



開發人員指南

Amazon Simple Notification Service



Amazon Simple Notification Service: 開發人員指南

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

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 Amazon SNS ?	1
運作方式	1
存取 Amazon SNS	2
Amazon SNS 常見案例	3
應用程式整合	3
應用程式提醒	4
使用者通知	4
行動推送通知	4
Amazon SNS 的定價	4
Amazon SNS 功能	4
共用服務	6
使用 AWS SDKs	8
建立主題並發佈訊息	10
設定	10
建立帳戶和 IAM 使用者	10
後續步驟	12
步驟 1：建立主題	12
AWS Management Console	13
AWS SDKs	15
步驟 2：建立主題的訂閱	30
讓端點訂閱 Amazon SNS 主題	30
步驟 3：發佈訊息	32
AWS Management Console	32
AWS SDKs	33
大型訊息酬載	57
訊息屬性	68
訊息批次處理	72
步驟 4：刪除訂閱和主題	75
AWS Management Console	75
AWS SDKs	76
後續步驟	86
使用 FIFO 主題的訊息排序和重複資料刪除	87
高輸送量 FIFO 主題	87
使用案例	87

分割區與資料分配	88
分割區與資料分配	88
啟用高輸送量	88
啟用 Amazon SQS 佇列的高輸送量模式	89
FIFO 主題使用案例	89
訊息排序詳細資料	90
訊息群組	94
使用訊息群組 ID 傳遞資料以改善效能	95
訊息交付	96
訊息篩選	96
訊息重複資料刪除	98
訊息安全性	99
訊息耐久性	100
訊息封存與重播功能	102
什麼是訊息封存與重播功能	102
對於主題擁有者	103
對於主題訂閱用戶	107
程式碼範例	111
FIFO 範例(AWS SDKs)	111
FIFO 範例 (AWS CloudFormation)	123
訊息篩選	128
訂閱篩選政策範圍	128
訂閱篩選政策	128
Amazon SNS 範例篩選政策	129
篩選政策限制條件	132
AND/OR 邏輯	135
索引鍵比對	139
數值比對	141
字串值比對	143
套用訂閱篩選政策	150
AWS Management Console	151
AWS CLI	151
AWS SDKs	152
Amazon SNS API	157
AWS CloudFormation	157
移除訂閱篩選政策	158

使用 AWS Management Console	158
使用 AWS CLI	158
使用 Amazon SNS API	159
訊息資料保護	160
什麼是訊息資料保護	160
為什麼要使用訊息資料保護	160
資料保護政策	161
什麼是資料保護政策？	161
資料保護政策結構概觀	161
如何判斷 IAM 主體	163
資料保護政策操作	164
資料保護政策範例	172
建立資料保護政策	178
刪除資料保護政策	187
資料識別符	188
受管資料識別符	188
自訂資料識別符	223
訊息交付	226
原始訊息交付	226
使用 AWS Management Console啟用原始訊息交付	226
訊息格式範例	227
Amazon SQS 訂閱的訊息屬性和原始訊息傳遞	228
跨帳戶交付	228
佇列擁有者建立訂閱	228
沒有擁有佇列的使用者建立訂閱	230
如何強制訂閱以要求對取消訂閱請求進行身分驗證？	232
跨區域遞送	233
選擇加入區域	233
訊息傳遞狀態	235
交付狀態記錄的先決條件	236
使用 AWS Management Console設定交付狀態記錄日誌	237
使用 AWS SDKs 設定交付狀態記錄	238
AWS 用於設定主題屬性的 SDK 範例	240
使用 AWS CloudFormation設定交付狀態記錄日誌	248
訊息傳遞重試	250
傳遞通訊協定和政策	250

傳遞政策階段	251
建立 HTTP/S 傳遞政策	251
無效字母佇列	256
訊息傳遞為何失敗？	257
無效字母佇列的運作方式	257
訊息如何移至無效字母佇列？	258
如何將訊息從無效字母佇列中移出？	258
如何監控和記錄無效字母佇列？	258
設定無效字母佇列	259
訊息封存與分析	264
資源管理和最佳化	265
標記	265
為分配成本加上標籤	265
為存取控制加上標籤	266
為資源搜尋和過濾加上標籤	267
設定標籤	268
Amazon SNS 事件來源和目的地	274
事件來源	274
分析	274
應用程式整合	275
帳單與成本管理	275
商業應用程式	276
運算	276
容器	277
客戶參與	278
資料庫	279
開發人員工具	280
前端網頁與行動裝置	280
遊戲開發	281
物聯網	281
機器學習	282
管理與治理	283
媒體	285
移轉和傳輸	285
聯網和內容交付	286
安全、身分和合規	287

無伺服器	288
儲存	288
其他事件來源	289
事件目的地	290
A2A 目的地	290
A2P 目的地	291
應用程式至應用程式傳訊	294
散發到 Firehose 交付串流	294
先決條件	295
訂閱交付串流到主題	296
跨多個交付串流目的地管理訊息	297
訊息封存和分析範例使用案例	310
擴散到 Lambda 函數	321
先決條件	322
使用函數訂閱主題	322
擴散到 Amazon SQS 佇列	323
訂閱佇列到主題	324
使用 將 Amazon SNS 自動化至 Amazon SQS 訊息 AWS CloudFormation	330
HTTPS 端點的廣發通知	338
讓端點訂閱主題	339
驗證訊息簽章	347
剖析訊息格式	352
事件 AWS 分叉管道的散發事件	361
AWS 事件分叉管道的運作方式	362
部署 AWS 事件分叉管道	365
部署和測試事件分支管道範例應用程式	366
將事件管線訂閱主題	374
使用 EventBridge 排程器	382
設定執行角色	382
建立排程	382
相關資源	386
Application-to-person傳訊	387
行動簡訊	387
Amazon SNS 如何傳遞我的簡訊？	388
開始使用	389
來源身分	394

組態	395
SMS 訊息的最佳實務	468
傳送行動裝置推送通知	482
Amazon SNS 使用者通知的運作方式	482
使用 Amazon SNS 設定推送通知	483
設定行動應用程式	483
使用 Amazon SNS 傳送行動推播通知	500
行動應用程式屬性	511
行動應用程式事件	514
行動推送 API 動作	517
常見的行動推送 API 錯誤	518
行動推送 TTL	528
支援地區	530
行動推播通知的最佳實務	530
電子郵件訂閱設定和管理	531
AWS Management Console	532
AWS SDKs	533
程式碼範例	565
基本概念	578
您好 Amazon SNS	579
動作	591
案例	764
建置應用程式以將資料提交至 DynamoDB 資料表	765
建置 Amazon SNS 應用程式	765
為推播通知建立平台端點	767
建立無伺服器應用程式來管理相片	770
建立 Amazon Textract Explorer 應用程式	773
建立並發布到 FIFO 主題	775
偵測映像中的人物和物件	787
將簡訊發布到主題	788
發布大型訊息	794
發布簡訊	797
將訊息發佈至佇列	806
使用 API Gateway 來調用 Lambda 函數	921
使用排程事件來呼叫 Lambda 函數	923
無伺服器範例	925

使用 Amazon SNS 觸發條件調用 Lambda 函數	925
安全	936
資料加密	937
使用伺服器端加密來保護資料	937
金鑰管理	939
使用伺服器端加密設定主題加密	945
使用加密的 Amazon SQS 佇列訂閱設定主題加密	947
使用 VPC 端點保護流量	952
建立一個 VPC 端點	953
建立 VPC 政策	954
從 VPC 發佈訊息	955
訊息資料保護安全性	967
Identity and Access Management	968
目標對象	968
使用身分驗證	968
使用政策管理存取權	971
存取控制	973
存取控制使用案例	973
金鑰存取政策概念	974
架構概觀	976
使用存取原則語言	978
評估邏輯	979
Amazon SNS 存取控制的範例案例	984
Amazon SNS 如何與 IAM 搭配使用	994
AWS 受管政策	995
政策動作	1000
政策資源	1001
政策條件索引鍵	1001
ACL	1002
ABAC	1002
臨時憑證	1003
主體許可	1003
服務角色	1003
服務連結角色	1004
身分型政策範例	1004
身分型政策	1007

資源型政策	1007
使用身分型政策	1008
管理自訂 IAM 政策	1015
使用暫時登入資料	1015
API 許可參考	1017
日誌記錄和監控	1020
CloudTrail 日誌	1020
使用 CloudWatch 監控主題	1030
法規遵循驗證	1043
恢復能力	1044
基礎架構安全	1044
安全最佳實務	1045
預防性最佳實務	1045
使用 對主題進行故障診斷 AWS X-Ray	1049
主動追蹤	1049
許可	1050
啟用主動追蹤	1050
使用 AWS SDK 在 Amazon SNS 主題上啟用主動追蹤	1050
使用 CLI 在 Amazon SNS AWS 主題上啟用主動追蹤	1051
使用 在 Amazon SNS 主題上啟用主動追蹤 AWS CloudFormation	1051
確認已啟用主動追蹤	1051
測試	1052
Amazon SNS 文件歷史記錄	1054
.....	mlxi

什麼是 Amazon SNS ?

Amazon Simple Notification Service (Amazon SNS) 是一項全受管服務，可將訊息從發佈者（生產者）交付給訂閱者（消費者）。發佈者透過製作並傳送訊息到主題（其為邏輯存取點和通訊管道）與訂閱者進行非同步的通訊。

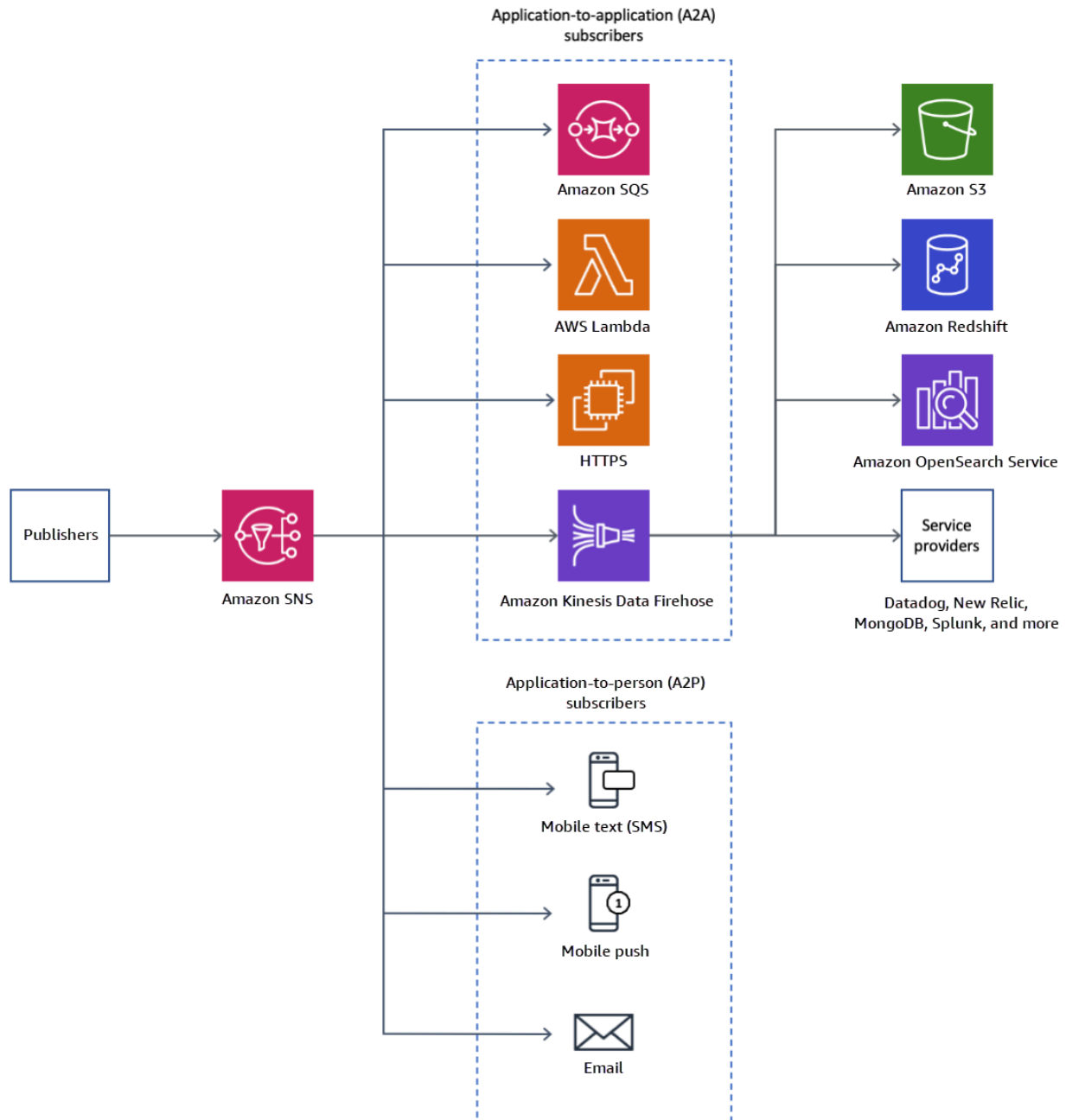
運作方式

在 SNS 中，發佈者會將訊息傳送到做為通訊管道的主題。主題充當邏輯存取點，確保訊息在不同平台上交付給多個訂閱者。

SNS 主題的訂閱者可以透過不同的端點接收訊息，取決於其使用案例，例如：

- Amazon SQS
- Lambda
- HTTP(S) 端點
- 電子郵件
- 行動推送通知
- 行動文字訊息 (SMS)
- Amazon Data Firehose
- 服務提供者（例如 Datadog、MongoDB、Splunk）

