi per Amazon SNS

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Le sessioni di accesso inoltrato (FAS) utilizzano le autorizzazioni del principale che chiama un Servizio AWS, combinate con la richiesta di effettuare richieste Servizio AWS ai servizi downstream. Per i dettagli delle policy relative alle richieste FAS, consulta [Forward access sessions](#).

Ruoli di servizio per Amazon SNS

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un [ruolo IAM](#) che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta [Create a role to delegate permissions to an Servizio AWS](#) nella Guida per l'utente IAM.

Warning

La modifica delle autorizzazioni per un ruolo di servizio potrebbe compromettere la funzionalità di Amazon SNS. Modifica i ruoli del servizio solo quando Amazon SNS fornisce le indicazioni per farlo.

Ruoli collegati ai servizi per Amazon SNS

Supporta i ruoli collegati ai servizi: no

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati al servizio vengono visualizzati nel tuo account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati al servizio, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta [Servizi AWS supportati da IAM](#). Trova un servizio nella tabella che include un Yes nella colonna Service-linked

ruolo (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate su identità per Amazon Simple Notification Service

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse di Amazon SNS. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta [Creazione di policy IAM \(console\)](#) nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Amazon SNS, 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.

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse Amazon SNS nell'account. Queste azioni possono comportare costi aggiuntivi per l' Account AWS. Quando si creano o modificano policy basate sull'identità, seguire 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 policy 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 maggiori informazioni, consulta [Policy gestite da AWS](#) o [Policy gestite da AWS per le funzioni dei processi](#) nella Guida per l'utente di IAM.
- Applicazione delle autorizzazioni con privilegio minimo - Quando si impostano le autorizzazioni con le policy IAM, concedere solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegio minimo. Per maggiori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta [Policy e autorizzazioni in IAM](#) nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso - Per limitare l'accesso ad azioni e risorse è possibile aggiungere una condizione alle policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate

utilizzando SSL. Puoi anche utilizzare le condizioni per concedere l'accesso alle azioni del servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio CloudFormation. Per maggiori informazioni, consultare la sezione [Elementi delle policy JSON di IAM: condizione](#) nella Guida per l'utente di IAM.

- Utilizzo dello strumento di analisi degli accessi IAM per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali - Lo strumento di analisi degli accessi IAM convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio (JSON) della policy IAM e alle best practice di IAM. Lo strumento di analisi degli accessi IAM offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per maggiori informazioni, consultare [Convalida delle policy per il Sistema di analisi degli accessi IAM](#) nella Guida per l'utente di IAM.
- Richiedi l'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o un utente root nel Account AWS tuo, attiva l'MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungere le condizioni MFA alle policy. Per maggiori informazioni, consultare [Protezione dell'accesso API con MFA](#) nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta [Best practice di sicurezza in IAM](#) nella Guida per l'utente di IAM.

Utilizzo della console Amazon SNS

Per accedere alla console Amazon SNS, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentirti di elencare e visualizzare i dettagli sulle risorse Amazon SNS 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 consentire autorizzazioni minime per la console per gli utenti che effettuano chiamate solo verso AWS CLI o l'API. AWS Al contrario, è opportuno concedere l'accesso solo alle azioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che utenti e ruoli possano continuare a utilizzare la console Amazon SNS, collega anche Amazon **ConsoleAccess** SNS **ReadOnly** AWS o la policy gestita alle entità. Per maggiori informazioni, consulta [Aggiunta di autorizzazioni a un utente](#) nella Guida per l'utente di IAM.

Altri tipi di policy

AWS supporta tipi di policy aggiuntivi che possono impostare le autorizzazioni massime concesse dai tipi di policy più comuni:

- Limiti delle autorizzazioni: imposta il numero massimo di autorizzazioni che una policy basata su identità ha la possibilità di concedere a un'entità IAM. Per ulteriori informazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente di IAM.
- Politiche di controllo del servizio (SCPs): specificano le autorizzazioni massime per un'organizzazione o un'unità organizzativa in AWS Organizations. Per ulteriori informazioni, consultare [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations.
- Politiche di controllo delle risorse (RCPs): imposta le autorizzazioni massime disponibili per le risorse nei tuoi account. Per ulteriori informazioni, consulta [Politiche di controllo delle risorse \(RCPs\)](#) nella Guida per l'AWS Organizations utente.
- Policy di sessione: policy avanzate passate come parametro quando si crea una sessione temporanea per un ruolo o un utente federato. Per maggiori informazioni, consultare [Policy di sessione](#) nella Guida per l'utente IAM.

Più tipi di policy

Quando a una richiesta si applicano più tipi di policy, le autorizzazioni risultanti sono più complicate da comprendere. Per scoprire come si AWS determina se consentire o meno una richiesta quando sono coinvolti più tipi di policy, consulta [Logica di valutazione delle policy](#) nella IAM User Guide.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa politica include le autorizzazioni per completare questa azione sulla console o utilizzando l'API o a livello di codice. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ]
    }
  ]
}
```



```
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
  },
  {
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
      "iam:GetGroupPolicy",
      "iam:GetPolicyVersion",
      "iam:GetPolicy",
      "iam:ListAttachedGroupPolicies",
      "iam:ListGroupPolicies",
      "iam:ListPolicyVersions",
      "iam:ListPolicies",
      "iam:ListUsers"
    ],
    "Resource": "*"
  }
]
```

Policy basate su identità per Amazon SNS

Supporta le policy basate sull'identità: sì

Le policy basate sull'identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente](#) nella Guida per l'utente di IAM.

Con le policy basate sull'identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta [Guida di riferimento agli elementi delle policy JSON IAM](#) nella Guida per l'utente IAM.

Esempi di policy basate su identità per Amazon SNS

Per visualizzare esempi di policy basate su identità Amazon SNS, consulta [Esempi di policy basate su identità per Amazon Simple Notification Service](#).

Policy basate su risorse all'interno di Amazon SNS

Supporta le policy basate su risorse

Sì

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy di bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. In una policy basata sulle risorse è obbligatorio [specificare un'entità principale](#). I principali possono includere account, utenti, ruoli, utenti federati o. Servizi AWS

Per consentire l'accesso multi-account, è possibile specificare un intero account o entità IAM in un altro account come entità principale in una policy basata sulle risorse. Per ulteriori informazioni, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Utilizzo di policy basate su identità con Amazon SNS

Amazon Simple Notification Service si integra con AWS Identity and Access Management (IAM) in modo da poter specificare quali azioni Amazon SNS un utente può eseguire con Account AWS le risorse Amazon SNS. Puoi specificare un particolare argomento nella policy. Ad esempio, puoi utilizzare delle variabili durante la creazione di una policy IAM che autorizza certi utenti della tua organizzazione a utilizzare l'operazione Publish con specifici argomenti nel tuo Account AWS. Per ulteriori informazioni, consulta la pagina sulle [Variabili delle policy](#) nella guida Uso di IAM.

Important

L'utilizzo di Amazon SNS con IAM; non comporta modifiche nell'utilizzo di Amazon SNS. Le operazioni di Amazon SNS esistenti rimangono immutate e non vengono aggiunte nuove operazioni di Amazon SNS relative agli utenti e al controllo accessi.

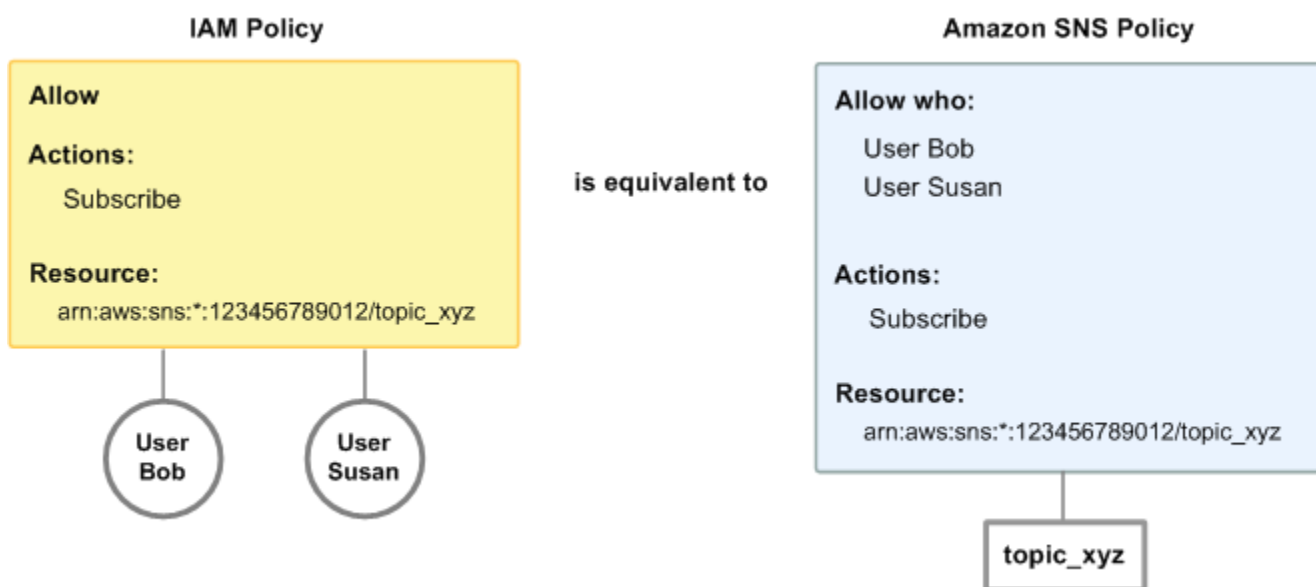
Per esempi di policy che coprono le operazioni e le risorse di Amazon SNS, consulta [Esempi di policy per Amazon SNS](#).

Utilizzo congiunto di policy IAM e Amazon SNS;

Una policy IAM; consente di limitare l'accesso degli utenti ad operazioni e argomenti di Amazon SNS. Una policy IAM può limitare l'accesso solo agli utenti all'interno AWS del tuo account, non ad altri Account AWS.

Puoi utilizzare una policy Amazon SNS con un particolare argomento per stabilire quali utenti sono autorizzati a utilizzare quell'argomento (ad esempio, chi può pubblicare messaggi nell'argomento, chi può eseguire una sottoscrizione, ecc.). Le policy di Amazon SNS possono concedere l'accesso ad altri Account AWS utenti o a utenti interni al tuo. Account AWS

Per autorizzare gli utenti ad accedere ai tuoi argomenti Amazon SNS, puoi utilizzare le policy IAM, le policy Amazon SNS o entrambe. Nella maggior parte dei casi, puoi ottenere gli stessi risultati con le une o le altre. Ad esempio, il diagramma seguente mostra l'equivalenza tra una policy IAM e una policy Amazon SNS. La policy IAM consente l'Subscription di Amazon SNS per l'argomento chiamato topic_xyz nel tuo Account AWS. La policy IAM è associata agli utenti Bob e Susan (il che significa che Bob e Susan dispongono delle autorizzazioni indicate nella policy). Anche la policy Amazon SNS fornisce a Bob e Susan l'autorizzazione per accedere all'operazione Subscribe per l'argomento topic_xyz.



Note

L'esempio precedente mostra policy semplici senza condizioni. Puoi specificare una particolare condizione in una qualsiasi delle policy e ottenere lo stesso risultato.

C'è una differenza tra le AWS policy di IAM e Amazon SNS: il sistema di policy di Amazon SNS consente di concedere l'autorizzazione ad Account AWS altri, mentre la policy IAM 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, una policy IAM e una policy Amazon SNS si applicano entrambe a Bob. La policy IAM gli concede l'autorizzazione per qualsiasi Subscribe argomento, mentre la Account AWS policy di Amazon SNS gli concede il permesso di Publish usare su un argomento specifico (topic_xyz). Il diagramma seguente illustra questo concetto.

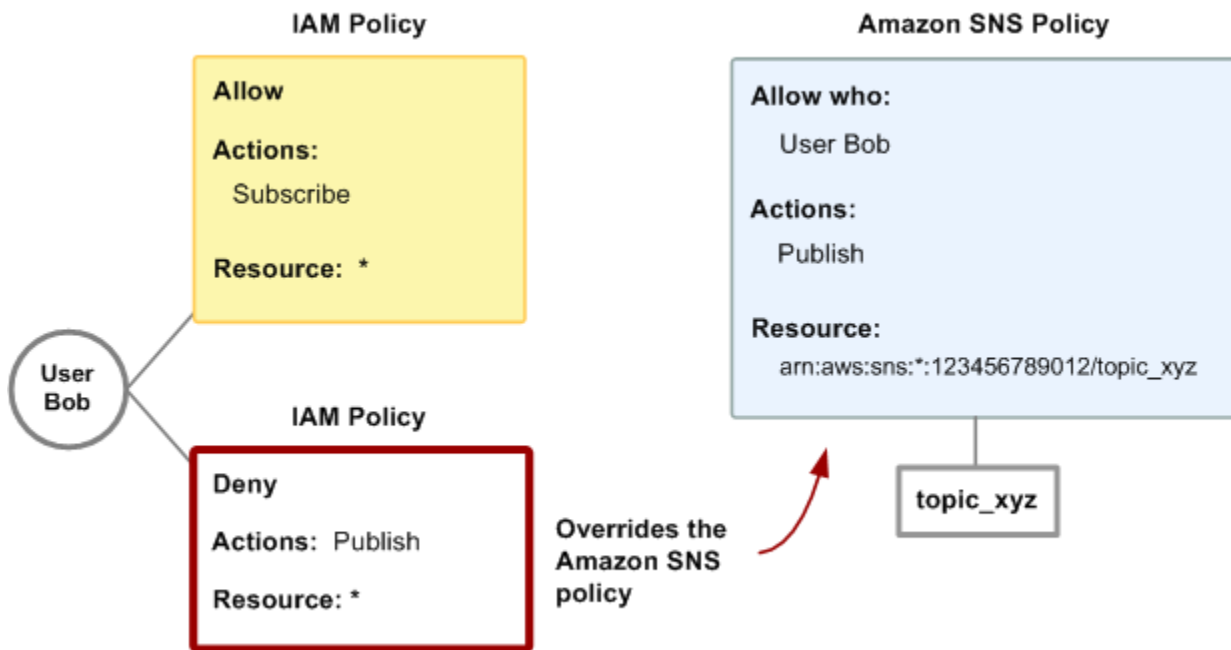


Se Bob dovesse inviare una richiesta di sottoscrizione a qualsiasi argomento dell' AWS account, la policy IAM consentirebbe l'azione. Se Bob dovesse inviare una richiesta di pubblicazione di un messaggio in topic_xyz, la policy Amazon SNS autorizzerebbe l'operazione.

Example 2

In questo esempio, facciamo riferimento all'esempio 1 (dove due policy si applicano a Bob). Supponiamo che Bob pubblichi dei messaggi in topic_xyz quando non dovrebbe farlo e che di conseguenza tu intenda negargli completamente la possibilità di pubblicare negli argomenti. La cosa più semplice da fare è aggiungere una policy IAM che gli nega l'accesso all'operazione Publish per tutti gli argomenti. Questa terza policy sostituisce la policy Amazon SNS che lo autorizzava inizialmente a pubblicare nell'argomento topic_xyz, in quanto un rifiuto esplicito sovrascrive sempre

un'autorizzazione (per ulteriori informazioni sulla logica di valutazione delle policy, consulta [Logica della valutazione](#)). Il diagramma seguente illustra questo concetto.



Per esempi di policy che coprono le operazioni e le risorse di Amazon SNS, consulta [Esempi di policy per Amazon SNS](#).

Formato ARN della risorsa Amazon SNS

Per Amazon SNS, gli argomenti sono il solo tipo di risorsa che è possibile specificare in una policy. Di seguito viene riportato il formato Amazon Resource Name (ARN) per gli argomenti.