SNS 同時支援 Application-to-Application (A2A) 和 Application-to-Person (A2P) 訊息，提供在不同應用程式之間或直接傳送到行動電話、電子郵件地址等的彈性。



存取 Amazon SNS

視您偏好的互動方法而定 AWS CLI，您可以透過主控台或 AWS SDKs存取和管理 Amazon SNS。主控台為基本任務提供圖形界面，而 AWS CLI 和 SDKs為更複雜的使用案例提供進階組態和自動化功能。

- 所以此 [Amazon SNS 主控台](#) 提供便利的使用者介面，用於建立主題和訂閱、傳送和接收訊息，以及監控事件和記錄。
- AWS Command Line Interface (AWS CLI) 可讓您直接存取 Amazon SNS API 以進行進階組態和自動化使用案例。如需詳細資訊，請參閱[搭配 AWS CLI 使用 Amazon SNS](#)。
- AWS 提供多種語言 SDKs。如需詳細資訊，請參閱[開發套件與工具組](#)。

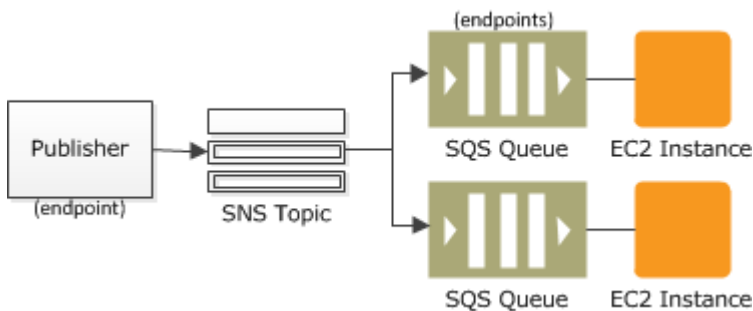
Amazon SNS 常見案例

使用這些常見的 Amazon SNS 案例來實作可擴展、事件驅動的架構，並確保應用程式和使用者之間的可靠、即時通訊。

應用程式整合

廣發性案例是指將發佈至 SNS 主題的訊息複寫並推送至多個端點，例如 Firehose 交付串流、Amazon SQS 佇列、HTTP(S) 端點和 Lambda 函數。這會允許平行非同步處理。

例如，您可以開發一個應用程式，以在系統收到產品訂單時發佈訊息到 SNS 主題。接著訂閱該 SNS 主題的 SQS 佇列均會收到相同的新訂單通知。連接到其中一個 SQS 佇列的 Amazon Elastic Compute Cloud (Amazon EC2) 伺服器執行個體可處理或履行訂單業務。您也可以將另一個 Amazon EC2 伺服器執行個體連接到資料倉儲，以便分析所有收到的訂單。



使用「發散」的另一個方法為使用您的測試環境複製傳送到生產環境的資料。延續前一個範例，您還可以为同一個 SNS 主題訂閱另一個 SQS 佇列，以處理新的訂單。透過將此新 SQS 佇列連接到測試環境，您便可利用從生產環境接收的資料，持續改進並測試您的應用程式。

Important

在將任何生產資料傳送到測試環境之前，請務必考慮資料隱私權和安全性。

如需詳細資訊，請參閱下列資源：

- [散發到 Firehose 交付串流](#)
- [將 Amazon SNS 通知散發至 Lambda 函數以進行自動化處理](#)
- [將 Amazon SNS 通知散發至 Amazon SQS 佇列以進行非同步處理](#)
- [將 Amazon SNS 通知散發至 HTTPS 端點](#)
- [使用 Amazon SNS 和 AWS 運算、儲存、資料庫和網路服務進行事件驅動運算](#)

應用程式提醒

應用程式和系統提醒是由預先定義的閾值觸發的通知。Amazon SNS 可以透過簡訊和電子郵件將這些通知傳送給指定的使用者。舉例來說，當某個事件發生時，您可以立即收到通知，例如針對 Amazon EC2 Auto Scaling 群組的特定變更、上傳到 Amazon S3 儲存貯體的新檔案或在 Amazon CloudWatch 中違反指標閾值。如需詳細資訊，請參閱 [Amazon CloudWatch 使用者指南](#) 中的設定 Amazon SNS 通知。

使用者通知

Amazon SNS 可以將推送電子郵件訊息和文字訊息 (SMS 訊息) 傳送給個人或群組。例如，您可以將電子商務訂單確認作為使用者通知傳送。如需詳細資訊，請參閱 [使用 Amazon SNS 傳送行動簡訊](#) 中的使用 Amazon SNS 訊息傳送簡訊。

行動推送通知

行動推送通知可讓您將訊息直接傳送到行動應用程式。舉例來說，您可以使用 Amazon SNS 傳送更新通知到應用程式。通知訊息可以包含下載與安裝更新的連結。如需使用 Amazon SNS 傳送推送通知訊息的詳細資訊，請參閱 [使用 Amazon SNS 傳送行動推播通知](#)。

Amazon SNS 的定價

Amazon SNS 沒有預付費用。您可以根據發佈的訊息數量、傳遞的通知數量，以及管理主題和訂閱的任何其他 API 呼叫來付費。傳遞定價會因端點類型而有所不同。您可以使用 Amazon SNS 免費方案免費開始。如需詳細資訊，請參閱 [全球簡訊定價](#)。

Amazon SNS 功能

Amazon SNS 提供一組完整的功能，旨在增強應用程式和使用者之間的傳訊。這些功能可實現無縫通訊、安全訊息交付和強大的訊息管理，確保各種簡訊使用案例的高可用性、耐用性和靈活性。

應用程式至應用程式傳訊

[Application-to-application 傳訊](#) 支援訂閱者，例如 Amazon Data Firehose 交付串流、Lambda 函數、Amazon SQS 佇列、HTTP/S 端點和 AWS 事件分叉管道。這允許在事件驅動型架構中有效率地傳遞訊息。

應用程式對人通知

[Application-to-person 通知](#) 可向訂閱用戶提供使用者通知，例如行動應用程式、行動電話號碼和電子郵件地址。

標準與 FIFO 主題

[FIFO 主題](#) 可確保嚴格的訊息排序、訊息分組和重複資料刪除，允許 FIFO 和標準佇列訂閱訊息處理。當訊息排序和可能的重複並不重要時，會使用 [標準主題](#)，以支援更廣泛的使用案例的所有交付通訊協定。

訊息持久性

Amazon SNS 使用多種合作的策略來提供訊息持久性：

- 發佈的訊息會儲存在多個地理位置分開的伺服器 and 資料中心。
- 如果訂閱的端點無法使用，Amazon SNS 會執行 [傳遞重試政策](#)。
- 若要保留傳遞重試政策結束前未傳遞的任何訊息，您可以建立 [無效字母佇列](#)。

訊息封存、重播和分析

您可以使用 Amazon SNS 以多種方式封存訊息，包括訂閱 [Firehose 交付串流至 SNS 主題](#)，這可讓您將通知傳送至分析端點，例如 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon Redshift 資料表等。此外，Amazon SNS FIFO 主題支援訊息封存與重播功能作為無程式碼的就地訊息封存，可讓主題擁有者將訊息儲存 (或封存) 在其主題內。接著訂閱用戶就可以將封存的訊息擷取 (或重播) 回訂閱的端點。如需更多資訊，請參閱 [FIFO 主題的 Amazon SNS 訊息封存和重播](#)。

訊息屬性

[Amazon SNS 訊息屬性](#) 可讓您提供有關訊息的任何任意中繼資料。

訊息篩選

根據預設，每個訂閱者會接收發佈到主題的每個訊息。若要接收一部分的訊息，訂閱者必須將篩選政策指派給主題訂閱。訂閱者也可以定義篩選政策範圍，以啟用以承載或屬性為基礎的篩選。篩選政策範圍的預設值為 MessageAttributes。當內送郵件屬性符合篩選政策屬性時，訊息會傳遞至

訂閱的端點。否則，會篩選出訊息。當篩選政策範圍為 `MessageBody`，篩選政策屬性會與承載相符。如需詳細資訊，請參閱[訊息篩選](#)。

訊息安全性

伺服器端加密會使用提供的加密金鑰，保護存放在 Amazon SNS 主題中的訊息內容 AWS KMS。如需詳細資訊，請參閱[the section called “使用伺服器端加密來保護資料”](#)您也可以在 Amazon SNS 與虛擬私有雲端 (VPC) 之間建立私有連線。如需詳細資訊，請參閱[the section called “使用 VPC 端點保護流量”](#)。

AWS 通常與 Amazon SNS 搭配使用的 服務

將 Amazon SNS 與多項 AWS 雲端 服務整合，以提升訊息處理、改善存取控制、啟用事件驅動型處理，以及自動化資源。此整合可最佳化效能、強化安全性並簡化操作。

Amazon CloudWatch

Amazon CloudWatch 為 Amazon SNS 提供監控和可觀測性，協助您追蹤訊息傳遞、偵測異常狀況，以及疑難排解問題。透過 CloudWatch，您可以：

- 監控 Amazon SNS 指標，例如跨主題和訂閱發佈、交付或失敗的訊息數量。
- 設定 CloudWatch 警示，以在 Amazon SNS 指標超過預先定義的閾值時觸發自動動作，例如高交付失敗或限流。
- 使用 CloudWatch Logs 擷取傳送至 HTTP/S、Lambda 和 Amazon SQS 端點的訊息的 Amazon SNS 交付狀態，以進行偵錯和稽核。Amazon SQS

如需詳細資訊，請參閱[使用 Amazon CloudWatch 監控 Amazon SNS 主題](#)。

Amazon SQS

Amazon SQS 是一種全受管訊息佇列服務，可在分散式軟體元件之間進行安全、耐用且可擴展的通訊。它透過緩衝訊息、確保可靠的交付，以及防止因訊息遺失而導致系統失敗，來協助解耦應用程式架構。Amazon SQS 會以下列方式與 Amazon SNS 整合：

- [無效字母佇列](#) – Amazon SNS 可以將無法傳遞的訊息路由到 Amazon SQS 無效字母佇列，以進行疑難排解和重新處理。
- [主題訂閱](#) – 您可以將 Amazon SQS 佇列訂閱 Amazon SNS 主題，允許 Amazon SNS 使用 Amazon SQS 將訊息分發給多個消費者。
- [FIFO 佇列支援](#) – Amazon SQS FIFO 佇列可以訂閱 Amazon SNS FIFO 主題，確保嚴格的訊息排序和恰好一次的處理。[標準 Amazon SQS 佇列](#)也可以訂閱 Amazon SNS 主題，但不保證排序的訊息傳遞或重複資料刪除。

AWS CloudFormation

AWS CloudFormation 使用基礎設施即程式碼 (IaC) 自動佈建和管理 AWS 資源，包括 Amazon SNS 主題和訂閱。透過 AWS CloudFormation，您可以：

- 在可重複使用、版本控制的範本中定義 Amazon SNS 主題、訂閱和許可。
- 確保跨多個 和 區域部署一致的 Amazon SNS 資源。AWS 帳戶
- 使用變更集更新或修改 Amazon SNS 組態，無需手動介入。

如需詳細資訊，請參閱 [「AWS CloudFormation 使用者指南」](#)。

AWS CloudTrail

CloudTrail 可讓您了解 Amazon SNS 的 API 活動，協助您監控和稽核對 Amazon SNS 主題、訂閱和訊息的存取。透過 CloudTrail，您可以：

- 追蹤對 Amazon SNS 進行的 API 呼叫，包括誰存取或修改了主題、訂閱和許可。
- 基於安全與合規目的分析日誌，以偵測未經授權或非預期的活動。
- 與 Amazon CloudWatch 或 整合 AWS Security Hub，以根據不尋常的 Amazon SNS 動作建立提醒。

如需更多資訊，請參閱 [使用 記錄 AWS SNS API 呼叫 AWS CloudTrail](#)。

AWS Lambda

AWS Lambda 是一種無伺服器運算服務，可自動執行程式碼以回應事件，無需佈建或管理伺服器。它可讓您建置事件驅動型應用程式，以自動擴展並在高可用性的運算環境中執行。

Amazon SNS 可讓您訂閱 Lambda 函數至 Amazon SNS 主題，以與 Lambda 整合。當 Amazon SNS 主題收到訊息時，它可以觸發 Lambda 函數，啟用即時處理、自動化和應用程式邏輯執行。此整合通常用於：

- [事件驅動處理](#) – 自動觸發函數以回應 Amazon SNS 訊息。
- [資料轉換](#) – 修改或篩選 Amazon SNS 訊息，然後再將其轉送至其他 服務。
- 自動化工作流程 – 處理應用程式提醒、系統監控或事件協同運作的通知。

AWS Identity and Access Management (IAM)

IAM 提供 AWS 資源的安全存取控制，可讓您管理誰可以存取您的 Amazon SNS 主題、他們可以執行的動作，以及在哪些條件下執行的動作。透過 IAM，您可以：

- 先驗證使用者和服務，再與 Amazon SNS 主題互動。
- 定義精細的許可，以指定使用者可以發佈、訂閱或管理哪些 Amazon SNS 主題。

- 使用身分型政策來強制執行安全最佳實務，例如限制對特定 AWS 帳戶、IP 地址或條件的存取。

如需詳細資訊，請參閱[使用以身分為基礎的政策搭配 Amazon SNS](#)。

AWS Key Management Service (AWS KMS)

AWS KMS 透過啟用伺服器端加密 (SSE) 以維護訊息機密性，增強 Amazon SNS 的安全性。透過 AWS KMS，您可以：

- 使用 AWS 受管或客戶受管加密金鑰 (CMK) 加密靜態 Amazon SNS 訊息。CMKs
- 透過定義限制誰可以發佈或訂閱的精細金鑰政策，控制對 Amazon SNS 主題的存取。
- 透過 稽核金鑰使用量，以確保符合安全和法規要求 AWS CloudTrail。

如需詳細資訊，請參閱[管理 Amazon SNS 加密金鑰和成本](#)。

AWS X-Ray

X-Ray 提供 Amazon SNS 的追蹤，協助您透過事件驅動架構來分析和偵錯訊息流程。使用 X-Ray，您可以：

- 跨多個端點追蹤 Amazon SNS 訊息傳遞 AWS 服務，例如 Lambda、Amazon SQS 和 HTTP/S 端點。
- 透過視覺化訊息的發佈、交付和處理時間來識別延遲瓶頸。
- 偵測 Amazon SNS 訊息流程中的錯誤和重試，以疑難排解失敗的交付或緩慢的處理時間。

Amazon SNS

如需詳細資訊，請參閱[Amazon SNS 中的主動追蹤](#)。

搭配 AWS SDK 使用 Amazon SNS

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
適用於 C++ 的 AWS SDK	適用於 C++ 的 AWS SDK 程式碼範例
AWS CLI	AWS CLI 程式碼範例
適用於 Go 的 AWS SDK	適用於 Go 的 AWS SDK 程式碼範例

SDK 文件	代碼範例
適用於 Java 的 AWS SDK	適用於 Java 的 AWS SDK 程式碼範例
適用於 JavaScript 的 AWS SDK	適用於 JavaScript 的 AWS SDK 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
適用於 .NET 的 AWS SDK	適用於 .NET 的 AWS SDK 程式碼範例
適用於 PHP 的 AWS SDK	適用於 PHP 的 AWS SDK 程式碼範例
AWS Tools for PowerShell	AWS Tools for PowerShell 程式碼範例
適用於 Python (Boto3) 的 AWS SDK	適用於 Python (Boto3) 的 AWS SDK 程式碼範例
適用於 Ruby 的 AWS SDK	適用於 Ruby 的 AWS SDK 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

如需 Amazon SNS 的特定範例，請參閱 [使用 AWS SDKs Amazon SNS 程式碼範例](#)。

可用性範例

找不到所需的內容嗎？請使用本頁面底部的提供意見回饋連結申請程式碼範例。

建立 Amazon SNS 主題並發佈訊息

本主題提供管理 Amazon SNS 資源的基礎步驟，特別著重於主題、訂閱和訊息發佈。首先，您將為 Amazon SNS 設定必要的存取許可，確保您擁有建立和管理 Amazon SNS 資源的正確許可。接下來，您將建立新的 Amazon SNS 主題，做為管理和交付訊息給訂閱用戶的中心中心。建立主題後，您將繼續建立此主題的訂閱，允許特定端點接收發佈至主題的訊息。

主題和訂閱到位後，您將發佈訊息到主題，觀察 Amazon SNS 如何有效率地將訊息傳遞到所有訂閱的端點。最後，您將了解如何同時刪除訂閱和主題，完成您管理的 Amazon SNS 資源生命週期。此方法可讓您清楚了解 Amazon SNS 中的基本操作，讓您具備使用 Amazon SNS 主控台管理訊息工作流程所需的實用技能。

設定 Amazon SNS 存取權限

第一次使用 Amazon SNS 之前，您必須先完成以下步驟。

建立 AWS 帳戶 和 IAM 使用者

若要存取任何 AWS 服務，您必須先建立 [AWS 帳戶](#)。您可以使用 AWS 帳戶 來檢視您的活動和用量報告，以及管理身分驗證和存取。

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 AWS 帳戶

1. 開啟 <https://portal.aws.amazon.com/billing/signup>。
2. 請遵循線上指示進行。

註冊程序的一部分包括接聽電話或文字訊息，並在電話鍵盤上輸入驗證碼。

當您註冊時 AWS 帳戶，會建立 AWS 帳戶 根使用者。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行 [需要根使用者存取權的任務](#)。

AWS 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護您的 AWS 帳戶 根使用者 AWS IAM Identity Center、啟用和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶 根使用者

1. 選擇根使用者並輸入 AWS 帳戶 您的電子郵件地址，以帳戶擁有者 [AWS Management Console](#) 身分登入。在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的 [以根使用者身分登入](#)。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

如需說明，請參閱《IAM 使用者指南》中的 [為您的 AWS 帳戶 根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

1. 啟用 IAM Identity Center。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的 [啟用 AWS IAM Identity Center](#)。

2. 在 IAM Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱 AWS IAM Identity Center 《使用者指南》中的 [使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

以具有管理存取權的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

如需使用 IAM Identity Center 使用者登入的說明，請參閱 AWS 登入 《使用者指南》中的 [登入 AWS 存取入口網站](#)。

指派存取權給其他使用者

1. 在 IAM Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的 [建立許可集](#)。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[新增群組](#)。

後續步驟

現在您已準備好使用 Amazon SNS，請開始使用：

1. [建立 Amazon SNS 主題](#)
2. [建立 Amazon SNS 主題的訂閱](#)
3. [發佈 Amazon SNS 訊息](#)
4. [刪除 Amazon SNS 主題和訂閱](#)

建立 Amazon SNS 主題

Amazon SNS 主題是做為通訊頻道的邏輯存取點。主題可讓您將多個端點（例如 AWS Lambda、Amazon SQS、HTTP/S 或電子郵件地址）分組。

若要廣播訊息生產者系統（例如，電子商務網站）的訊息，而此系統會搭配需要其訊息的其他多個服務（例如，結帳和履行系統），您可以為生產者系統建立主題。

第一個最常見的 Amazon SNS 任務是建立主題。此頁面顯示如何使用 AWS Management Console、適用於 Java 的 AWS SDK 和適用於 .NET 的 AWS SDK 來建立主題。

在建立期間，您可以選擇主題類型（標準或 FIFO）並命名主題。建立主題後，即無法變更主題類型或名稱。所有其他組態選項在建立主題期間都是可選的，您可以稍後編輯它們。

Important

請勿在主題名稱中新增個人身分識別資訊 (PII) 或其他機密或敏感資訊。包括 CloudWatch Logs 在內的其他 Amazon Web Services 可存取主題名稱。主題名稱不適用於私有或敏感資料。

使用 建立主題 AWS Management Console

在 Amazon SNS 中建立主題可建立訊息分佈的基礎，讓您能夠發佈可散發給多個訂閱用戶的訊息。此步驟對於設定主題的類型、加密設定和存取政策至關重要，可確保主題符合組織的安全性、合規性和操作要求。

1. 登入 [Amazon SNS 主控台](#)。
2. 執行以下任意一項：
 - 如果 AWS 帳戶 之前沒有在您的 下建立任何主題，請閱讀 首頁上的 Amazon SNS 說明。
 - 如果主題之前已在 下建立 AWS 帳戶，請在導覽面板上選擇主題。
3. 在主題頁面上，選擇建立主題。
4. 在 Create topic (建立主題) 頁面上，於 Details (詳細資訊) 區段中，執行以下作業：
 - a. 對於 Type (類型)，選擇主題類型 (Standard (標準)或FIFO)。
 - b. 輸入新主題的名稱。對於[FIFO 主題](#)，新增.fifo到名稱的結尾。
 - c. (選用) 為主題輸入 Display name (顯示名稱)。

Important

訂閱電子郵件端點時，Amazon SNS 主題顯示名稱和傳送電子郵件地址 (例如 no-reply@sns.amazonaws.com) 的組合字元計數不得超過 320 個 UTF-8 字元。在為 Amazon SNS 主題設定顯示名稱之前，您可以使用第三方編碼工具來驗證傳送地址的長度。

- d. (選用) 對於 FIFO 主題，您可以選擇內容型訊息重複資料刪除功能，以啟用預設的重複訊息刪除功能。如需詳細資訊，請參閱[FIFO 主題的 Amazon SNS 訊息重複資料刪除](#)。
5. (選用) 展開 Encryption (加密) 區段並執行下列動作。如需詳細資訊，請參閱[使用伺服器端加密保護 Amazon SNS 資料](#)。
 - a. 選擇 Enable encryption (啟用加密)。
 - b. 指定 AWS KMS 金鑰。如需詳細資訊，請參閱[重要用語](#)。

每個 KMS 類型均會顯示 Description (描述)、Account (帳戶) 和 KMS ARN。

⚠ Important

若您並非 KMS 的擁有者，或者您登入的帳戶並無 `kms:ListAliases` 和 `kms:DescribeKey` 的許可，即無法在 Amazon SNS 主控台上檢視 KMS 相關資訊。

請要求 KMS 的擁有者授與您這些許可。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[AWS KMS API 許可：動作和資源參考](#)。

- 預設會選取 Amazon SNS（預設）alias/aws/sns 的 AWS 受管 KMS。

ℹ Note

請謹記以下幾點：

- 第一次使用 AWS Management Console 為主題指定 Amazon SNS 的 AWS 受管 KMS 時，會 AWS KMS 建立 Amazon SNS 的 AWS 受管 KMS。
- 或者，當您第一次在已啟用 SSE 的主題上使用 Publish 動作時，會 AWS KMS 建立 Amazon SNS 的 AWS 受管 KMS。

- 若要從 AWS 您的帳戶使用自訂 KMS，請選擇 KMS 金鑰欄位，然後從清單中選擇自訂 KMS。

ℹ Note

如需建立自訂 KMS 的說明，請參閱 AWS Key Management Service 開發人員指南中的[建立金鑰](#)

- 若要從您的帳戶或其他 AWS 帳戶使用自訂 KMS ARN AWS，請在 KMS 金鑰欄位中輸入它。

6. (選用) 根據預設，只有主題擁有者可以發布或訂閱主題。若要設定其他存取許可，請展開 Access policy (存取政策) 區段。如需詳細資訊，請參閱[Amazon SNS 中的 Identity and Access Management](#)及[Amazon SNS 存取控制的範例案例](#)。

Note

當您使用主控台建立主題時，預設政策會使用 `aws:SourceOwner` 條件金鑰。此金鑰類似於 `aws:SourceAccount`。

7. (選用) 若要設定 Amazon SNS 如何在訊息傳遞嘗試失敗後重試，請展開 **Delivery retry policy (HTTP/S) (傳遞重試政策 (HTTP/S))** 區段。如需詳細資訊，請參閱[Amazon SNS 訊息傳遞重試](#)。
8. (選用) 若要設定 Amazon SNS 如何將訊息傳遞情形記錄到 CloudWatch，請展開 **Delivery status logging (傳遞狀態記錄)** 區段。如需詳細資訊，請參閱[Amazon SNS 訊息傳遞狀態](#)。
9. (選用) 若要將中繼資料標籤新增至主題，請展開 **Tags (標籤)** 區段，輸入 Key(金鑰) 和 Value (值) (選用)，然後選擇 **Add tag (新增標籤)**。如需詳細資訊，請參閱[Amazon SNS 主題標記](#)。
10. 請選擇建立主題。

該主題隨即建立，並且 **MyTopic** 頁面即會顯示。

主題的名稱、ARN、(選用) 顯示名稱和主題擁有者 AWS 的帳戶 ID 會顯示在詳細資訊區段中。

11. 將主題 ARN 複製到剪貼簿，例如：

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

使用 AWS SDK 建立主題

若要使用 AWS SDK，您必須使用登入資料進行設定。如需詳細資訊，請參閱 AWS SDK 和工具參考指南中的[共享的配置和認證文件](#)。

下列程式碼範例示範如何使用 `CreateTopic`。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

建立具有特定名稱的主題。

```
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;
    }
}
```

```
}
```

建立具有名稱、特定 FIFO 和重複資料刪除屬性的新主題。

```
/// <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;
    }

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [CreateTopic](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! 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();
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [CreateTopic](#)。

CLI

AWS CLI

建立 SNS 主題

下列 create-topic 範例會建立名為 my-topic 的 SNS 主題。

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

輸出：

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

如需詳細資訊，請參閱《[AWS 命令列界面使用者指南](#)》中的[搭配 Amazon SQS 和 Amazon SNS 使用命令列界面](#)。AWS

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CreateTopic](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}

```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [CreateTopic](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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 "";  
  }  
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [CreateTopic](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;  
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [CreateTopic](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun createSNSTopic(topicName: String): String {  
    val request =  
        CreateTopicRequest {  
            name = topicName  
        }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val result = snsClient.createTopic(request)  
        return result.topicArn.toString()  
    }  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 `CreateTopic`。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}

```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-managing-topics.html#create-a-topic>。
- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [CreateTopic](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [CreateTopic](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

# 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
  end
end