```
arn:aws:sns:region:account_ID:topic_name
```

Per ulteriori informazioni [ARNs](#) in merito ARNs, consulta la Guida per l'utente di IAM.

Example

Di seguito è riportato un ARN per un argomento denominato MyTopic nella regione us-east-2, appartenente a 123456789012. Account AWS

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Example

Se disponevi di un argomento denominato MyTopic in ciascuna delle diverse regioni supportate da Amazon SNS, puoi specificare l'argomento con il seguente ARN.

```
arn:aws:sns:*:123456789012:MyTopic
```

Puoi utilizzare i caratteri jolly * e ? nel nome dell'argomento. Ad esempio, quanto segue potrebbe fare riferimento a tutti gli argomenti creati da Bob ai quali ha aggiunto il prefisso bob_.

```
arn:aws:sns:*:123456789012:bob_*
```

Per convenienza, quando crei un argomento, Amazon SNS restituisce l'ARN dell'argomento nella risposta.

Operazioni dell'API Amazon SNS

In una policy IAM, si può specificare qualsiasi operazione che Amazon SNS offre. Tuttavia, le operazioni `ConfirmSubscription` e `Unsubscribe` non richiedono l'autenticazione, il che significa che anche se specifichi quelle operazioni in una policy, IAM; non limiterà l'accesso degli utenti alle stesse.

Ogni operazione che specifichi in una policy deve essere preceduta dalla stringa in minuscolo `sns:`. Per esempio, per specificare tutte le operazioni di Amazon SNS, bisogna utilizzare `sns:*`. Per un elenco delle operazioni, consulta la [Documentazione di riferimento API di Amazon Simple Notification Service](#).

Chiavi di policy Amazon SNS

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 [Actions, resources and condition keys Servizi AWS nella IAM User Guide](#). Per un elenco di 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'utente IAM.

Amazon SNS utilizza le chiavi specifiche dei servizi descritte di seguito. Utilizza queste chiavi nelle policy che limitano l'accesso alle richieste `Subscribe`.

- `sns:endpoint`–URL, indirizzo e-mail o ARN di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza questa chiave con le condizioni di stringa (vedi [Esempi di policy per Amazon SNS](#)) per limitare l'accesso a specifici endpoint (ad esempio, `*@yourcompany.com`).
- `sns:protocol`– il valore `protocol` di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza questa chiave con le condizioni di stringa (vedi [Esempi di policy per Amazon SNS](#)) per limitare la pubblicazione a specifici protocolli di consegna (ad esempio, `https`).

Esempi di policy per Amazon SNS

In questa sezione vengono illustrati varie semplici policy per il controllo dell'accesso degli utenti a Amazon SNS.

Note

In futuro, è possibile che Amazon SNS aggiunga nuove operazioni che dovrebbero essere logicamente incluse in una delle seguenti policy, a seconda degli obiettivi definiti della policy.

Example Esempio 1: autorizzare un gruppo a creare e gestire argomenti

In questo esempio, creiamo una policy che autorizza l'accesso a `CreateTopic`, `ListTopics`, `SetTopicAttributes` e `DeleteTopic`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
"sns>DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example Esempio 2: autorizzare il reparto IT a pubblicare messaggi in un particolare argomento

In questo esempio, creiamo un gruppo per il reparto IT e assegniamo una policy che autorizza l'accesso a `Publish` per l'argomento desiderato.

```
{
  "Statement": [{
    "Effect": "Allow",
```

```

    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 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 policy Amazon SNS; o IAM; per consentire a un partner di pubblicare in uno specifico argomento. Se il tuo partner dispone di una Account AWS policy Amazon SNS, potrebbe essere più semplice utilizzare una policy Amazon SNS. 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). A questo proposito, devi considerare il partner come un utente nella tua azienda e utilizzare una policy IAM; anziché una policy Amazon SNS;.

Per questo esempio, creiamo un gruppo chiamato `WidgetCo` che rappresenta la società partner, creiamo un utente per la persona (o l'applicazione) specifica presso l'azienda partner che deve accedere e quindi inseriamo l'utente nel gruppo.

Quindi alleghiamo una politica che concede al gruppo l'`Publish` accesso all'argomento specifico denominato `WidgetPartnerTopic`.