```

```

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
end

```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-create-topic.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 [CreateTopic](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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(() )  
}
```

- 如需 API 詳細資訊，請參閱適用於 Rust API 的 AWS SDK 參考中的 [CreateTopic](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP API 的 AWS SDK 參考》中的 [CreateTopic](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [CreateTopic](#)。

建立 Amazon SNS 主題的訂閱

若要接收發佈到[主題](#)的訊息，您必須訂閱[端點](#)至該主題。當您讓端點訂閱主題時，該端點會開始接收發佈到相關聯主題的訊息。

Note

其他 AWS 帳戶 中的 HTTP(S) 端點、電子郵件地址 AWS 和資源需要先確認訂閱，才能接收訊息。

讓端點訂閱 Amazon SNS 主題

訂閱 Amazon SNS 主題的端點可讓訊息傳遞至指定的端點，確保適當的系統或使用者在訊息發佈至主題時收到通知。此步驟對於將主題連結至消費者至關重要，無論他們是應用程式、電子郵件收件人或其他服務，都允許跨系統進行無縫通訊。

1. 登入 [Amazon SNS 主控台](#)。
2. 在左導覽窗格中，選擇訂閱。
3. 在訂閱頁面，選擇建立訂閱。
4. 在建立訂閱頁面上，於詳細資訊區段中，執行以下作業：

- a. 對於ARN 主題，選擇主題的 Amazon Resource Name (ARN)。此值是建立 Amazon SNS 主題時產生的 AWS ARN，例如 `arn:aws:sns:us-east-2:123456789012:your_topic`。
- b. 在 Protocol (通訊協定)，選擇端點類型。可用的端點類型為：
 - [HTTP/HTTPS](#)
 - [Email/Email-JSON \(電子郵件/電子郵件-JSON\)](#)
 - [Amazon Data Firehose](#)
 - [Amazon SQS](#)

 Note

若要訂閱[SNS FIFO 主題](#)，請選擇此選項。

- [AWS Lambda](#)
 - [平台應用程式端點](#)
 - [SMS](#)
- c. 對於Endpoint (端點)，輸入端點值，例如電子郵件地址或 Amazon SQS 佇列的 ARN。
 - d. 僅限 Firehose 端點：針對訂閱角色 ARN，指定您為寫入 Firehose 交付串流所建立之 IAM 角色的 ARN。如需詳細資訊，請參閱[訂閱 Firehose 交付串流至 Amazon SNS 主題的先決條件](#)。
 - e. (選用) 對於 Firehose、Amazon SQS、HTTP/S 端點，您也可以啟用原始訊息傳遞。如需詳細資訊，請參閱[Amazon SNS 原始訊息交付](#)。
 - f. (選用) 若要設定篩選政策，請展開 Subscription filter policy (訂閱篩選政策) 區段。如需詳細資訊，請參閱[Amazon SNS 訂閱篩選政策](#)。
 - g. (選用) 若要啟用以承載為基礎的篩選，請將 Filter Policy Scope 設定為 MessageBody。如需詳細資訊，請參閱[Amazon SNS 訂閱篩選政策範圍](#)。
 - h. (選用) 若要設定訂閱的無效字母佇列，請展開 Redrive policy (dead-letter queue) (重新磁碟機政策 (無效字母佇列)) 區段。如需詳細資訊，請參閱[Amazon SNS 無效字母佇列](#)。
 - i. 選擇 Create subscription (建立訂閱)。

主控台會建立訂閱並開啟訂閱的 Details (詳細資訊) 頁面

發佈 Amazon SNS 訊息

在您 [create an Amazon SNS topic](#) (建立 Amazon SNS 主題) 並讓端點 [subscribe](#) (訂閱) 主題之後，您可以將訊息 publish (發布) 至主題。發布訊息時，Amazon SNS 會嘗試將訊息傳送到已訂閱的[端點](#)。

將訊息發布到使用 AWS Management Console 的 Amazon SNS 主題

1. 登入 [Amazon SNS 主控台](#)。
2. 在左側導覽窗格中，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選取主題然後選擇 Publish message (發布訊息)。

主控台會開啟 Publish message to topic (將訊息發布至主題) 頁面。

4. 在 Message details (訊息詳細資訊) 區段中，執行下列動作：
 - a. (選用) 輸入訊息 Subject (主旨)。
 - b. 對於 [FIFO topic](#) (FIFO 主題)，輸入 Message group ID (訊息群組 ID)。相同訊息群組中的訊息會依發布順序傳遞。
 - c. 對於 FIFO 主題，請輸入 Message deduplication ID (訊息重複資料刪除 ID)。如果您啟用內容為基礎的訊息重複資料刪除的主題設定，此 ID 是選擇性的。
 - d. (選用) 針對 [mobile push notifications](#) (行動推播通知)，輸入 Time to Live (TTL) (存留時間 (TTL)) 值 (以秒為單位)。這是推播通知服務 (例如 Apple 推播通知服務 (APN) 或 Firebase 雲端訊息 (FCM)) 傳送訊息到端點所需的時間量。
5. 在 Message body (訊息內文) 區段中，執行下列任一動作：
 - a. 選擇 Identical payload for all delivery protocols (所有傳送協定的相同酬載) 然後輸入訊息。
 - b. 選擇 Custom payload for each delivery protocol (各傳送協定自訂酬載)，然後使用 JSON 物件定義傳送至各協定的訊息。

如需詳細資訊，請參閱 [使用平台特定承載發佈 Amazon SNS 通知](#)。
6. 在 Message attributes (訊息屬性) 區段中，新增您要讓 Amazon SNS 與 FilterPolicy 訂閱屬性比對的任何屬性，用以判斷訂閱的端點是否對發布的訊息感興趣。
 - a. 針對 Type (類型)，選擇屬性類型，例如 String.Array。

Note

針對屬性類型 `String.Array`，將陣列放在方括號 (`[]`) 中。在陣列內，以雙引號括住字串值。您不需要對數字或關鍵字 `true`、`false` 以及 `null` 使用引號。

- b. 輸入屬性名稱，例如 `customer_interests`。
- c. 輸入屬性數值，例如 `["soccer", "rugby", "hockey"]`。

如果屬性類型為 `String` (字串)、`String.Array`，或 `Number` (數字)，且篩選政策範圍未明確設為 `MessageBody`，Amazon SNS 會以訂閱的[篩選政策](#) (如果在傳送訊息到該訂閱之前出現) 來評估訊息屬性。

如需詳細資訊，請參閱[Amazon SNS 訊息屬性](#)。

7. 選擇 Publish message (發佈訊息)。

訊息會發布到主題，主控台會開啟主題的 Details (詳細資訊) 頁面。

使用 AWS SDK 將訊息發布至主題

若要使用 AWS SDK，您必須使用登入資料進行設定。如需詳細資訊，請參閱 AWS SDK 和工具參考指南中的[共享的配置和認證文件](#)。

下列程式碼範例示範如何使用 Publish。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[AWS 程式碼範例儲存庫](#)中設定和執行。

發布訊息至主題。

```
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}");
    }
}
```



```
    }  
}
```

將訊息發佈至具有群組、複寫和屬性選項的主題。

```
/// <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);
}
}

```

將使用者的選擇套用至發佈動作。

```

    /// <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;
    }

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [發佈](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
//! 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();
}
```

發佈具有 屬性的訊息。

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

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

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

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

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

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

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

    cleanUp(topicARN,
        queueURLS,
```

```
        subscriptionARNS,  
        snsClient,  
        sqsClient);  
  
    return false;  
}
```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的[發佈](#)。

CLI

AWS CLI

範例 1：將訊息發布至主題

下列 `publish` 範例會將指定的訊息發佈到指定的 SNS 主題。訊息來自文字檔案，可讓您包含換行符號。

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

`message.txt` 的內容：

```
Hello World  
Second Line
```

輸出：

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

範例 2：將簡訊發布至電話號碼

下列 `publish` 範例會將訊息 `Hello world!` 發佈至電話號碼 `+1-555-555-0100`。

```
aws sns publish \  
    --message "Hello world!" \  
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

```
--phone-number +1-555-555-0100
```

輸出：

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

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [Publish](#)。

Go

SDK for Go V2

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [發佈](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[發佈](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

將訊息發佈至具有群組、複寫和屬性選項的主題。

```
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();
}
}

```

- 如需詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/sns-examples-publishing-messages.html>。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的[發佈](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

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

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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[發佈](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-subscribing-unsubscribing-topics.html#publish-a-message-to-an-sns-topic>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[發佈](#)。

PowerShell

PowerShell V4 的工具

範例 1：此範例顯示使用單一 MessageAttribute 內嵌宣告來發佈訊息。

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

範例 2：此範例顯示預先宣告多個 MessageAttributes 的訊息發佈。

```

$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

```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 在 Cmdlet 參考 (V4) 中[發佈](#)。

PowerShell V5 的工具

範例 1：此範例顯示使用單一 MessageAttribute 內嵌宣告來發佈訊息。

```

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

```

範例 2：此範例顯示預先宣告多個 MessageAttributes 的訊息發佈。

```

$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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 在 Cmdlet 參考 (V5) 中[發佈](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

發佈具有屬性的郵件，以便訂閱可以根據屬性進行篩選。

```
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

```

發佈根據訂閱者的通訊協定採用不同形式的訊息。

```

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

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[發佈](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
# 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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-send-message.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的[發佈](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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(())
}

```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的 AWS SDK API 參考》中的[發佈](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
    oo_result = lo_sns->publish(
testing purposes. " oo_result is returned for
    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.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的[發佈](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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)")
```

- 如需 API 詳細資訊，請參閱在適用於 AWS Swift 的 SDK API 參考中[發佈](#)。

使用 Amazon SNS 和 Amazon S3 發布大型訊息

若要發布大型 Amazon SNS 訊息，您可以使用[適用於 Java 的 Amazon SNS 擴充用戶端程式庫](#)或[適用於 Python 的 Amazon SNS 擴充用戶端程式庫](#)。這些程式庫對於大於目前最大 256 KB 的訊息 (最大 2 GB) 非常有用。程式庫會將實際酬載儲存到 Amazon S3 儲存貯體，並將儲存的 Amazon S3 物件的參考發布到該主題。訂閱的 Amazon SQS 佇列可以使用[適用於 Java 的 Amazon SQS 擴展客戶端程式庫](#)從 Amazon S3 取消參照和擷取酬載資料。其他端點 (例如 Lambda) 可以使用[酬載卸載 Java 通用程式庫 AWS](#) 來取消引用並截取酬載。

 Note

Amazon SNS 延伸用戶端程式庫與標準和 FIFO 主題相容。

適用於 Java 的 Amazon SNS 擴充用戶端程式庫

先決條件

下列是使用[適用於 Java 的 Amazon SNS 擴充用戶端程式庫](#)的先決條件：

- AWS 開發套件。本頁面上的範例使用 AWS Java 開發套件。若要安裝和設定 SDK，請參閱《適用於 Java 的 AWS SDK 開發人員指南》中的[設定適用於 Java 的 AWS SDK](#)。
- AWS 帳戶 具有適當登入資料的。若要建立 AWS 帳戶，請導覽至 [AWS 首頁](#)，然後選擇建立 AWS 帳戶。遵循指示。

如需登入資料的相關資訊，請參閱《適用於 Java 的 AWS SDK 開發人員指南》中的[設定 AWS 登入資料和開發區域](#)。

- Java 8 或更高版本。
- 適用於 Java 的 Amazon SNS 擴充用戶端程式庫 (也可從 [Maven](#) 取得)。

設定訊息儲存

Amazon SNS 延伸用戶端程式庫使用適用於 AWS 訊息儲存和擷取的承載卸載 Java 通用程式庫。您可以配置以下 Amazon S3 [訊息儲存選項](#)：

- 自訂訊息大小閾值 – 承載和屬性超過此大小的訊息會自動存放在 Amazon S3 中。
- **alwaysThroughS3** flag – 將此值設定為 true，強制將所有訊息承載存放在 Amazon S3 中。例如：

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

- 自訂 KMS 金鑰 – 用於 Amazon S3 儲存貯體中伺服器端加密的金鑰。
- 儲存貯體名稱 – 用於儲存訊息承載的 Amazon S3 儲存貯體名稱。

範例：將訊息發布到 Amazon SNS，其中儲存在 Amazon S3 中的有效酬載

以下程式碼範例顯示做法：

- 建立範例主題和佇列。
- 訂閱佇列以接收來自主題的訊息。
- 發布測試訊息。

訊息酬載儲存在 Amazon S3 中，並發布對該訊息的參考。Amazon SQS 延伸用戶端是用來接收訊息。

適用於 Java 1.x 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

若要發布大型訊息，請使用適用於 Java 的 Amazon SNS 擴充用戶端程式庫。您傳送的訊息會參考包含實際訊息內容的 Amazon S3 物件。

```
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());
    }
}
```

其他端點通訊協定

Amazon SNS 和 Amazon SQS 函式庫都使用 [AWS的酬載卸載 Java 通用程式庫](#)，以使用 Amazon S3 存放和擷取訊息酬載。任何啟用 Java 的端點 (例如，在 Java 中實作的 HTTPS 端點) 都可以使用相同的庫來解除參照消息內容。

無法使用適用於的承載卸載 Java 通用程式庫的端點仍然 AWS 可以發佈存放在 Amazon S3 中的承載訊息。以下是上述代碼範例發布的 Amazon S3 參考範例：

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

適用於 Python 的 Amazon SNS 擴充用戶端程式庫

先決條件

下列是使用[適用於 Java 的 Amazon SNS 擴充用戶端程式庫](#)的先決條件：

- AWS 開發套件。本頁面上的範例使用 AWS Python SDK Boto3。若要安裝和設定 SDK，請參閱 [AWS SDK for Python](#) 說明文件。
- AWS 帳戶 具有適當登入資料的。若要建立 AWS 帳戶，請導覽至 [AWS 首頁](#)，然後選擇建立 AWS 帳戶。遵循指示。

如需憑證的資訊，請參閱 AWS SDK for Python 開發人員指南 中的[憑證](#)。

- Python 3.x (或更高版本) 和 pip。
- 適用於 Python 的 Amazon SNS 擴充用戶端程式庫(也可從 [PyPI](#) 取得)。

設定訊息儲存

下列屬性可在 Boto3 Amazon SNS [用戶端](#)、[主題](#)和[PlatformEndPoint](#) 物件上使用，以設定 Amazon S3 訊息儲存選項。

- **large_payload_support** – 將存放大型訊息的 Amazon S3 儲存貯體名稱。
- **use_legacy_attribute** – 如果為 True，則所有已發佈的訊息都會使用舊版預留訊息屬性 (SQSLargePayloadSize)，而非目前的預留訊息屬性 (ExtendedPayloadSize)。

- **message_size_threshold** – 將郵件儲存在大型訊息儲存貯體中的臨界值。不能小於 0，或大於 262144。預設值為 262144。
- **always_through_s3** – 如果是 True，則所有訊息都存放在 Amazon S3 中。預設值為 False。
- **s3_client** – 用來將 client 物件存放至 Amazon S3 的 Boto3 Amazon S3 物件。如果您想要控制 Amazon S3 用戶端（例如，自訂 Amazon S3 組態或登入資料），請使用此選項。如果先前未設定，則第一次使用 `boto3.client("s3")` 時預設為。

範例：將訊息發布到 Amazon SNS，其中儲存在 Amazon S3 中的有效酬載

以下程式碼範例顯示做法：

- 建立範例 Amazon SNS 主題和 Amazon SQS 佇列。
- 將政策連接至 Amazon SQS 佇列，以接收來自 Amazon SNS 主題的訊息。
- 訂閱佇列以接收來自主題的訊息。
- 使用 Amazon SNS 延伸用戶端、主題資源和 PlatformEndpoint 資源發佈測試訊息。
- 訊息酬載儲存在 Amazon S3 中，並發布對該訊息的參考。
- 列印佇列中已發布的訊息以及從 Amazon S3 擷取的原始訊息。

若要發布大型訊息，請使用適用於 Python 的 Amazon SNS 擴充用戶端程式庫。您傳送的訊息會參考包含實際訊息內容的 Amazon S3 物件。

```
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)

```

輸出

```

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-
xxxxxxxxxxxx"}]

```

```
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-xxxxxxxxxxxx"}]
```

```
Message Stored in S3 Bucket is: This message should be published to S3 using the PlatformEndpoint resource
```

Amazon SNS 訊息屬性

Amazon SNS 支援傳遞訊息屬性，可讓您提供與訊息相關的結構化中繼資料項目 (例如時間戳記、地理空間資料、簽章及識別符)。啟用[原始訊息交付](#)以後，針對 SQS 訂閱最多可傳送 10 個訊息屬性。如要傳送超過 10 個訊息屬性，則必須停用原始訊息交付。具有 10 個以上訊息屬性且導向已啟用「原始訊息交付」的 Amazon SQS 訂閱的訊息，將會因用戶端錯誤而遭到捨棄。

訊息屬性為選用且與訊息內文分開，但與訊息內文一起傳送。接收者可以使用這項資訊來決定如何處理訊息，而不必先處理訊息內文。

如需有關使用 AWS Management Console 或 傳送具有屬性的訊息的資訊 適用於 Java 的 AWS SDK，請參閱 [將訊息發布到使用 AWS Management Console 的 Amazon SNS 主題](#) 教學課程。

Note

只有在訊息結構是字串而非 JSON 時，才會傳送訊息屬性。

您也可以使用訊息屬性來協助結構化行動裝置端點的推送通知訊息。在此情況下，訊息屬性只會用來協助結構化推送通知訊息。屬性不會交付到端點，與搭配訊息屬性將訊息傳送到 Amazon SQS 端點時不同。

您也可以使用訊息屬性，讓您的訊息可透過使用訂閱篩選政策進行篩選。您可以將篩選政策套用到主題訂閱。以設為 MessageAttributes (預設) 的篩選政策範圍套用篩選政策時，訂閱只會接收到擁有政策所接受屬性的訊息。如需詳細資訊，請參閱[Amazon SNS 訊息篩選](#)。

Note

使用訊息屬性進行篩選時，值必須是有效的 JSON 字串。這樣做可確保訊息傳遞至已啟用訊息屬性篩選的訂閱。

訊息屬性項目和驗證

每項訊息屬性均是由以下項目組成：

- **Name** - 訊息屬性名稱可包含下列字元：A-Z、a-z、0-9、底線 (_)、連字號 (-) 和句號 (.)。名稱不能以句號開頭或結尾，且不可連續使用句號。名稱會區分大小寫，且不能與訊息的其他所有屬性名稱重複。名稱長度上限為 256 個字元。名稱無法以 AWS. 或 Amazon. (或任何外殼的變體) 開頭，因為這類字首會預訂給 Amazon Web Services 使用。
- **類型** - 支援的訊息屬性資料類型為 String、String.Array、Number 和 Binary。資料類型內容上的限制與訊息內文相同。如需詳細資訊，請參閱 [訊息屬性資料類型和驗證](#) 一節。
- **數值** - 使用者指定的訊息屬性數值。對於字串資料類型，值屬性必須遵循與訊息內文相同的內容限制。不過，如果訊息屬性用於篩選，則值必須是有效的 JSON 字串，以確保與 Amazon SNS 訂閱篩選政策的相容性。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中 [Publish](#) (發佈) 動作。

名稱、類型和值不能為空或 null。此外，訊息本文不能為空或 null。訊息屬性的所有部分，包括名稱、類型和值，均包含在訊息大小限制中，目前限制為 256 KB。

訊息屬性資料類型和驗證

訊息屬性資料類型可識別 Amazon SNS 如何處理訊息屬性的值。例如，如果類型為數字，Amazon SNS 會驗證其是否為數字。

Amazon SNS 支援下列所有端點的邏輯資料類型，除非另有說明：

- **字串** - 字串為 UTF-8 二進位編碼的 Unicode。如需代碼值的清單，請參閱 http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters。

Note

訊息屬性不支援代理值。例如，使用代理值來表示表情符號會出現以下錯誤：Invalid attribute value was passed in for message attribute。

- **String.Array** - 格式化為字串的陣列，可包含多個值。值可以是字串、數字或關鍵字 true、false 和 null。數字或布林值類型的 String.Array 不需要引號。以逗號分隔的多個 String.Array 值。

AWS Lambda 訂閱不支援此資料類型。如果您為 Lambda 端點指定此資料類型，則會以 String 資料類型中，Amazon SNS 提供給 Lambda 的 JSON 酬載傳遞。

- 數字 - 數字為正負整數或浮點數字。數字需有足夠的範圍和精準度，方可涵蓋大多數整數、浮點數、雙精度浮點數一般支援的可能數值。一數字的值可從 -10^9 至 10^9 ，準確度到包含小數點後的 5 位數。前後的零會截去。

AWS Lambda 訂閱不支援此資料類型。如果您為 Lambda 端點指定此資料類型，則會以 String 資料類型中，Amazon SNS 提供給 Lambda 的 JSON 酬載傳遞。

- 二進位 - 二進位類型屬性可儲存任何二進位資料；例如壓縮資料、加密資料或影像。

為行動推播通知保留的訊息屬性

下表列出為行動推播通知服務 (您可用來結構化您的推播通知訊息) 保留的訊息屬性：

推送通知服務	預留的訊息屬性
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>

推送通知服務	預留的訊息屬性
百度	<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>
	<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 如果訊息屬性不符合其需求，Apple 將拒絕 Amazon SNS 通知。如需其他詳細資訊，請參閱 Apple 開發人員網站上的[傳送通知請求至 APN](#)。

Amazon SNS 訊息批次處理

什麼是訊息批次處理？

在單一 Publish API 請求中，將訊息發佈至標準或 FIFO 主題的替代方案，在使用 Amazon SNS PublishBatch API 於單一 API 請求中發佈多達 10 則訊息。批次傳送訊息可協助您降低與使用 Amazon SNS 連線分散式應用程式 ([A2A 訊息](#)) 或將通知傳送給人員 ([A2P 訊息](#)) 相關聯的成本，降低成本的倍數高達 10 倍。Amazon SNS 會根據您的營運區域，針對每秒可發佈到主題的訊息數量提供配額。如需 API 配額的詳細資訊，請參閱《AWS 一般參考指南》中的 [Amazon SNS 端點和配額](#) 頁面。

Note

您在單一 PublishBatch API 請求中傳送的所有訊息的總彙總大小不得超過 262,144 位元組 (256 KiB)。

該 PublishBatch API 對 IAM 政策使用相同的 Publish API 動作。

訊息批次處理如何運作？

使用 PublishBatch API 發佈訊息類似於使用 Publish API 發佈訊息。主要區別在於 PublishBatch API 請求中的每個訊息需獲得所指派的唯一批次 ID (最多 80 個字元)。如此一來，Amazon SNS 可以針對批次中的每個訊息傳回個別 API 回應，以確認已發佈每個訊息或發佈失敗。對於發佈到 FIFO 主題的訊息，除了包含指派唯一批次 ID 之外，您還需要包含每個訊息的 MessageDeduplicationID 和 MessageGroupId。

範例

將 10 則訊息的批次發佈至標準主題

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

將 10 則訊息的批次發佈至 FIFO 主題

```
// 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);
    } catch (SnsException e) {
        // Handle exception
    }
}
```



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

刪除 Amazon SNS 主題和訂閱

刪除主題時，會以非同步方式刪除其相關聯的訂閱。雖然客戶仍然可以存取這些訂閱，但即使您使用相同名稱重新建立主題，訂閱也不再與主題相關聯。如果發佈者嘗試發佈訊息到已刪除的主題，發佈者將會收到錯誤訊息，指出主題不存在。同樣地，只要嘗試訂閱已刪除的主題也會產生錯誤訊息。您無法刪除正在進行確認的訂閱。Amazon SNS 會在 48 小時後自動刪除未確認的訂閱。

使用 刪除 Amazon SNS 主題或訂閱 AWS Management Console

刪除 Amazon SNS 主題或訂閱可確保有效的資源管理、防止不必要的使用，並保持 Amazon SNS 主控台井然有序。此步驟有助於避免閒置資源的潛在成本，並移除不再需要的主題或訂閱，以簡化管理。

使用 刪除主題 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。

2. 在左側導覽窗格中，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選取主題，然後選擇 Edit (編輯)。
4. 在 Delete topic (刪除主題) 對話方塊中輸入 delete me，然後選擇 Delete (刪除)。

主控台刪除主題。

使用 刪除訂閱 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在左側導覽窗格中，選擇訂閱。
3. 在訂閱頁面上，選取狀態為已確認的訂閱，然後選擇刪除。
4. 在 Delete subscription (刪除訂閱) 對話方塊中，選擇 Delete (刪除)。

主控台刪除訂閱。

使用 AWS SDK 刪除訂閱和主題

若要使用 AWS SDK，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS SDK 和工具參考指南中的 [共享的配置和認證文件](#)。

下列程式碼範例示範如何使用 DeleteTopic。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

藉由主題 ARN 刪除該主題。

```
/// <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;
}

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [DeleteTopic](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! 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();
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

CLI

AWS CLI

刪除 SNS 主題

下列 delete-topic 範例會刪除指定的 SNS 主題。

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

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteTopic](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [DeleteTopic](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DeleteTopic](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```

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({});

```

匯入 SDK 和用戶端模組，然後呼叫 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
  //   }
  // }
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }
}
```



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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 DeleteTopic。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [DeleteTopic](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [DeleteTopic](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import AWSSNS

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

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

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [DeleteTopic](#)。

後續步驟

現在您已經建立有訂閱的主題，並已將訊息傳送到主題，您可以再試試下列各項：

- 探索 [AWS 開發人員中心](#)。
- 請至[安全性](#)章節了解如何保護您的資料。
- 對主題啟用[伺服器端加密](#)。
- 對已訂閱的[加密 Amazon Simple Queue Service \(Amazon SQS\) 佇列](#)主題啟用伺服器端加密。
- 訂閱 [AWS 事件分叉管道](#)主題。

使用 Amazon SNS FIFO 主題的訊息排序和重複資料刪除策略

本主題提供 Amazon SNS FIFO (First-In-First-Out) 主題的功能資訊，以及它們如何與 [Amazon SQS FIFO 佇列](#) 整合。您將了解如何一起使用這些服務，以確保嚴格的訊息排序和重複資料刪除，這對於需要資料一致性的應用程式至關重要。此內容涵蓋 Amazon SNS FIFO 主題有益的特定使用案例，可讓您深入了解訊息順序和唯一性至關重要的情況。

您也將了解訊息排序、訊息分組的技術詳細資訊，以及這些詳細資訊如何影響訊息傳遞。訊息重複資料刪除主題說明防止重複訊息的機制，確保每則訊息只會處理一次。此外，您將了解訊息篩選、安全性和耐久性，這對於維護訊息系統的完整性和可靠性至關重要。此內容也包含訊息封存和重播的相關資訊，提供管理訊息歷史記錄的策略。也提供實用的程式碼範例，協助您在自己的應用程式中實作這些功能，讓您實際體驗 Amazon SNS FIFO 主題及其與 Amazon SQS FIFO 佇列的整合。