Desideriamo inoltre impedire al WidgetCo gruppo di fare altro con gli argomenti, quindi aggiungiamo una dichiarazione che nega l'autorizzazione a qualsiasi azione di Amazon SNS Publish diversa da quella su argomenti diversi. WidgetPartnerTopic Ciò è necessario solo se è presente una policy generica altrove nel sistema che fornisce agli utenti ampio accesso a Amazon SNS.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
  },
  {
    "Effect": "Deny",
    "NotAction": "sns:Publish",
    "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
  }
]
```

Policy di Amazon SNS per gli endpoint di posta elettronica

Amazon SNS normalizza gli indirizzi e-mail nelle policy di accesso agli endpoint rimuovendo gli spazi bianchi iniziali e finali, convertendo tutti i caratteri in minuscolo e normalizzando i caratteri speciali e le sequenze di escape. Per questo motivo, è necessario utilizzare indirizzi e-mail in minuscolo nelle definizioni delle politiche per garantire la corretta corrispondenza delle richieste di abbonamento.

Politica di esempio:

```
{
  "Condition": {
    "StringEquals": {
      "sns:Endpoint": "user.name@example.com"
    }
  }
}
```

I seguenti indirizzi e-mail soddisfano la condizione precedente:

- user.name@example.com
- USER.NAME@EXAMPLE.COM

- `User.N\ame@example.com`

Politiche di Amazon SNS per gli endpoint HTTP/HTTPS

Amazon SNS normalizza gli HTTP/HTTPS endpoint nelle policy di accesso agli endpoint convertendo lo schema e il nome host in lettere minuscole e preservando esattamente la porta, il percorso, i parametri di query e i frammenti. Per questo motivo, è necessario utilizzare lo schema e il nome host in lettere minuscole nelle definizioni delle politiche per garantire la corretta corrispondenza delle richieste di abbonamento.

```
{
  "Condition": {
    "StringEquals": {
      "sns:Endpoint": "https://example.com:443/path?A=B"
    }
  }
}
```

Quanto segue HTTP/HTTPS corrisponde alla condizione di cui sopra:

- `HTTPS://EXAMPLE.COM:443/path?A=B`
- `HTTPS://example.com:443/path?A=B`
- `HTTPS://ExAmPlE.cOm:443/path?A=B`

Gestione di policy Amazon SNS IAM personalizzate

Le policy IAM personalizzate consentono di specificare le autorizzazioni per singoli utenti, gruppi o ruoli IAM, concedendo o limitando l'accesso a risorse e azioni specifiche AWS. Quando gestisci le risorse Amazon SNS, le policy IAM 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 le policy IAM personalizzate per Amazon SNS:

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Dal pannello di navigazione, scegli Politiche.
3. Per creare una nuova policy IAM personalizzata, scegli Crea policy e scegli SNS. Per modificare una policy esistente, seleziona la policy dall'elenco e scegli Modifica policy.

4. Nell'editor delle policy, definisci le autorizzazioni per l'accesso alle risorse Amazon SNS. Puoi specificare azioni, risorse e condizioni in base ai tuoi requisiti specifici.
5. Per concedere le autorizzazioni per le azioni Amazon SNS, includi le azioni Amazon SNS pertinenti `sns:Publish` come `sns:DeleteTopic` e nella tua `sns:Subscribe` politica IAM. Definisci l'ARN (Amazon Resource Name) degli argomenti Amazon SNS a cui si applicano le autorizzazioni.
6. Specificate gli utenti, i gruppi o i ruoli IAM a cui allegare la policy. Puoi collegare la policy direttamente agli utenti o ai gruppi IAM o associarla ai ruoli IAM utilizzati dalle Servizi AWS nostre applicazioni.
7. Rivedi la configurazione della policy IAM per assicurarti che sia in linea con i requisiti di controllo degli accessi. Una volta verificata, salva le modifiche alla policy.
8. Allega la policy IAM personalizzata agli utenti, ai gruppi o ai ruoli IAM pertinenti all'interno del tuo Account AWS. Ciò concede loro le autorizzazioni definite nella politica per la gestione delle risorse Amazon SNS.

Utilizzo delle 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 i ruoli IAM e l'accesso federato tramite protocolli standard del settore come SAML e OpenID Connect (OIDC).

Per gestire efficacemente l'accesso alle AWS risorse, è essenziale comprendere i seguenti concetti chiave:

- **Ruoli IAM:** i ruoli vengono utilizzati per delegare l'accesso alle AWS risorse. I ruoli possono essere assunti da entità come istanze Amazon EC2, funzioni Lambda o utenti di altri. Account AWS
- **Utenti federati:** si tratta di utenti autenticati tramite provider di identità esterni (IdPs) utilizzando SAML o OIDC. L'accesso federato è consigliato per gli utenti umani, mentre i ruoli IAM 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 Amazon SNS. Le librerie SDKs e API 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 dei ruoli IAM](#) e [Fornitura dell'accesso agli utenti autenticati esternamente \(federazione delle identità\)](#) nella Guida per l'utente IAM.

Example Esempio di richiesta HTTPS

L'esempio seguente dimostra come autenticare una richiesta Amazon SNS utilizzando credenziali di sicurezza temporanee ottenute AWS Security Token Service da (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 azione su Amazon SNS (ad esempio CreateTopic) e assicurati di utilizzare 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 [Use Signature Version 4 per la firma](#) nella Amazon EBS User Guide.
4. Invia la richiesta: includi l'X-Amz-Security-Token intestazione della richiesta per passare le credenziali di sicurezza temporanee ad Amazon SNS.

Autorizzazioni API Amazon SNS: riferimento a operazioni e risorse

Il seguente elenco fornisce informazioni specifiche all'implementazione del controllo accessi di Amazon SNS:

- Ogni policy deve coprire un solo argomento (durante la scrittura di una policy, non includere dichiarazioni che coprono differenti argomenti)
- Ogni policy deve avere una policy Id univoca
- Ogni dichiarazione in una policy deve avere una dichiarazione sid univoca

Quote delle policy

La tabella seguente elenca le quote massime per una dichiarazione della policy.

Name	Quota massima
Byte	30 KB
Dichiarazioni	100
Principali	Da 1 a 200 (0 non è valido).
Risorsa	1 (0 è nullo. Il valore deve corrispondere all'ARN dell'argomento della policy).

Operazioni di policy Amazon SNS valide

Amazon SNS supporta le operazioni elencate nella tabella seguente.

Azione	Description
sns: AddPermission	Autorizza l'aggiunta di autorizzazioni alla policy dell'argomento.
sms: DeleteTopic	Autorizza l'eliminazione di un argomento.
sms: GetDataProtectionPolicy	Concede l'autorizzazione per recuperare la policy di protezione dei dati dell'argomento
sms: GetTopicAttributes	Autorizza la ricezione di tutti gli attributi di argomento.
sms: ListSubscriptionsByTopic	Autorizza il recupero di tutte le sottoscrizioni a uno specifico argomento.

Azione	Description
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 PublishBatch in 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 chiavi specifiche ai servizi descritte di seguito. Puoi utilizzare queste chiavi nelle policy che limitano l'accesso alle richieste `Subscribe`.

- `sns:endpoint-URL`, indirizzo e-mail o ARN di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza con le condizioni di stringa (vedi [Esempi di policy per Amazon SNS](#)) per limitare l'accesso a specifici endpoint (ad esempio, `*@example.com`).
- `sns:protocol`– il valore `protocol` di una richiesta `Subscribe` o di una sottoscrizione confermata in precedenza. Utilizza con le condizioni di stringa (vedi [Esempi di policy per Amazon SNS](#)) per limitare la pubblicazione a specifici protocolli di consegna (ad esempio, `https`).

Important

Quando utilizzi una policy per il controllo accessi mediante `sns:Endpoint`, tieni presente che i problemi DNS potrebbero alterare la risoluzione dei nomi dell'endpoint in futuro.

Risoluzione dei problemi di identità e accesso ad Amazon Simple Notification Service

Utilizza le informazioni seguenti per eseguire la diagnosi e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di Amazon SNS.

Non dispongo dell'autorizzazione per eseguire un'operazione in Amazon SNS

Se ricevi un errore che indica che non disponi dell'autorizzazione per eseguire un'operazione, le tue policy devono essere aggiornate in modo che ti sei consentito eseguire tale operazione.

Il seguente esempio di errore si verifica quando l'utente `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa `my-example-widget` fittizia, ma non dispone di autorizzazioni `sns:GetWidget` fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
sns:GetWidget on resource: my-example-widget
```

In questo caso, la policy deve essere aggiornata in modo che Mateo possa accedere alla risorsa `my-example-widget` mediante l'operazione `sns:GetWidget`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire `iam:PassRole`

Se ricevi un errore che indica che non disponi dell'autorizzazione per eseguire l'operazione `iam:PassRole`, per poter passare un ruolo ad Amazon SNS dovrai aggiornare le policy.

Alcuni Servizi AWS consentono di passare un ruolo esistente a quel servizio invece di creare un nuovo ruolo di servizio o un ruolo collegato al servizio. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

Il seguente esempio di errore si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in Amazon SNS. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per trasmettere il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Se hai bisogno di aiuto, contatta il tuo AWS amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse Amazon SNS

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per i servizi che supportano politiche basate sulle risorse o liste di controllo degli accessi (ACLs), puoi utilizzare tali policy per consentire alle persone di accedere alle tue risorse.

Per maggiori informazioni, consulta gli argomenti seguenti:

- Per sapere se Amazon SNS supporta queste funzionalità, consulta [Come funziona Amazon SNS con IAM](#).
- Per scoprire come fornire l'accesso alle tue risorse attraverso Account AWS le risorse di tua proprietà, consulta [Fornire l'accesso a un utente IAM in un altro Account AWS di tua proprietà](#) nella IAM User Guide.
- Per scoprire come fornire l'accesso alle tue risorse a terze parti Account AWS, consulta [Fornire l'accesso a soggetti Account AWS di proprietà di terze parti](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta [Fornire l'accesso a utenti autenticati esternamente \(federazione delle identità\)](#) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Registrazione e monitoraggio in Amazon SNS

Amazon SNS ti consente di tracciare e monitorare l'attività di messaggistica registrando le chiamate API 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 flussi di lavoro di messaggistica. Questo argomento comprende quanto segue:

- [Registrazione delle chiamate API AWS SNS utilizzando AWS CloudTrail](#). Questa registrazione ti consente di tenere traccia delle azioni eseguite sugli argomenti di Amazon SNS, come la creazione di argomenti, la gestione delle sottoscrizioni e la pubblicazione dei messaggi. Analizzando

CloudTrail i log, puoi identificare chi ha effettuato richieste API specifiche e quando sono state effettuate, aiutandoti a controllare e risolvere i problemi dell'utilizzo di Amazon SNS.

- [Monitoraggio degli argomenti di Amazon SNS tramite CloudWatch](#). CloudWatch fornisce metriche che consentono di osservare le prestazioni e lo stato degli argomenti di Amazon SNS in tempo reale. Imposta allarmi in base a questi parametri, in modo da rispondere tempestivamente a eventuali anomalie, come errori di consegna o latenza elevata dei messaggi. Questa funzionalità di monitoraggio garantisce la possibilità di mantenere l'affidabilità del sistema di messaggistica basato su SNS risolvendo in modo proattivo i potenziali problemi.

Registrazione delle chiamate API AWS SNS utilizzando AWS CloudTrail

AWS SNS è integrato con [AWS CloudTrail](#), un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o un. Servizio AWS CloudTrail acquisisce tutte le chiamate API per SNS come eventi. Le chiamate acquisite includono chiamate dalla console SNS e chiamate di codice alle operazioni dell'API SNS. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a SNS, l'indirizzo IP da cui è stata effettuata, quando è stata effettuata e ulteriori dettagli.

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 le credenziali utente root o utente.
- Se la richiesta è stata effettuata per conto di un utente del Centro identità IAM.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro Servizio AWS.

CloudTrail è attivo nel tuo account Account AWS quando crei l'account e hai automaticamente accesso alla cronologia degli CloudTrail eventi. La cronologia CloudTrail degli eventi fornisce un record visualizzabile, ricercabile, scaricabile e immutabile degli ultimi 90 giorni di eventi di gestione registrati in un. Regione AWS Per ulteriori informazioni, consulta [Lavorare con la cronologia degli CloudTrail eventi](#) nella Guida per l'utente.AWS CloudTrail Non sono CloudTrail previsti costi per la visualizzazione della cronologia degli eventi.

Per un registro continuo degli eventi degli Account AWS ultimi 90 giorni, crea un trail o un data store di eventi [CloudTrailLake](#).

CloudTrail sentieri

Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Tutti i percorsi creati utilizzando il Console di gestione AWS sono multiregionali. È possibile creare un trail per una singola Regione o per più Regioni tramite AWS CLI. La creazione di un percorso multiregionale è consigliata in quanto consente di registrare l'intera attività del proprio Regioni AWS account. Se si crea un trail per una singola Regione, è possibile visualizzare solo gli eventi registrati nella Regione AWS del trail. Per ulteriori informazioni sui trail, consulta [Creating a trail for your Account AWS](#) e [Creating a trail for an organization](#) nella Guida per l'utente di AWS CloudTrail .

Puoi inviare gratuitamente una copia dei tuoi eventi di gestione in corso al tuo bucket Amazon S3 CloudTrail creando un percorso, tuttavia ci sono costi di storage di Amazon S3. [Per ulteriori informazioni sui CloudTrail prezzi, consulta la pagina Prezzi.AWS CloudTrail](#) Per informazioni sui prezzi di Amazon S3, consulta [Prezzi di Amazon S3](#).

CloudTrail Archivi di dati sugli eventi di Lake

CloudTrail Lake ti consente di eseguire query basate su SQL sui tuoi eventi. CloudTrail [Lake converte gli eventi esistenti in formato JSON basato su righe in formato Apache ORC](#). ORC è un formato di archiviazione a colonne ottimizzato per il recupero rapido dei dati. Gli eventi vengono aggregati in archivi di dati degli eventi, che sono raccolte di eventi immutabili basate sui criteri selezionati applicando i [selettori di eventi avanzati](#). I selettori applicati a un archivio di dati degli eventi controllano quali eventi persistono e sono disponibili per l'esecuzione della query. Per ulteriori informazioni su CloudTrail Lake, consulta [Working with AWS CloudTrail Lake](#) nella Guida per l'utente.AWS CloudTrail

CloudTrail Gli archivi e le richieste di dati sugli eventi di Lake comportano dei costi. Quando crei un datastore di eventi, scegli l'[opzione di prezzo](#) da utilizzare per tale datastore. L'opzione di prezzo determina il costo per l'importazione e l'archiviazione degli eventi, nonché il periodo di conservazione predefinito e quello massimo per il datastore di eventi. [Per ulteriori informazioni sui CloudTrail prezzi, consulta la sezione Prezzi.AWS CloudTrail](#)

Eventi relativi ai dati SNS in CloudTrail

Gli [eventi di dati](#) forniscono informazioni sulle operazioni delle risorse eseguite su o in una risorsa (ad esempio, lettura o scrittura su un oggetto Amazon S3). Queste operazioni sono definite anche operazioni del piano dei dati. Gli eventi di dati sono spesso attività che interessano volumi elevati di dati. Per impostazione predefinita, CloudTrail non registra gli eventi relativi ai dati. La cronologia CloudTrail degli eventi non registra gli eventi relativi ai dati.

Per gli eventi di dati sono previsti costi aggiuntivi. Per ulteriori informazioni sui CloudTrail prezzi, consulta la sezione [AWS CloudTrail Prezzi](#).

Puoi registrare gli eventi relativi ai dati per i tipi di risorse SNS utilizzando la CloudTrail console o AWS CLI le operazioni CloudTrail API. Per ulteriori informazioni su come registrare gli eventi relativi ai dati, consulta [Registrazione di eventi relativi ai dati con Console di gestione AWS](#) e [Registrazione di eventi di dati con AWS Command Line Interface](#) nella Guida per l'utente AWS CloudTrail .

La tabella seguente elenca i tipi di risorse SNS per i quali è possibile registrare gli eventi relativi ai dati. La colonna Data event type (console) mostra il valore da scegliere dall'elenco Data event type (console) sulla CloudTrail console. La colonna del valore resources.type mostra il `resources.type` valore, da specificare durante la configurazione dei selettori di eventi avanzati utilizzando o. AWS CLI CloudTrail APIs La CloudTrail colonna Dati APIs registrati mostra le chiamate API registrate per il tipo di risorsa. CloudTrail

Tipo di evento di dati (console)	valore resources.type	Dati registrati APIs su CloudTrail
Argomento SNS	AWS::SNS::Topic	• Publish • PublishBatch
Endpoint della piattaforma SNS	<code>AWS::SNS::PlatformEndpoint</code>	• Publish Per ulteriori dettagli, consulta AdvancedEventSelector l' AWS CloudTrail API Reference.

Note

Il tipo di risorsa SNS non `AWS::SNS::PhoneNumber` è registrato da. CloudTrail

È possibile configurare selettori di eventi avanzati per filtrare i campi `eventName`, `readOnly` e `resources.ARN` per registrare solo gli eventi importanti per l'utente. Per ulteriori informazioni su questi campi, consulta [AdvancedFieldSelector](#) in Riferimento API AWS CloudTrail .

Per informazioni sulla registrazione degli eventi relativi ai dati, vedere [Registrazione degli eventi relativi ai dati con Console di gestione AWS](#) e [Registrazione degli eventi relativi ai dati con the nella Guida per l' AWS CLI utente](#). CloudTrail

Eventi di gestione SNS in CloudTrail

[Gli eventi di gestione](#) forniscono informazioni sulle operazioni di gestione eseguite sulle risorse presenti nell'ambiente Account AWS. Queste operazioni sono definite anche operazioni del piano di controllo (control-plane). Per impostazione predefinita, CloudTrail registra gli eventi di gestione.

AWS SNS registra le seguenti operazioni del piano di controllo SNS come eventi di CloudTrail gestione.

- [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 [Unsubscribe](#) le azioni [ConfirmSubscription](#), queste 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

Esempi di eventi SNS

Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'operazione API richiesta, la data e l'ora dell'operazione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi gli eventi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra un CloudTrail evento che dimostra le azioni **ListTopics**, **CreateTopic**, e **DeleteTopic**.

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "userName": "Bob",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "ListTopics",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "nextToken": "ABCDEF1234567890EXAMPLE=="
      },
      "responseElements": null,
      "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
      "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
    {
      "eventVersion": "1.02",
      "userIdentity": {
        "type": "IAMUser",
        "userName": "Bob",
        "principalId": "EX_PRINCIPAL_ID",
```

```

    "arn": "arn:aws:iam::123456789012:user/Bob",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2014-09-30T00:00:00Z",
  "eventSource": "sns.amazonaws.com",
  "eventName": "CreateTopic",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-sdk-java/unknown-version",
  "requestParameters": {
    "name": "hello"
  },
  "responseElements": {
    "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
  },
  "requestID": "example7-5cd3-5323-8a00-f1889011fee9",
  "eventID": "examplec-4f2f-4625-8378-130ac89660b1",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
},
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "Bob",
    "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:user/Bob",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2014-09-30T00:00:00Z",
  "eventSource": "sns.amazonaws.com",
  "eventName": "DeleteTopic",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "127.0.0.1",
  "userAgent": "aws-sdk-java/unknown-version",
  "requestParameters": {
    "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
  },
  "responseElements": null,
  "requestID": "example5-4faa-51d5-aab2-803a8294388d",
  "eventID": "example8-6443-4b4d-abfd-1b867280d964",
  "eventType": "AwsApiCall",

```

```

    "recipientAccountId": "123456789012"
  }
]
}

```

L'esempio seguente mostra un CloudTrail evento che dimostra l'**Publish**azione.

```

{
  "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"
}
}

```

L'esempio seguente mostra un CloudTrail evento che dimostra l'**PublishBatch**azione.