Amazon SNS 中的高輸送量 FIFO 主題

Amazon SNS 中的高輸送量 FIFO 主題可有效管理高訊息輸送量，同時保持嚴格的訊息順序，確保處理大量訊息的應用程式的可靠性和可擴展性。此解決方案非常適合需要高輸送量和排序訊息交付的案例。若要使用高輸送量 FIFO 主題改善訊息輸送量，建議增加訊息群組的數量。如需高輸送量訊息配額的詳細資訊，請參閱 [Amazon SNS 服務配額](#) Amazon Web Services 一般參考。

Amazon SNS FIFO 主題高輸送量的使用案例

下列使用案例強調高輸送量 FIFO 主題的各種應用程式，展示其在各產業和情境中的效率：

- 即時資料處理：處理事件處理或遙測資料擷取等即時資料串流的應用程式，可以受益於高輸送量 FIFO 主題，以處理訊息的持續流入，同時保留其順序以進行準確分析。
- 電子商務訂單處理：在維護客戶交易訂單至關重要的電子商務平台上，高輸送量 FIFO 主題可確保訂單按順序交付，即使在尖峰購物季節也不會延遲。
- 金融服務：處理高頻率交易或交易資料的金融機構依賴高輸送量 FIFO 主題來處理市場資料和交易，同時遵守嚴格的訊息排序法規要求。
- 媒體串流：串流平台和媒體分發服務利用高輸送量 FIFO 主題來管理媒體檔案和串流內容的交付，確保使用者能順暢播放體驗，同時維持內容交付的正確順序

Amazon SNS FIFO 主題高輸送量的分割區和資料分佈

使用高輸送量主題時，Amazon SNS 會將 FIFO 主題資料分散到各個分割區。分割區是主題的容量配置，該主題會自動複寫到內的多個可用區域 AWS 區域。您未管理分割區。反之，Amazon SNS 會根據輸入速率，代表您自動管理分割區。

對於 FIFO 主題，Amazon SNS 會在下列情況修改主題中的分割區數量：

- 如果目前的發佈率接近或超過現有分割區可支援的範圍，則會分配額外的分割區，直到主題達到區域配額為止。如需配額的相關資訊，請參閱中的 [Amazon SNS 服務配額](#) Amazon Web Services 一般參考。
- 如果目前的分割區使用率低，則可能會減少分割區的數目。

分割區管理會在背景自動進行，而且對您的應用程式是透明的。您的主題和訊息隨時可用。

Note

如果您突然且大幅增加主題的流量，同時傳送一般磁碟區數倍，則可能會發生暫時 [發佈](#) API 限流。此限流可持續長達重複資料刪除時段的持續時間，同時主題會向上擴展以適應增加的流量。

依訊息群組 ID 分發資料

將訊息發佈至 FIFO 主題時，Amazon SNS 會使用每則訊息的訊息群組 ID 的值，做為內部雜湊函數的輸入。雜湊函數的輸出值會決定處理訊息的分割區，指定的分割區可能會處理一或多個訊息群組 IDs。

Note

Amazon SNS 已針對 FIFO 主題分割區之間的項目統一分佈進行最佳化，無論您 AWS 使用的訊息群組 IDs 可以有大量不同值的分割區數量為何。

在 Amazon SNS FIFO 主題上啟用高輸送量

依預設，Amazon SNS FIFO 主題會設定為進行主題層級重複資料刪除，這會由 [FifoThroughputScope](#) 設定為的主題屬性控制，Topic 並且具有更多限制的輸送量配額，請參閱中的 [Amazon SNS 服務配額](#) Amazon Web Services 一般參考。

若要啟用 Amazon SNS FIFO 主題的高輸送量，請將 `FifoThroughputScope` 屬性更新為 `MessageGroup`。此變更可以透過主控台或使用 AWS CLI 和 SDK 完成，也可以在建立主題期間設定，Amazon SNS 建議此變更可提供最佳客戶體驗，並減少您的主題受到調節的機會。

Important

啟用主題的 `FifoThroughputScope` 至 `MessageGroup`，該主題就無法還原為 `Topic` 輸送量。

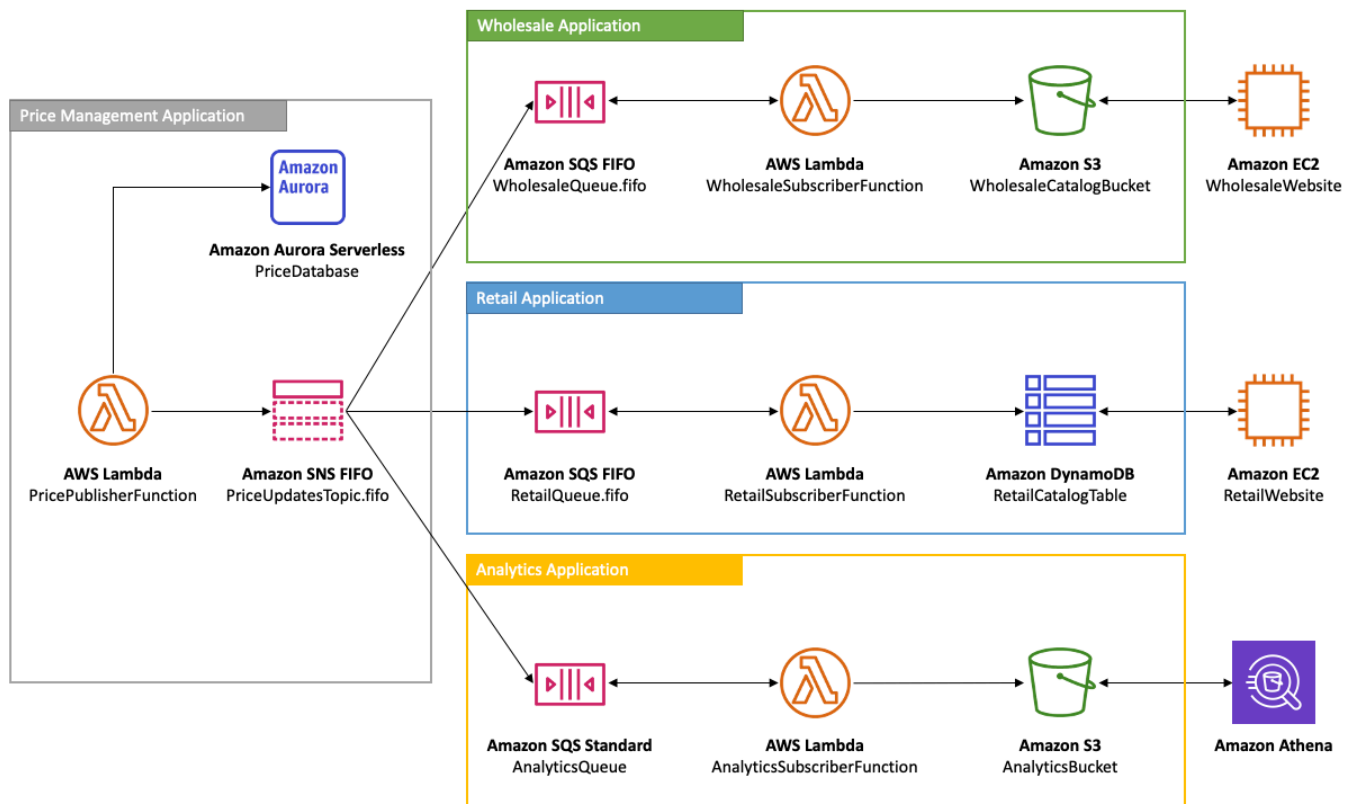
為任何訂閱的 Amazon SQS FIFO 佇列啟用高輸送量模式

發佈至啟用高輸送量的 Amazon SNS FIFO 主題，並訂閱一或多個 Amazon SQS FIFO 佇列時，建議您在 Amazon SQS FIFO 佇列上啟用高輸送量，讓 Amazon SNS FIFO 高輸送量主題順利交付。如需詳細資訊，請參閱 Amazon 簡單佇列服務開發人員指南中的 [FIFO 佇列的高輸送量](#)。

Amazon SNS FIFO 主題範例使用案例

下列範例說明由汽車零件製造商使用 Amazon SNS FIFO 主題和 Amazon SQS 佇列建置的電子商務平台。此平台包含四個無伺服器應用程式：

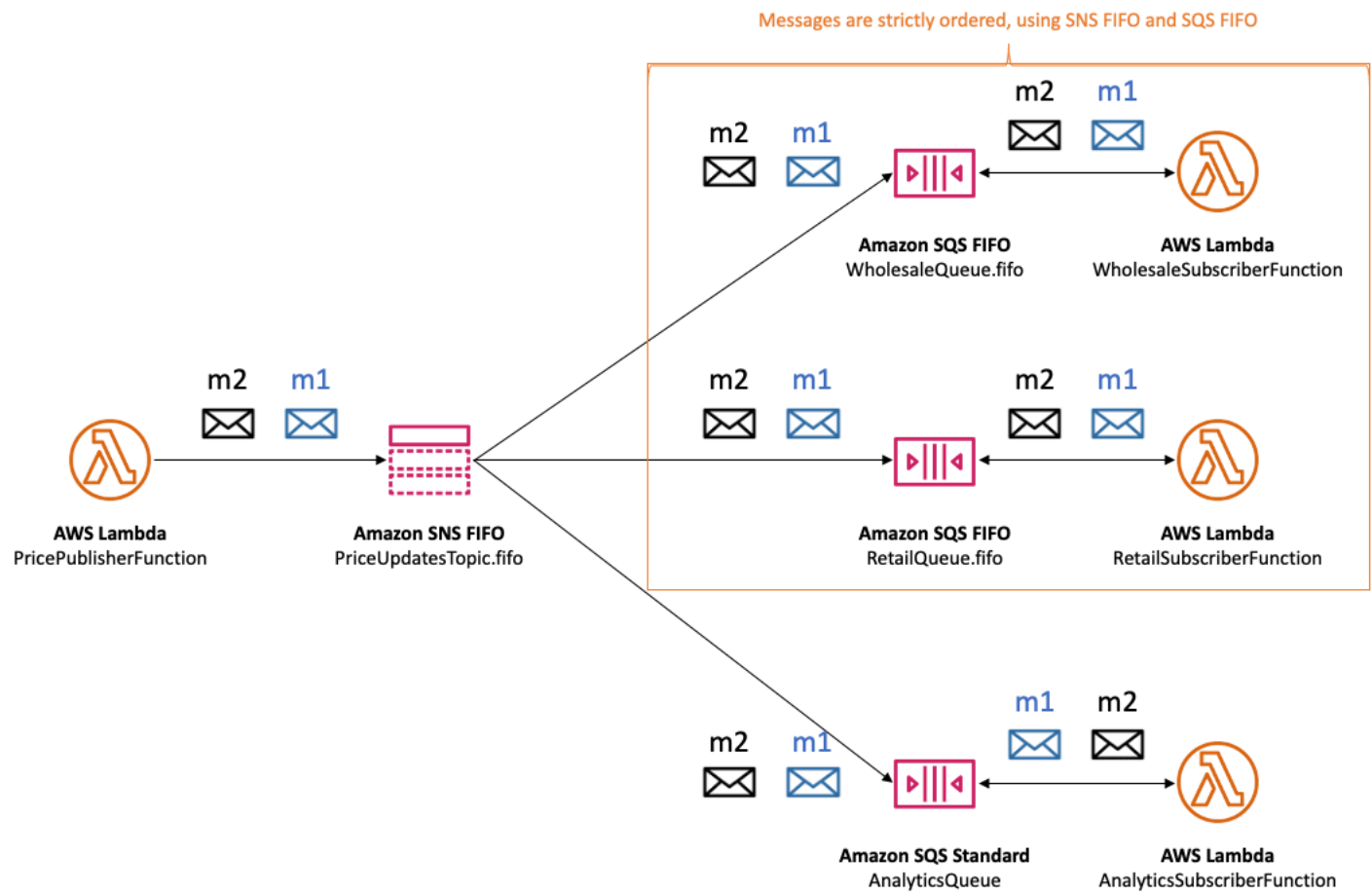
- 庫存管理員使用價格管理應用程式來設定股票中的每個項目的價格。在這家公司，產品價格可以根據貨幣匯率波動，市場需求和銷售策略的變化而變化。價格管理應用程式使用 AWS Lambda 函數，每當價格變化時，將價格更新發布到 Amazon SNS FIFO 主題。
- 批發應用程式提供了一個網站的後端，汽車車身商店和汽車製造商可以大量購買公司的汽車零件。為了獲得價格變化通知，批發應用程式將其 Amazon SQS FIFO 佇列訂閱到價格管理應用程式的 Amazon SNS FIFO 主題。
- 零售應用程式提供另一個網站的後端，車主和汽車改裝愛好者可以為他們的車輛購買個別汽車零件。若要取得價格變更通知，零售應用程式也會將其 Amazon SQS FIFO 佇列訂閱至價格管理應用程式的 Amazon SNS FIFO 主題。
- 一種可彙總價格更新並將其存放到 Amazon S3 儲存貯體的分析應用程式，讓 Amazon Athena 能夠查詢儲存貯體以用於商業智慧 (BI) 目的。為了獲得價格變化通知，分析應用程式將其 Amazon SQS 標準佇列訂閱到價格管理應用程式的 Amazon SNS FIFO 主題。與其他應用程式不同，分析應用程式不需要嚴格排序價格更新。