```

{
  "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-aaaaea0ace2f1",
  "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"
  }
}
```

Per informazioni sul contenuto dei CloudTrail record, consultate il [contenuto dei CloudTrail record](#) nella Guida per l'AWS CloudTrail utente.

Monitoraggio degli argomenti di Amazon SNS tramite CloudWatch

Amazon SNS e Amazon CloudWatch sono integrati in modo da poter raccogliere, visualizzare e analizzare i parametri per ogni notifica Amazon SNS attiva. Dopo aver configurato CloudWatch Amazon SNS, puoi ottenere informazioni più dettagliate sulle prestazioni degli argomenti di Amazon SNS, delle notifiche push e delle consegne di SMS. Ad esempio, puoi impostare un allarme per ricevere una notifica tramite e-mail se una soglia specificata viene raggiunta per un parametro Amazon SNS come `NumberOfNotificationsFailed`. Per un elenco di tutte le metriche CloudWatch inviate da Amazon SNS, consulta [Parametri di Amazon SNS](#). Per ulteriori informazioni sulle notifiche push Amazon SNS, consulta [Invio di notifiche push per dispositivi mobili con Amazon SNS](#).

Note

Le metriche con cui configuri CloudWatch gli argomenti di Amazon SNS vengono raccolte automaticamente e inserite CloudWatch a intervalli di 1 minuto. Queste metriche sono raccolte su tutti gli argomenti che soddisfano le linee guida per essere attivi. CloudWatch Un argomento è considerato attivo CloudWatch per un massimo di sei ore dall'ultima attività (ovvero qualsiasi chiamata API) sull'argomento.

I parametri di Amazon SNS riportati in sono gratuiti CloudWatch; vengono forniti come parte del servizio Amazon SNS.

Visualizza i CloudWatch parametri per Amazon SNS

Puoi monitorare i parametri per Amazon SNS utilizzando CloudWatch la console CloudWatch, l'interfaccia a riga di comando (CLI) propria o utilizzando l'API a livello di codice. CloudWatch Le procedure seguenti mostrano come accedere ai parametri mediante la Console di gestione AWS.

Per visualizzare le metriche utilizzando la console CloudWatch

1. Accedi alla [console CloudWatch](#).
2. Nel pannello di navigazione, scegli Metrics (Parametri).
3. Nella scheda All metrics (Tutti i parametri) scegliere SNS, quindi scegliere una delle seguenti dimensioni:
 - Country, SMS Type (Paese, tipo SMS)
 - PhoneNumber
 - Topic Metrics (Parametri argomento)
 - Metrics with no dimensions (Parametri senza dimensioni)
4. Per visualizzare ulteriori dettagli, scegli un elemento specifico. Ad esempio, se scegli Topic Metrics e poi scegli NumberOfMessagesPublished, viene visualizzato il numero medio di messaggi Amazon SNS pubblicati per un periodo di 1 minuto nell'intervallo di tempo di 6 ore.
5. Per visualizzare i parametri di utilizzo di Amazon SNS, nella scheda All metrics (Tutti i parametri), scegli Usage (Utilizzo) e seleziona il parametro di utilizzo di Amazon SNS target (ad esempio, NumberOfMessagesPublishedPerAccount).

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 Console di gestione AWS 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 i parametri Amazon SNS per individuare quello per cui vuoi impostare un allarme. Seleziona il parametro per il quale vuoi creare un allarme e scegli Continue (Continua).
4. Indica i valori Name (Nome), Description (Descrizione), Threshold (Soglia) e Time (Ora) per il parametro e scegli Continue (Continua).
5. Scegli Alarm (Allarme) come stato dell'allarme. Se desideri CloudWatch inviarti un'e-mail quando viene raggiunto lo stato di allarme, scegli un argomento Amazon SNS 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 Create New Email Topic (Crea nuovo argomento e-mail) per creare un nuovo argomento Amazon SNS, gli indirizzi e-mail devono essere verificati prima di ricevere le notifiche. Le e-mail sono inviate solo quando l'allarme passa allo stato definito. Se lo stato cambia prima della verifica degli indirizzi e-mail, questi non riceveranno una notifica.

6. A questo punto, la procedura guidata per la creazione di allarmi ti consente di esaminare l'allarme che stai per creare. Se devi apportare delle modifiche, puoi utilizzare i collegamenti Edit (Modifica) a destra. Al termine, scegli Create Alarm (Crea allarme).

[Per ulteriori informazioni sull'utilizzo CloudWatch e sugli allarmi, consulta la CloudWatch documentazione.](#)

Parametri di Amazon SNS

Amazon SNS invia le seguenti metriche a CloudWatch

Namespace	Metrica	Description
AWS/SNS	NumberOfMessagesPublished	Il numero di messaggi pubblicati ai tuoi argomenti Amazon SNS. Unità: numero Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma
AWS/SNS	NumberOfNotificationsDelivered	Il numero di messaggi consegnati con successo dai tuoi argomenti Amazon SNS agli endpoint sottoscritti. Affinché un tentativo di invio abbia successo, la sottoscrizione dell'endpoint deve accettare il messaggio. Una sottoscrizione accetta un messaggio se a.) manca di un criterio di filtro o b.) i criteri di filtro includono attributi che corrispondono a quelli assegnati al messaggio. Se la sottoscrizione rifiuta il messaggio, il tentativo di invio non viene conteggiato per questo parametro. Unità: numero Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma

Namespace	Metrica	Description
AWS/SNS	NumberOfNotificationsFailed	Il numero di messaggi non consegnati da Amazon SNS. Per Amazon SQS, e-mail, SMS o endpoint push mobili, il parametro aumenta di 1 quando Amazon SNS interrompe i tentativi di consegna dei messaggi. Per gli endpoint HTTP o HTTPS, il parametro include ogni tentativo di invio non riuscito, inclusi i nuovi tentativi che seguono il tentativo iniziale. Per tutti gli altri endpoint, il numero aumenta di 1 quando il messaggio non viene distribuito (indipendentemente dal numero di tentativi). Questo parametro non include i messaggi rifiutati da policy di filtro di sottoscrizione. Puoi controllare il numero di nuovi tentativi per gli endpoint HTTP. Per ulteriori informazioni, consulta Tentativi di consegna dei messaggi di Amazon SNS. Unità: numero Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Namespace	Metrica	Description
AWS/SNS	NumberOfNotificationsFilteredOut	Il numero di messaggi rifiutati da policy di filtro di sottoscrizione. Una policy di filtro rifiuta un messaggio quando gli attributi del messaggio non corrispondono agli attributi della policy. Unità: numero Dimensioni valide: 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: numero 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: numero Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Namespace	Metrica	Description
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidAttributes	Il numero di messaggi rifiutati dalle policy di filtro di sottoscrizione perché gli attributi dei messaggi non sono validi - ad esempio perché l'attributo JSON non è formattato correttamente. Unità: numero Dimensioni valide: 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: numero Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Namespace	Metrica	Description
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidMessageBody	Il numero di messaggi rifiutati dalle policy di filtro delle sottoscrizioni perché il corpo del messaggio non è valido per il filtro (ad esempio, corpo del messaggio JSON non valido). Unità: numero Dimensioni valide: 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: numero 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: numero Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: somma, media

Namespace	Metrica	Description
AWS/SNS	PublishSize	Dimensione dei messaggi pubblicati. Per le chiamate PublishBatch API, questa metrica riporta la dimensione totale dell'intero batch come singolo punto dati, non le dimensioni dei singoli messaggi. Unità: byte Dimensioni valide: applicazione PhoneNumber, piattaforma e TopicName Statistiche valide: minimo, massimo, medio e conteggio

Namespace	Metrica	Description
AWS/SNS	SMSMonthToDateSpentUSD	I costi che hai cumulato dall'inizio del mese di calendario corrente per l'invio di SMS. Puoi impostare un allarme per questa metrica per sapere quando i tuoi month-to-date addebiti si avvicinano alla quota di spesa mensile per SMS del tuo account. Quando Amazon SNS determina che l'invio di un SMS comporter ebbe un costo superiore a tale limite, interrompe la pubblicazione di SMS nel giro di alcuni minuti. Per informazioni sull'impostazione del limite di spesa SMS mensile o sulla richiesta di aumento del limite di spesa con AWS, consulta Impostazione delle preferenze di messaggistica SMS in Amazon SNS. Unità: USD Dimensioni valide: nessuna Statistiche valide: somma
AWS/SNS	SMSSuccessRate	Il tasso di consegne SMS riuscite. Unità: numero Dimensioni valide: PhoneNumber Statistiche valide: somma, media, campioni di dati

Dimensioni per i parametri Amazon SNS

Amazon Simple Notification Service invia le seguenti dimensioni a CloudWatch.

Dimensione	Description
Application	Filtri sugli oggetti dell'applicazione, che rappresentano un'app e un dispositivo registrati con uno dei servizi di notifica push supportati, ad APNs esempio FCM.
Application,Platform	Filtri sugli oggetti dell'applicazione e della piattaforma, dove gli oggetti della piattaforma sono per i servizi di notifica push supportati, come APNs FCM.
Country	Filtra in base al paese o alla regione di destinazione di un SMS. Il paese o la regione è rappresentato dal relativo codice ISO 3166-1 alpha-2.
PhoneNumber	Filtra il numero di telefono quando pubblichi SMS direttamente su un numero di telefono (senza argomento).
Platform	Filtri sugli oggetti della piattaforma per i servizi di notifica push, come APNs FCM.
TopicName	Filtra in base ai nomi di argomenti Amazon SNS.
SMSType	Filtra in base al tipo di SMS. Può essere promozionale o transazionale.

Parametri di utilizzo di Amazon SNS

Amazon Simple Notification Service invia i seguenti parametri di utilizzo a CloudWatch.

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
AWS/Usage	SNS	ResourceCount	NumberOfMessagesPublished	Risorsa	Il numero di messaggi

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
			PublishedPerAccount		pubblicati sugli argomenti di Amazon SNS nel tuo AWS account. • Unità: nessuna • Statistiche valide: Sum
AWS/Usage	SNS	ResourceCount	ApproximateNumberOfTopics	Risorsa	• Il numero approssimativo di argomenti del tuo AWS account. • Unità: nessuna • Statistiche valide: Average (Media), Minimum (Minimo), Maximum (Massimo), Sum (Somma)

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
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)

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
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)

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