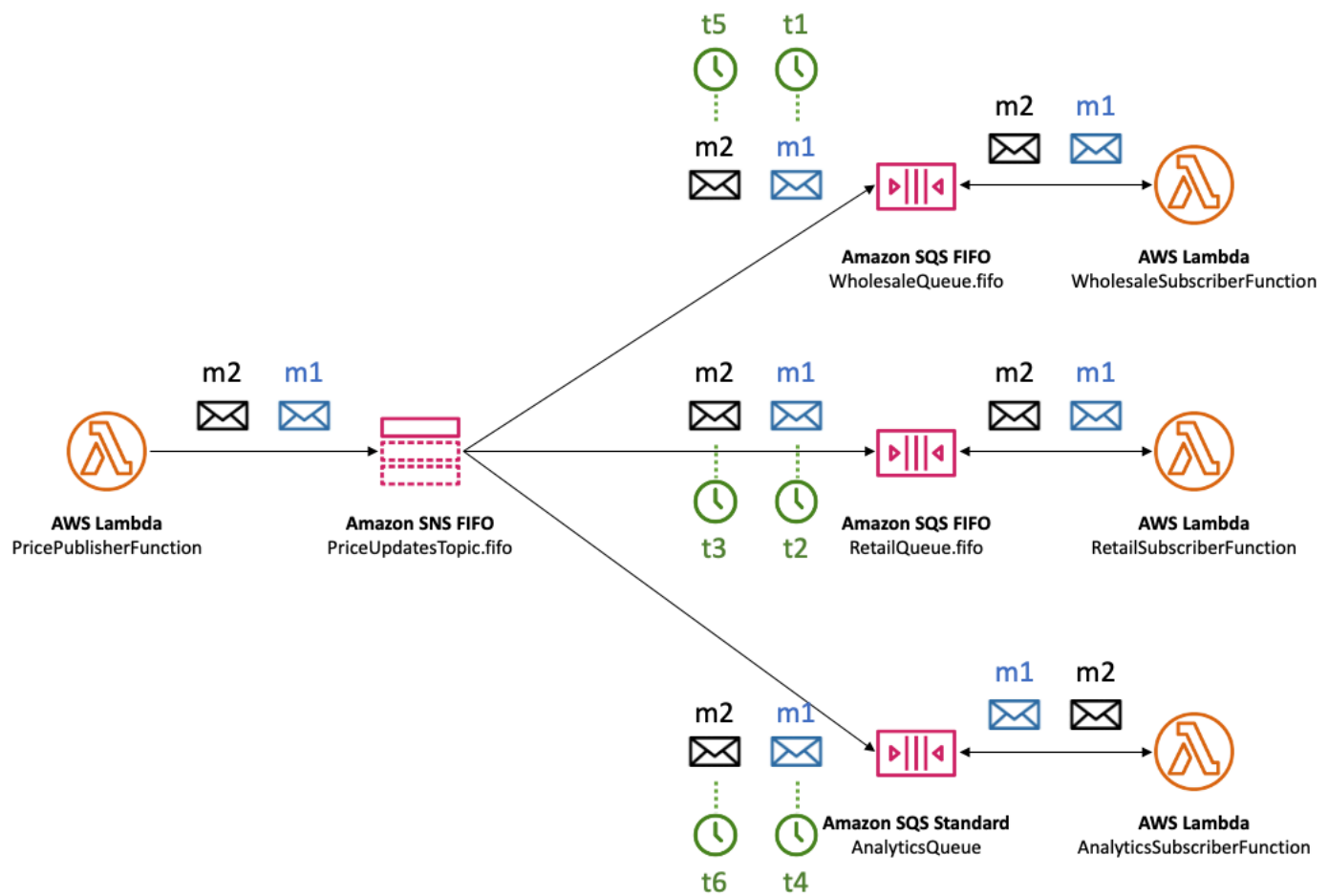
為了讓批發和零售應用程式以正確的順序接收價格更新，價格管理應用程式必須使用嚴格訂購的訊息分配系統。使用 Amazon SNS FIFO 主題和 Amazon SQS FIFO 佇列可以按順序處理訊息，而不會重複。如需詳細資訊，請參閱 [FIFO 主題的 Amazon SNS 訊息排序詳細資訊](#)。如需實作此使用案例的代碼片段，請參閱 [FIFO 主題的 Amazon SNS 程式碼範例](#)。

FIFO 主題的 Amazon SNS 訊息排序詳細資訊

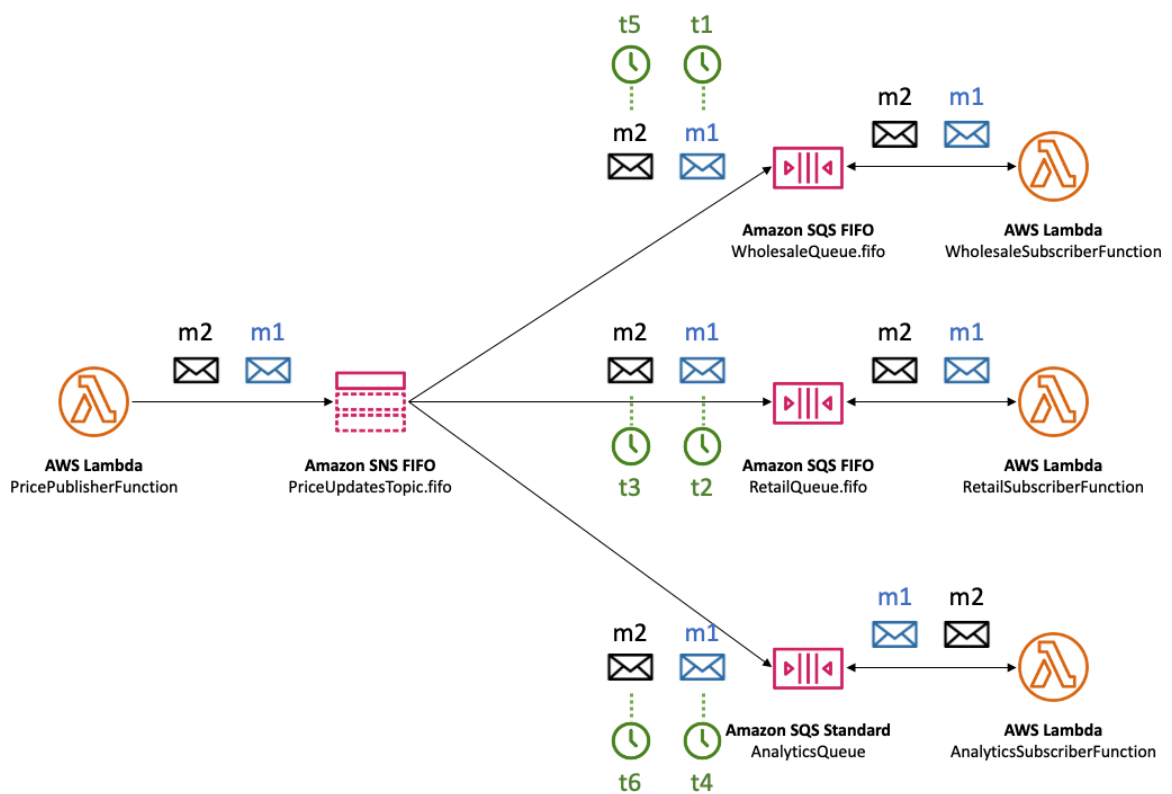
Amazon SNS FIFO 主題一律會依訊息發布到主題的確切順序，將訊息傳送到訂閱的 Amazon SQS 佇列，並僅傳送一次。訂閱 Amazon SQS FIFO 佇列後，佇列的取用者會以訊息傳送到佇列接收訊息的確切順序，且不會重複。不過，如果訂閱了 Amazon SQS 標準佇列，佇列的取用者可能會接收非按順序排列的訊息，而且會接收不止一次。如此可進一步使發布者和訂閱用戶解耦，讓訂閱用戶在訊息消耗和成本最佳化方面具有更大的彈性，如下圖所示，根據 [Amazon SNS FIFO 主題範例使用案例](#)。



請注意，訂閱者沒有隱含的順序。此如下列範例所示：m1 首先交付給批發訂閱用戶，然後交付給零售訂閱用戶，接著是分析訂閱用戶。訊息 m2 首先傳遞給零售訂閱用戶，然後傳遞給批發訂閱用戶，最後傳遞給分析訂閱用戶。雖然這兩個訊息以不同的順序傳遞給訂閱用戶，但訊息順序會保留每個 Amazon SQS FIFO 訂閱用戶。每個訂閱者與任何其他訂閱者彼此獨立。

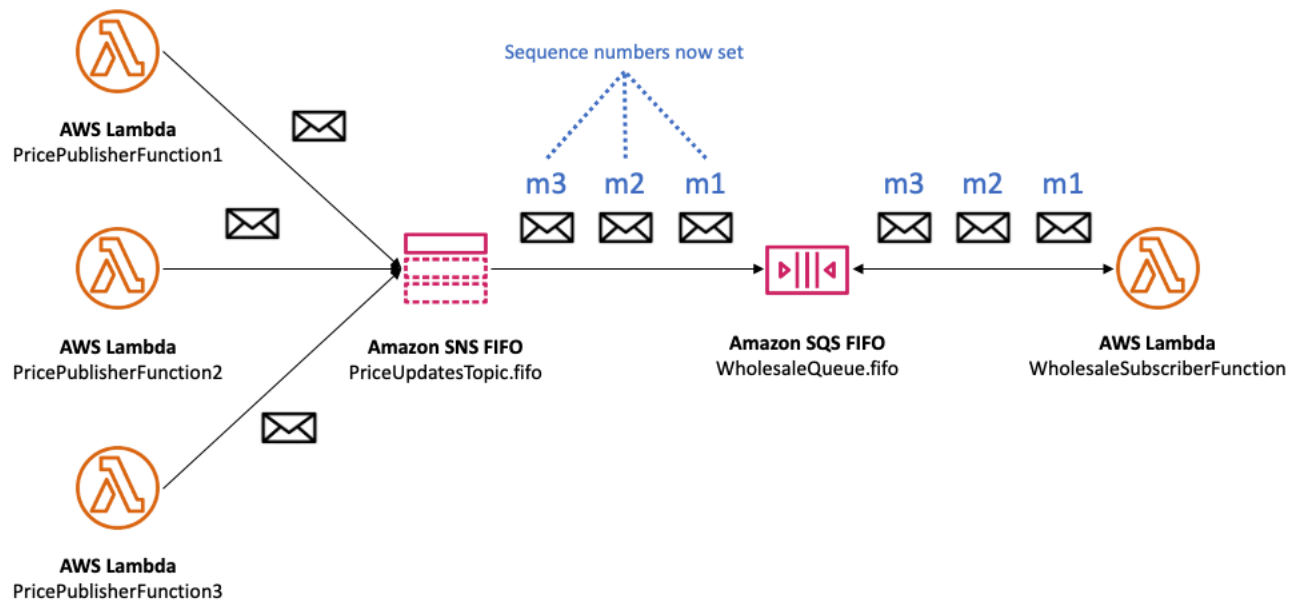


如果 Amazon SQS 佇列訂閱用戶無法連線，則可能會不同步。例如，假設批發應用程式佇列擁有者錯誤地改變了 [Amazon SQS 佇列政策](#)，以防止 Amazon SNS 服務主體將訊息傳遞到佇列。在這種情況下，傳遞到批發佇列的價格更新失敗，而零售和分析佇列的傳遞成功，導致訂閱用戶不同步。當批發應用程式佇列擁有者更正其佇列政策時，Amazon SNS 會繼續將訊息傳送到訂閱佇列。任何在目標佇列設定不正確時發布至主題的訊息都會被捨棄，除非對應的訂閱具有已設定的 [無效字母佇列](#)。



您可以讓多個應用程式 (或同一應用程式中的多個執行緒) 並行將訊息發佈至 SNS FIFO 主題。當您執行這項操作時，您可以有效率地將訊息順序委派給 Amazon SNS 服務。若要判斷已建立的訊息順序，您可以檢查序號。

序號是 Amazon SNS 指派給每封訊息之大型、不連續的數字。序號的長度是 128 位元，而且會在每個訊息群組繼續增加。序號會傳遞至訂閱的 Amazon SQS 佇列，做為訊息內文的一部分。但是，如果您啟用原始訊息交付，則傳遞至 Amazon SQS 佇列的訊息不包含序號或任何其他 Amazon SNS 訊息中繼資料。