AWS/Usage	SNS	CallCount	- AddPermission - CheckIfPhoneNumberIsOptedOut - CreatePlatformApplication - CreatePlatformEndpoint - ConfirmSubscription - CreateSMSandboxPhoneNumber - CreateTopic - DeleteEndpoint - DeletePlatformApplication - DeleteSMSandboxPhoneNumber - DeleteTopic	"Hello, World!"	- Il numero di chiamate API per l'API Amazon SNS selezionata nel tuo AWS account. - Il contenuto non è consentito nella sezione finale. - Unità: nessuna - Statistiche valide: Sum

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
			• <code>GetEndpointAttributes</code> • <code>GetPlatformApplicationAttributes</code> • <code>GetSMSAttributes</code> • <code>GetSMSSandboxAccountStatus</code> • <code>GetSubscriptionAttributes</code> • <code>GetTopicAttributes</code> • <code>ListEndpointsByPlatformApplication</code> • <code>ListOriginNumbers</code> • <code>ListPhoneNumbersOptedOut</code> • <code>ListPlatformApplications</code>		

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
			- ListSMSSandboxPhoneNumbers - ListSubscriptions - ListSubscriptionsByTopic - ListTagsForResource - ListTopics - OptInPhoneNumber - RemovePermission - SetEndpointAttributes - SetPlatformApplicationAttributes - SetSMSAttributes - SetSubscriptionAttributes - SetTopicAttributes		

Namespace	Servizio	Metrica	Risorsa	Tipo	Description
			- Subscribe - Unsubscribe - UntagResource - VerifySMSSandboxPhoneNumber		

Convalida della conformità per Amazon SNS

Per sapere se un Servizio AWS programma rientra nell'ambito di specifici programmi di conformità, consulta Servizi AWS la sezione [Ambito per programma di conformità](#) di conformità e scegli il programma di conformità che ti interessa. Per informazioni generali, consulta Programmi di [AWS conformità Programmi](#) di di .

È possibile scaricare report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta [Scaricamento dei report in AWS Artifact](#) .

La vostra responsabilità di conformità durante l'utilizzo Servizi AWS è determinata dalla sensibilità dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e dai regolamenti applicabili. Per ulteriori informazioni sulla responsabilità di conformità durante l'utilizzo Servizi AWS, consulta la [Documentazione AWS sulla sicurezza](#).

Resilienza in Amazon SNS

La resilienza in Amazon SNS è garantita dall'utilizzo AWS dell'infrastruttura globale, che ruota attorno Regioni AWS alle zone di disponibilità. Regioni AWS offrono zone di disponibilità fisicamente separate e isolate collegate tramite reti a bassa latenza, ad alto throughput e altamente ridondanti. Questa architettura consente un failover senza interruzioni tra le zone di disponibilità, rendendo le applicazioni e i database intrinsecamente più tolleranti ai guasti e scalabili rispetto alle tradizionali infrastrutture di data center. Utilizzando le zone di disponibilità, gli abbonati Amazon SNS beneficiano

di una maggiore disponibilità e affidabilità, garantendo la consegna dei messaggi nonostante potenziali interruzioni. [Per ulteriori informazioni sulle zone di disponibilità, Regioni AWS consulta Global Infrastructure.AWS](#)

Inoltre, gli abbonamenti agli argomenti di Amazon SNS possono essere configurati con nuovi tentativi di consegna e code di lettera morta, abilitando la gestione automatica degli errori transitori e garantendo che i messaggi raggiungano in modo affidabile le destinazioni previste.

Amazon SNS supporta anche il filtraggio e gli attributi dei messaggi, che consentono di personalizzare le strategie di resilienza in base ai casi d'uso specifici, migliorando la robustezza complessiva delle applicazioni.

Sicurezza dell'infrastruttura in Amazon SNS

In quanto servizio gestito, Amazon SNS è protetto dalle procedure di sicurezza di rete AWS globali riportate nella documentazione [sulle migliori pratiche per la sicurezza, l'identità e la conformità](#).

Utilizza le azioni AWS API per accedere ad Amazon SNS attraverso la rete. I client devono supportare Transport Layer Security (TLS) 1.2 o versioni successive. I client devono, inoltre, supportare le suite di cifratura con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE).

Le richieste devono essere firmate utilizzando sia un ID chiave di accesso che una chiave di accesso segreta associate a un'entità principale IAM. In alternativa, è possibile utilizzare [AWS Security Token Service](#) (AWS STS) per generare credenziali di sicurezza temporanee per le richieste di firma.

Puoi richiamare queste operazioni API da qualsiasi posizione di rete, ma Amazon SNS supporta le policy di accesso basate sulle risorse, che possono includere limitazioni in base all'indirizzo IP di origine. Puoi anche utilizzare le policy di Amazon SNS per controllare l'accesso da endpoint Amazon VPC specifici o specifici VPCs. Ciò isola efficacemente l'accesso alla rete a un determinato argomento di Amazon SNS solo dal VPC specifico all'interno della rete. AWS Per ulteriori informazioni, consulta [Limitare la pubblicazione a un argomento Amazon SNS solo da un endpoint VPC specifico](#).

Best practice di sicurezza per Amazon SNS

AWS offre molte funzionalità di sicurezza per Amazon SNS. Esaminare queste caratteristiche di sicurezza nel contesto delle policy di sicurezza personalizzate.

 Note

Le indicazioni per queste caratteristiche di sicurezza si applicano ai casi d'utilizzo comuni e alle implementazioni. Si consiglia di esaminare le best practice nel contesto del caso d'uso specifico, dell'architettura e del modello di minaccia.

Best practice di prevenzione

Di seguito sono riportate le best practice di prevenzione per la sicurezza per Amazon SNS.

Argomenti

- [Assicurarsi che gli argomenti non siano accessibili pubblicamente](#)
- [Implementazione dell'accesso con privilegi minimi](#)
- [Usa i ruoli IAM per applicazioni e AWS servizi che richiedono l'accesso ad Amazon SNS](#)
- [Implementare la crittografia lato server](#)
- [Applica la crittografia dei dati in transito](#)
- [Prendere in considerazione l'utilizzo di endpoint VPC per accedere a Amazon SNS](#)
- [Assicurarsi che le sottoscrizioni non siano configurate per il recapito agli endpoint http non elaborati](#)
- [Applica l'autenticazione all'annullamento dell'iscrizione](#)

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 argomento Amazon SNS, devi assicurarti che l'argomento non sia accessibile al pubblico (accessibile da tutti nel mondo o da qualsiasi utente AWS autenticato).

- Evitare di creare policy con `Principal` impostato su `"`.
- Evitare di utilizzare un carattere jolly (*). Assegnare invece un nome a uno o più utenti specifici.

Implementazione dell'accesso con privilegi minimi

Quando si concedono autorizzazioni, si decide chi le riceve, per quali argomenti vengono fornite le autorizzazioni e le operazioni API specifiche che si desidera consentire per tali argomenti. Implementare il principio del privilegio minimo è importante per ridurre i rischi per la sicurezza. Aiuta anche a ridurre l'effetto negativo di errori o intenti dannosi.

Seguire i consigli di sicurezza standard relativi alla concessione dei privilegi minimi. Cioè, concedere solo le autorizzazioni necessarie per eseguire un'attività specifica. È possibile implementare i privilegi minimi utilizzando una combinazione di policy di sicurezza relative all'accesso degli utenti.

Amazon SNS utilizza il modello entità di pubblicazione-sottoscrittore, che richiede tre tipi di accesso dell'account utente:

- Amministratori - Accesso alla creazione, alla modifica e all'eliminazione di argomenti. Gli amministratori controllano anche le policy degli argomenti.
- Editori - Accesso all'invio di messaggi negli argomenti.
- Subscribers - Accesso alla sottoscrizione agli argomenti.

Per ulteriori informazioni, consulta le sezioni seguenti:

- [Identity and Access Management in Amazon SNS](#)
- [Autorizzazioni API Amazon SNS: riferimento a operazioni e risorse](#)

Usa i ruoli IAM per applicazioni e AWS servizi che richiedono l'accesso ad Amazon SNS

Affinché applicazioni o AWS servizi, come Amazon EC2, possano accedere agli argomenti di Amazon SNS, devono utilizzare credenziali AWS valide nelle richieste API. AWS Poiché queste credenziali non vengono ruotate automaticamente, non è necessario archiviare AWS le credenziali direttamente nell'applicazione o nell'istanza EC2.

È preferibile usare un ruolo IAM per gestire credenziali provvisorie per le applicazioni o i servizi che devono accedere ad Amazon SNS. Quando utilizzi un ruolo, non è necessario distribuire credenziali a lungo termine (ad esempio, nome utente, password e chiavi di accesso) a un'istanza EC2 o a un servizio AWS , ad esempio AWS Lambda. Al contrario, il ruolo fornisce autorizzazioni temporanee che le applicazioni possono utilizzare quando effettuano chiamate ad altre risorse. AWS

Per ulteriori informazioni, consulta [Ruoli IAM](#) e [Scenari comuni per ruoli: utenti, applicazioni e servizi](#) nella Guida per l'utente IAM.

Implementare la crittografia lato server

Per attenuare i problemi di perdita di dati, utilizzare la crittografia dei dati inattivi per crittografare i messaggi utilizzando una chiave memorizzata in un percorso diverso da quello in cui vengono

archiviati i messaggi. La crittografia lato server (SSE) fornisce la crittografia dei dati inattivi. Amazon SNS esegue la crittografia dei dati a livello di messaggio quando li memorizza e decrittografa i messaggi per l'utente quando vi accede. SSE utilizza 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, consultare [Protezione dei dati di Amazon SNS con crittografia lato server e Gestione delle chiavi e dei costi di crittografia di Amazon SNS](#).

Applica la crittografia dei dati in transito

È possibile, ma non consigliato, pubblicare messaggi che non sono crittografati durante il transito utilizzando HTTP. Tuttavia, quando un argomento viene crittografato in inattività utilizzando AWS KMS, è necessario utilizzare HTTPS per la pubblicazione dei messaggi per garantire la crittografia sia a riposo che in transito. Sebbene l'argomento non rifiuti automaticamente i messaggi HTTP, l'utilizzo di HTTPS è necessario per mantenere gli standard di sicurezza.

AWS consiglia di utilizzare HTTPS anziché HTTP. Quando si utilizza HTTPS, i messaggi vengono crittografati automaticamente durante il transito, anche se l'argomento SNS non è crittografato. Senza HTTPS, un utente malintenzionato basato sulla rete può intercettare il traffico di rete o manipolarlo utilizzando un attacco come man-in-the-middle.

Per applicare solo le connessioni crittografate su HTTPS, aggiungere il [aws:SecureTransport](#) nel criterio IAM allegato agli argomenti SNS non crittografati. In questo modo gli editori di messaggi devono utilizzare HTTPS anziché HTTP. È possibile utilizzare il seguente criterio di esempio come guida:

JSON

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:111122223333:test-topic"
      ]
    }
  ],
}
```



```
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }
}
```

Prendere in considerazione l'utilizzo di endpoint VPC per accedere a Amazon SNS

Se si dispone di argomenti con cui è necessario essere in grado di interagire ma che non devono assolutamente essere esposti a Internet, utilizzare gli endpoint VPC per limitare l'accesso agli argomenti solo agli host all'interno di un determinato VPC. Puoi utilizzare le policy tematiche per controllare l'accesso agli argomenti da endpoint Amazon VPC specifici o da specifici VPCs

Gli endpoint VPC di Amazon SNS offrono due modi per controllare l'accesso ai messaggi:

- È possibile controllare le richieste, gli utenti o i gruppi autorizzati tramite un endpoint VPC specifico.
- Puoi controllare quali endpoint VPCs o VPC hanno accesso al tuo argomento utilizzando una policy tematica.

Per ulteriori informazioni, consultare [Creazione dell'endpoint](#) e [Creazione di una policy per endpoint VPC di Amazon per Amazon SNS](#).

Assicurarsi che le sottoscrizioni non siano configurate per il recapito agli endpoint http non elaborati

Evitare di configurare le sottoscrizioni per il recapito a endpoint http non elaborati. Disponete sempre di sottoscrizioni che consegnano a un nome di dominio endpoint. Ad esempio, una sottoscrizione configurata per la consegna a un endpoint, `http://1.2.3.4/my-path`, dovrebbe essere cambiata in `http://my.domain.name/my-path`.

Applica l'autenticazione all'annullamento dell'iscrizione

A meno che non ti venga richiesto di consentire l'annullamento dell'iscrizione non autenticato, come nel caso di cancellazione semplificata per e-mail o SMS, devi applicare l'autenticazione per annullare

l'iscrizione a un argomento. [Ciò è in linea con la raccomandazione relativa al controllo degli accessi con privilegi minimi.](#)

Puoi AuthenticateOnUnsubscribe impostarlo durante la conferma di un abbonamento.

True Se non si imposta il AuthenticateOnUnsubscribe flag su True quando si conferma un abbonamento Amazon SNS, le richieste di annullamento dell'iscrizione possono avere esito positivo, anche se non sono autenticate. Per ulteriori informazioni, consulta il riferimento all'API Amazon SNS o l'[ConfirmSubscription](#) esempio di [Python nella libreria Amazon Q Detector](#).

Ad esempio, per confermare un abbonamento e-mail utilizzando AWS CLI, copia il link dal testo «Conferma abbonamento» nella notifica e-mail. Tale URL ti fornirà le informazioni necessarie per richiamare il AWS CLI comando seguente.

```
aws sns confirm-subscription --region us-west-2 \  
    --topic-arn sns-topic-arn \  
    --token token-from-subscribe-notification \  
    --authenticate-on-unsubscribe true
```

Dove:

- aws-region è AWS la regione in cui si trova l'argomento. È disponibile anche nell'argomento ARN.
- sns-topic-arn è l'ARN dell'argomento. Questo è il testo dopo «TopicArn=» e prima di «&Token» nell'URL di conferma dell'iscrizione.
- token-from-subscribe-notification è la stringa UUID dopo «Token=» e prima di «&Endpoint» nell'URL di conferma dell'iscrizione.

Di seguito è riportato un esempio di URL:

```
https://sns.us-west-2.amazonaws.com/confirmation.html?TopicArn=arn:aws:sns:us-  
west-2:123456789012:sns-  
topic&Token=a1b2c3d4e5f6789012345678901234567890abcdef1234567890abcdef12345678901234567890
```

Risoluzione dei problemi relativi ad Amazon SNS utilizzando AWS X-Ray

AWS X-Ray raccoglie dati sulle richieste servite dall'applicazione e 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 alle AWS risorse a valle, ai microservizi, ai database e al Web HTTP. APIs

È possibile utilizzare X-Ray con Amazon SNS per tracciare e analizzare i messaggi che passano attraverso la tua applicazione. Puoi utilizzare il Console di gestione AWS per visualizzare la mappa delle connessioni tra Amazon SNS e altri servizi utilizzati dall'applicazione. È inoltre possibile utilizzare la console per visualizzare i parametri come la latenza media e le percentuali di errore. Per ulteriori informazioni, consultare [Amazon SNS e AWS X-Ray](#) nella AWS X-Ray Guida per sviluppatori.

Tracciamento attivo in Amazon SNS

AWS X-Ray Utilizzalo per tracciare e analizzare le richieste degli utenti mentre passano dagli argomenti di Amazon SNS ad Amazon Data [Firehose](#), [Amazon AWS Lambda](#) SQS e agli abbonamenti [endpoint](#). HTTP/S

Con X-Ray, hai una end-to-end visione di ogni richiesta, che ti consente di:

- Identifica il tuo argomento di Amazon SNS e quali servizi derivano dai relativi abbonamenti.
- Analizza le latenze, ad esempio:
 - Tempo dedicato all'argomento Amazon SNS prima dell'elaborazione.
 - Tempi di consegna per ogni endpoint sottoscritto.

Important

Gli argomenti di Amazon SNS con numerose sottoscrizioni possono raggiungere il limite di dimensioni consentito e non essere tracciati del tutto. Per informazioni sui limiti di dimensione dei documenti di traccia, consultate la sezione [X-ray service quotas](#) nella sezione General Reference. AWS

Se chiami un'API Amazon SNS da un servizio che viene già tracciato, Amazon SNS esegue il pass-through del tracciamento, anche se sull'API non è abilitato il tracciamento di X-Ray.

Amazon SNS supporta il tracciamento di X-Ray per gli argomenti standard e FIFO. È possibile abilitare X-Ray per un argomento Amazon SNS utilizzando la [console Amazon SNS](#), [l'API SetTopicAttributes Amazon SNS](#), [la documentazione di riferimento della CLI del Servizio di notifica semplice Amazon](#) o [CloudFormation](#).

Per ulteriori informazioni sull'utilizzo di Amazon SNS con X-Ray, consulta [Amazon SNS e AWS X-Ray](#) nella Guida per gli sviluppatori di AWS X-Ray .

Autorizzazioni per il tracciamento attivo

Quando si utilizza la console Amazon SNS, Amazon SNS tenta di creare le autorizzazioni necessarie per consentire all'argomento Amazon SNS di chiamare X-Ray. Il tentativo può essere rifiutato se si non dispone di autorizzazioni sufficienti per utilizzare la console Amazon SNS. Per ulteriori informazioni, consultare [Identity and Access Management in Amazon SNS](#) e [Esempi di casi per il controllo degli accessi Amazon SNS](#).

Quando si utilizza la CLI, è necessario configurare manualmente le autorizzazioni. Tali autorizzazioni vengono configurate utilizzando le policy delle risorse. Per ulteriori informazioni sull'utilizzo delle autorizzazioni richieste in X-Ray, consulta [Amazon SNS e AWS X-Ray](#).

Abilitazione del tracciamento attivo su un argomento Amazon SNS tramite la console AWS

Quando il tracciamento attivo è abilitato su un argomento Amazon SNS, legge l'ID di tracciamento, lo utilizza per inviare i dati al cliente e lo propaga ai servizi a valle.

1. Accedi alla [console Amazon SNS](#).
2. Scegli un argomento o creane uno nuovo. Per ulteriori dettagli sulla creazione di argomenti, consulta [Creare un argomento Amazon SNS](#).
3. Nella pagina Crea argomento, nella sezione Dettagli scegli un tipo di argomento: FIFO o Standard.
 - a. Inserisci un Nome per l'argomento.
 - b. (Facoltativo) Compilare il Display name (Nome visualizzato) per l'argomento.
4. Espandi Active tracing (Monitoraggio attivo) e scegli Use active tracing (Usa tracciamento attivo).

Dopo aver abilitato X-Ray per il tuo argomento Amazon SNS, puoi utilizzare la mappa dei servizi [X-Ray per visualizzare le tracce e le mappe dei servizi](#) per end-to-end l'argomento.

Abilitazione del tracciamento attivo su un argomento Amazon SNS utilizzando l'SDK AWS

Il seguente esempio di codice mostra come abilitare il tracciamento attivo su un argomento Amazon SNS utilizzando AWS l'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 argomento Amazon SNS utilizzando la CLI AWS

Il seguente esempio di codice mostra come abilitare il tracciamento attivo su un argomento Amazon SNS utilizzando la CLI AWS .

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name TracingConfig \  
  --attribute-value Active
```

Abilitazione del tracciamento attivo su un argomento Amazon SNS utilizzando CloudFormation

Lo CloudFormation stack seguente mostra come abilitare il tracciamento attivo su un argomento di Amazon SNS.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Verifica dell'abilitazione del tracciamento attivo per l'argomento

Puoi utilizzare la console Amazon SNS per verificare se il tracciamento attivo è abilitato per il tuo argomento o quando la policy delle risorse non è stata aggiunta.

1. Accedi alla [console Amazon SNS](#).
2. Nel pannello di navigazione a sinistra, selezionare Topics (Argomenti).
3. Nella pagina Topics (Argomenti), scegli un argomento.
4. Seleziona la scheda 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.

[Amazon SNS](#) > [Topics](#) > SampleTopic

SampleTopic

[Edit](#)[Delete](#)[Publish message](#)

Details

Name	Display name
SampleTopic	-
ARN	Topic owner
arn:aws:sns:us-east-1:242420583777:DeliveryRequest	123456789123
Type	
Standard	

[< tion policy](#) | [Delivery retry policy \(HTTP/S\)](#) | [Delivery status logging](#) | [Encryption](#) | [Integrations](#) | [>](#)

AWS X-Ray active tracing

**Active tracing may require additional permission.**

We couldn't find an AWS X-Ray resource policy that allows Amazon SNS to send trace data. To create that policy now, choose "Create policy".

[Create policy](#)

Active tracing

Active

Resource policy

Not found

Test del tracciamento attivo

1. Accedi alla [console Amazon SNS](#).
2. Creazione di un argomento Amazon SNS. Per informazioni dettagliate su come eseguire questa operazione, consulta [Per creare un argomento utilizzando il Console di gestione AWS](#).
3. Espandi Active tracing (Monitoraggio attivo) e scegli Use active tracing (Usa tracciamento attivo).
4. Pubblica un messaggio nel tuo argomento Amazon SNS. Per informazioni dettagliate su come eseguire questa operazione, consulta [Per pubblicare messaggi su argomenti di Amazon SNS utilizzando Console di gestione AWS](#).
5. Utilizza la [mappa dei servizi X-Ray](#) per visualizzare le end-to-end tracce e le mappe dei servizi per l'argomento.



Cronologia della documentazione di Amazon SNS

Nella tabella seguente sono descritte le modifiche recenti apportate a Amazon Simple Notification Service Guida per sviluppatori.

Le funzionalità del servizio vengono talvolta 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
La protezione dei dati dei messaggi Amazon SNS non è più aperta ai nuovi clienti.	La protezione dei dati dei messaggi Amazon SNS non è più disponibile per i nuovi clienti. Per ulteriori informazioni e indicazioni sulle alternative, consulta il messaggio di Amazon Simple Notification Service relativo alla modifica della disponibilità della protezione dei dati.	30 aprile 2026
La protezione dei dati dei messaggi Amazon SNS non sarà più aperta ai nuovi clienti a partire dal 30 aprile 2026.	La protezione dei dati dei messaggi Amazon SNS non sarà più disponibile per i nuovi clienti a partire dal 30 aprile 2026. Per ulteriori informazioni e indicazioni sulle alternative, consulta il messaggio di Amazon Simple Notification Service relativo alla modifica della disponibilità della protezione dei dati.	31 marzo 2026

[È stato aggiunto il supporto per gli endpoint dual-stack \(e\) IPv4 IPv6](#)

Amazon SNS ora supporta le IPv6 richieste API, consentendo ai client di connettersi agli endpoint pubblici utilizzando o il dual IPv4 IPv6 stack.

03 aprile 2025

[Amazon SNS ha aggiunto l'FifoThroughputScope e attributo per gli argomenti FIFO di Amazon SNS](#)

Amazon SNS supporta l'FifoThroughputScope attributo , che specifica la quota di throughput e il comportamento di deduplicazione da applicare all'argomento FIFO. I valori validi sono Topic e MessageGroup .

21 gennaio 2025

[AmazonSNSFullAccess e aggiornamenti delle policy gestiti AmazonSNS ReadOnlyAccess](#)

Amazon SNS ha aggiunto nuove autorizzazioni AmazonSNSFullAccess e policy AmazonSNSReadOnlyAccess gestite, che consentono un accesso aggiuntivo ad Amazon SNS tramite. Console di gestione AWS

24 settembre 2024

[Integrazione con Amazon SNS AWS End User Messaging SMS per la consegna di messaggi SMS](#)

Amazon SNS supporta nuove funzionalità come la gestione delle risorse SMS, la messaggistica bidirezionale, le autorizzazioni granulari per le risorse, le regole nazionali di blocco e la fatturazione centralizzata per tutti i messaggi AWS SMS senza apportare modifiche alle configurazioni o alla rete SMS globale utilizzata da Amazon SNS. AWS

24 settembre 2024

Supporto di Canada West (Calgary) per gli argomenti FIFO	Amazon SNS supporta l'argomento FIFO in Canada occidentale (Calgary).	28 marzo 2024
Supporto SMS di Amazon SNS in cinque nuove regioni	Amazon SNS ha aggiunto il supporto SMS alle seguenti regioni: Asia Pacifico (Hyderabad), Asia Pacifico (Melbourne), Medio Oriente (Emirati Arabi Uniti), Europa (Zurigo) ed Europa (Spagna).	8 febbraio 2024
Supporto HTTP v1 di Firebase Cloud Messaging (FCM)	Amazon SNS supporta le credenziali FCM v1.	18 gennaio 2024
Amazon SNS supporta gli SMS in Asia Pacifico (Giacarta)	Amazon SNS supporta la messaggistica SMS in Asia Pacifico (Giacarta)	14 dicembre 2023
CloudFormation supporto per la configurazione DeliveryStatusLogging per argomenti di Amazon SNS	CloudFormation è disponibile il supporto per la configurazione DeliveryStatusLogging durante la creazione o l'aggiornamento degli argomenti di Amazon SNS.	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

[Supporto aggiunto per la sottoscrizione di una coda standard a un argomento FIFO](#)

È possibile effettuare la sottoscrizione di una coda FIFO o di una coda standard di Amazon SQS a un argomento FIFO di Amazon SNS. Solo le code FIFO di Amazon SQS garantiscono che i messaggi vengano ricevuti in ordine e senza duplicati.

14 settembre 2023

[Aggiunto il supporto SMS per Israele \(Tel Aviv\)](#)

SMS di Amazon SNS è ora supportato nella regione Israele (Tel Aviv).

28 agosto 2023

[Supporto per il tracciamento attivo di X-Ray aggiunto agli argomenti FIFO](#)

In precedenza supportata solo con argomenti standard di Amazon SNS, AWS X-Ray ora traccia e analizza le richieste degli utenti mentre passano dagli argomenti FIFO agli abbonamenti Amazon Data Firehose, AWS Lambda Amazon SQS ed endpoint. HTTP/S

31 maggio 2023

[Supporto intestazione Content-Type avanzato](#)

Puoi impostare l'intestazione Content-Type nella policy di richiesta per specificare il tipo di supporto della notifica.

23 marzo 2023

[Aggiunto il supporto per il tracciamento attivo](#)

AWS X-Ray traccia e analizza le richieste degli utenti mentre viaggiano attraverso gli argomenti standard di Amazon SNS verso gli abbonamenti Amazon Data Firehose, AWS Lambda Amazon SQS ed endpoint. HTTP/S

8 febbraio 2023

[Registrazione dell'ID mittente di Singapore](#)

Aggiunte istruzioni per la registrazione del mittente IDs a Singapore.

10 gennaio 2023

[Filtro dei messaggi in base al payload](#)

Il filtro basato sul payload consente di filtrare i messaggi in base al payload ed evitare i costi associati all'elaborazione di dati indesiderati.

22 novembre 2022

<u>SHA256 algoritmo hash aggiunto per la firma dei messaggi Amazon SNS</u>	Support aggiunto per l'algoritmo SHA256 hash quando si utilizza la firma dei messaggi Amazon SNS.	15 settembre 2022
<u>Aggiunta di ulteriori aree alla messaggistica SMS</u>	Amazon SNS supporta la messaggistica SMS nelle seguenti regioni: Africa (Città del Capo), Asia Pacifico (Osaka), Europa (Milano) e AWS GovCloud (Stati Uniti orientali).	9 settembre 2022
<u>Aggiunta del supporto per la protezione dei dati dei messaggi</u>	La protezione dei dati dei messaggi protegge i dati pubblicati negli argomenti di Amazon SNS utilizzando policy di protezione dei dati per controllare e bloccare le informazioni sensibili che si spostano tra applicazioni AWS o servizi.	8 settembre 2022
<u>Nuova procedura di registrazione per i numeri verdi</u>	È stato aggiunto il supporto per l'invio di messaggi Amazon SNS utilizzando numeri verdi (TFN) a destinatari negli Stati Uniti.	1 agosto 2022

[Supporto per il controllo degli accessi basato sugli attributi \(ABAC\)](#)

Aggiunto il supporto per il controllo degli accessi basato su attributi (ABAC) per le azioni API, tra cui Publish e PublishBatch . ABAC è una strategia di autorizzazione che definisce le autorizzazioni di accesso in base a tag che possono essere allegati a risorse IAM, come utenti e ruoli IAM, e a AWS risorse, come gli argomenti di Amazon SNS, per semplificare la gestione delle autorizzazioni.

10 gennaio 2022

[Supporto per l'autenticazione basata su token Apple per le notifiche push](#)

Puoi autorizzare Amazon SNS a inviare notifiche push all'app iOS o macOS fornendo informazioni che identificano l'utente come sviluppatore dell'app.

28 ottobre 2021

[I nuovi mittenti di messaggi SMS vengono inseriti nella sandbox \(ambiente di sperimentazione\) SMS](#)

La sandbox SMS è disponibile per prevenire frodi e abusi e per aiutare a proteggere la tua reputazione come mittente. Mentre il tuo AWS account è nella sandbox SMS, puoi inviare messaggi SMS solo a numeri di telefono di destinazione verificati.

1 giugno 2021

[I nuovi mittenti di messaggi SMS vengono inseriti nella sandbox \(ambiente di sperimentazione\) SMS](#)

La sandbox SMS è disponibile per prevenire frodi e abusi e per aiutare a proteggere la tua reputazione come mittente. Mentre il tuo AWS account è nella sandbox SMS, puoi inviare messaggi SMS solo a numeri di telefono di destinazione verificati.

1 giugno 2021

[Nuovi attributi per l'invio di messaggi SMS ai destinatari in India](#)

Due nuovi attributi, ID entità e ID modello, sono ora necessari per l'invio di messaggi SMS ai destinatari in India.

22 aprile 2021

[Aggiornamenti agli operatori di filtro messaggi](#)

Un nuovo operatore, `cidr`, è disponibile per gli indirizzi IP e le subnet dell'origine dei messaggi corrispondenti. Ora è anche possibile verificare 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 supportano più l'uso di codici lunghi person-to-person (P2P) per le comunicazioni application-to-person (A2P) verso destinazioni negli Stati Uniti. Invece, è possibile utilizzare codici brevi, numeri verdi o un nuovo tipo di numero di origine chiamato 10DLC.

16 febbraio 2021

[Support per i parametri Amazon CloudWatch di 1 minuto](#)

Il CloudWatch parametro di 1 minuto per Amazon SNS è ora disponibile in tutte le regioni. AWS

28 gennaio 2021

[Support per endpoint](#)

È possibile abbonare i flussi di distribuzione di Firehose agli argomenti SNS. Ciò consente di inviare notifiche a endpoint di archiviazione e analisi come bucket Amazon Simple Storage Service (Amazon S3), tabelle Amazon Redshift, Amazon Service (Service) e altro ancora OpenSearch . OpenSearch

12 gennaio 2021

[I numeri di origine sono disponibili](#)

È possibile utilizzare i numeri di origine quando si inviano messaggi di testo (SMS).

23 ottobre 2020

Supporto per gli argomenti FIFO di Amazon SNS	Per integrare applicazioni distribuite che richiedono coerenza dei dati in tempo quasi reale, puoi utilizzare gli argomenti FIFO (First-in, First-Out) di Amazon SNS con le code FIFO di Amazon SQS.	22 ottobre 2020
La libreria client ampia Amazon SNS per Java è disponibile	Puoi utilizzare questa libreria per pubblicare messaggi Amazon SNS di grandi dimensioni.	25 agosto 2020
SSE è disponibile nelle regioni della Cina	La crittografia lato server (SSE) per Amazon SNS è disponibile nelle regioni della Cina.	20 gennaio 2020
Support per l'utilizzo DLQs per l'acquisizione di messaggi non recapitabili	È possibile utilizzare una coda Amazon SQS dead-letter con una sottoscrizione Amazon SNS per acquisire messaggi non recapitabili.	14 novembre 2019
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 risolvere i problemi dei messaggi che passano attraverso gli argomenti SNS utilizzando X-Ray.	24 luglio 2019

[Supporto per la corrispondenza delle chiavi degli attributi tramite l'operatore "exists"](#)

È possibile utilizzare l'operatore `exists` per controllare se un messaggio in ingresso dispone di un attributo la cui chiave compare nella policy di filtro.

5 luglio 2019

[Support per "Anything-but" consente la corrispondenza di più valori numerici](#)

Oltre alle stringhe multiple, Amazon SNS consente la corrispondenza "Anything-but" di più valori numerici.

5 luglio 2019

[Le note di rilascio di Amazon SNS sono disponibili come feed RSS](#)

Seguendo il titolo in questa pagina (Cronologia della documentazione), scegliere RSS.

22 giugno 2019

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.