Amazon SNS FIFO 主題定義訊息群組內容中的排序。如需詳細資訊，請參閱[FIFO 主題的 Amazon SNS 訊息分組](#)。

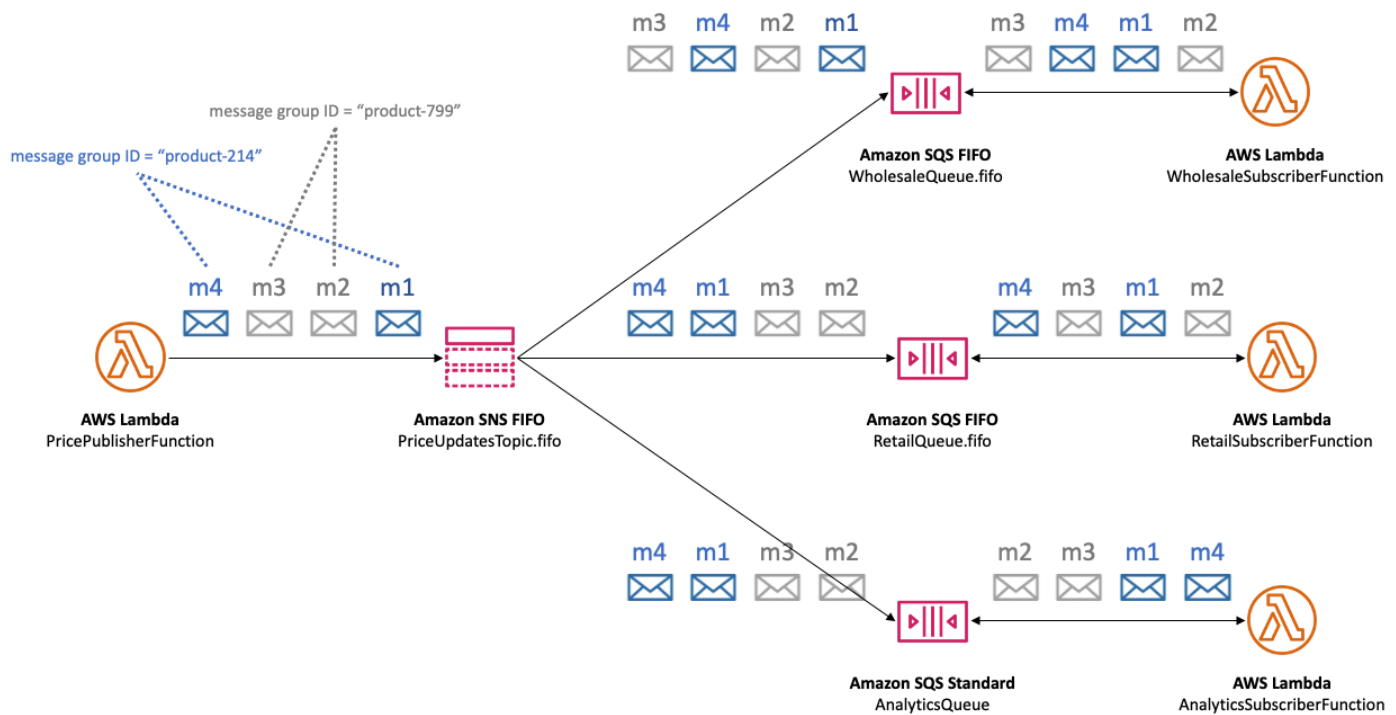
FIFO 主題的 Amazon SNS 訊息分組

屬於相同群組的郵件會依相對於群組的嚴格順序逐一處理。

當您發佈訊息到 Amazon SNS FIFO 主題時，則需要設定訊息群組 ID。群組 ID 是強制權杖，指定訊息屬於特定訊息群組。SNS FIFO 主題會將群組 ID 傳遞至訂閱的 Amazon SQS FIFO 佇列。SNS FIFO 主題或 SQS FIFO 佇列中的群組 ID 數目沒有限制。訊息群組識別碼不會傳遞至 Amazon SQS 標準佇列。

訊息群組與訂閱之間沒有相似性。因此，發佈至任何訊息群組的訊息會傳遞至所有訂閱的佇列，受限於附加至訂閱的任何篩選政策。如需更多詳細資訊，請參閱[FIFO 主題的 Amazon SNS 訊息傳遞](#)及[FIFO 主題的 Amazon SNS 訊息篩選](#)。

在[汽車零件價格管理範例使用案例](#)中，平台上銷售的每個產品都有專用的訊息群組。相同的 Amazon SNS FIFO 主題用於處理所有價格更新。價格更新的順序會保留在單一汽車零件產品的上下文中，但非跨多個產品。下圖顯示此運作方式。請注意，對於訊息群組 ID 為 product-214 的產品，訊息 m1 會在 m4 之前處理。在使用 Amazon SNS FIFO 到 Amazon SQS FIFO 的整個工作流程中都會保留此順序。同樣地，對於訊息群組 ID 為 product-799 的產品，訊息 m2 會在 m3 之前處理。不過，使用 Amazon SQS 標準佇列時，不再保證訊息順序，且訊息群組不存在。所以此產品-214 和產品-799 訊息群組彼此獨立，因此訊息的排序方式之間沒有任何關係。



使用訊息群組 ID 傳遞資料以改善效能

為了最佳化傳遞輸送量，Amazon SNS FIFO 主題會平行傳遞來自不同訊息群組的訊息，同時嚴格維護每個訊息群組內的訊息順序。每個單獨的訊息群組，每秒最多可傳遞 300 則訊息。因此，若單一主題要達到高輸送量，請使用大量不同的訊息群組 ID。透過利用訊息群組的多樣化設定，Amazon SNS FIFO 主題會自動在大量並行分區之間傳遞消息。

Note

Amazon SNS FIFO 主題已針對訊息群組 IDs 之間的訊息統一分佈進行最佳化，無論群組數量為何。AWS 建議您使用大量不同的訊息群組 IDs 來最佳化效能。

當使用高輸送量發至您的 Amazon SNS FIFO 主題，並訂閱一或多個 Amazon SQS FIFO 佇列時，建議您在佇列上啟用高輸送量。如需詳細資訊，請參閱 Amazon 簡單佇列服務開發人員指南中的 [FIFO 佇列的高輸送量](#)。

FIFO 主題的 Amazon SNS 訊息傳遞

Amazon SNS FIFO (先進先出) 主題支援交付到 Amazon SQS 標準和 FIFO 佇列，以便在以近即時方式整合需要資料一致性的分散式應用程式時，為客戶提供彈性和控制能力。

對於需要保留嚴格訊息排序或重複資料刪除的工作負載，Amazon SNS FIFO 主題與訂閱為交付端點的 [Amazon SQS FIFO 佇列](#) 相結合，可在作業和事件順序很重要或無法容忍重複的情況下，提供增強應用程式之間的簡訊功能。

對於容忍全力訂購和至少一次交付的工作負載，訂閱 [Amazon SQS 標準佇列](#) 到 Amazon SNS FIFO 主題，除了可以在不使用 FIFO 的工作負載之間共用佇列之外，還可以降低成本。

Note

若要將訊息從 Amazon SNS FIFO 主題散發到 AWS Lambda 函數，需要額外的步驟。首先，請訂閱該主題的 Amazon SQS FIFO 或標準佇列。然後配置佇列以觸發函數。如需詳細資訊，請參閱 AWS 運算部落格上的 [將 SQS FIFO 作為事件來源](#)。

SNS FIFO 主題無法將訊息傳遞至客戶管理的端點，例如電子郵件地址、行動應用程式、簡訊 (SMS) 的電話號碼或 HTTP (S) 端點。這些端點類型不保證保留嚴格的訊息順序。嘗試將客戶管理的端點訂閱至 SNS FIFO 主題會導致錯誤。

SNS FIFO 主題支援與標準主題相同的訊息篩選功能。如需詳細資訊，請參閱 [FIFO 主題的 Amazon SNS 訊息篩選](#) 與 AWS 運算部落格上的 [運用 Amazon SNS 訊息篩選簡化發佈/訂閱訊息](#) 文章。

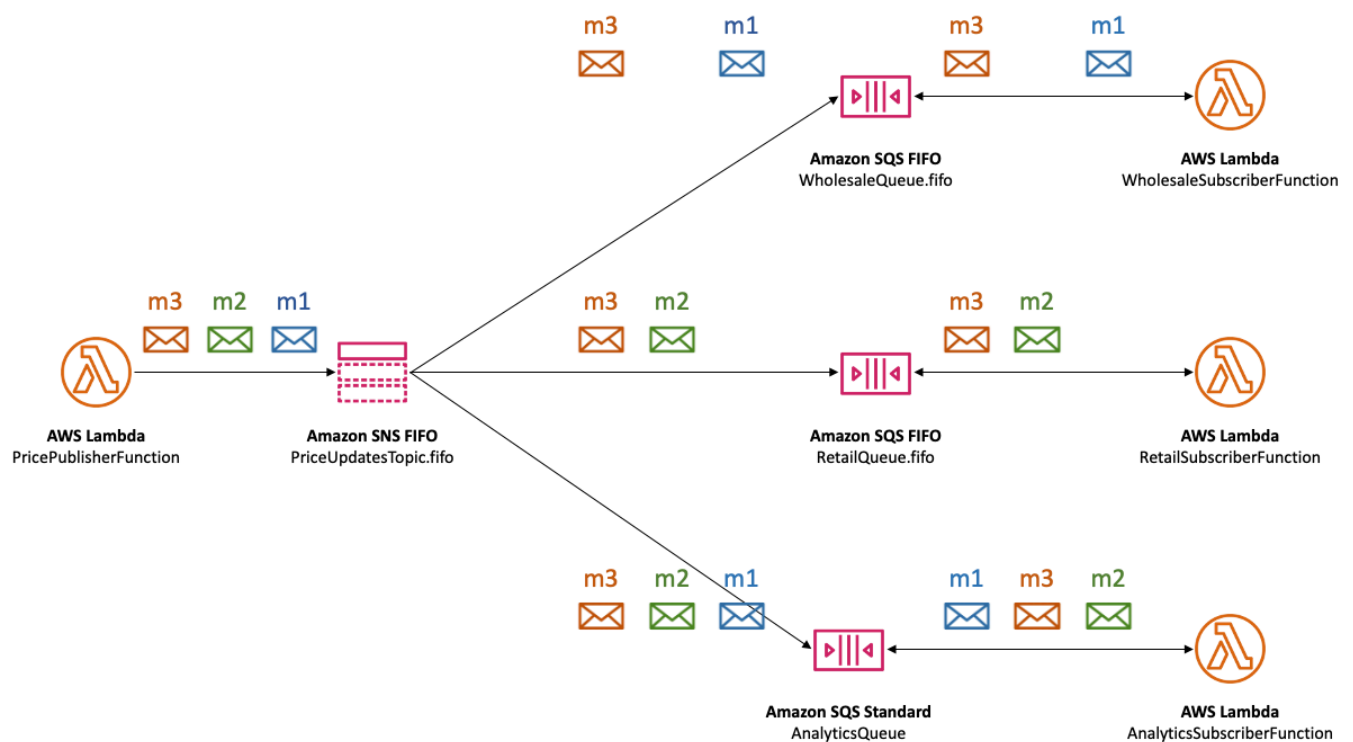
FIFO 主題的 Amazon SNS 訊息篩選

Amazon SNS FIFO 主題支援訊息篩選。透過從發佈者系統卸載訊息路由邏輯，以及從訂閱者系統卸載訊息篩選邏輯，藉此簡化您的架構。

當您將 Amazon SQS FIFO 或標準佇列訂閱至 SNS FIFO 主題時，可以使用訊息篩選來指定訂閱用戶接收訊息子集，而不是所有訊息。每個訂閱者都可以將自己的篩選政策設定為訂閱屬性。根據篩選政策範圍，篩選政策會與傳入訊息屬性或訊息內文相符。如果篩選政策符合，則主題會將郵件的副本傳送給訂閱者。如果沒有相符項目，主題就不會傳送訊息的副本。

在 [汽車零件價格管理範例使用案例](#) 中，假設已設定下列 Amazon SNS 篩選政策，且篩選政策範圍為 `MessageBody`：

- 對於批發佇列，篩選政策 `{"business":["wholesale"]}` 將每個具有名為 `business` 和 `wholesale` 索引鍵的訊息配對在一組值中。在下圖中，訊息 `m1` 裡其中一個索引鍵是值為 `wholesale` 的 `business`。訊息 `m3` 裡其中一個索引鍵是值為 `["wholesale,retail"]` 的 `business`。因此，兩者皆是 `m1` 和 `m3` 符合篩選政策的準則，並且兩個訊息都會傳遞至批發佇列。
- 對於零售佇列，篩選政策 `{"business":["retail"]}` 將每個具有名為 `business` 和 `retail` 索引鍵的訊息配對在一組值中。在圖表中，訊息 `m2` 裡其中一個索引鍵是值為 `retail` 的 `business`。訊息 `m3` 裡其中一個索引鍵是值為 `["wholesale,retail"]` 的 `business`。因此，兩者皆是 `m2` 和 `m3` 符合篩選政策的準則，而且兩個訊息都會傳遞至零售佇列。
- 對於分析佇列，我們希望 Amazon Athena 能夠接收所有記錄，因此不會套用篩選政策。



SNS FIFO 主題支援各種相符運算子，包括屬性字串值、屬性數值和屬性索引鍵。如需詳細資訊，請參閱[Amazon SNS 訊息篩選](#)。

SNS FIFO 主題不會將重複的訊息傳遞至訂閱的端點。如需詳細資訊，請參閱[FIFO 主題的 Amazon SNS 訊息重複資料刪除](#)。

FIFO 主題的 Amazon SNS 訊息重複資料刪除

Amazon SNS FIFO 主題和 Amazon SQS FIFO 佇列支援訊息重複資料刪除功能，只要符合下列條件，就能提供精確一次的訊息傳遞和處理：

- 訂閱的 Amazon SQS FIFO 佇列存在，且具有允許 Amazon SNS 服務主體將訊息傳遞到佇列的權限。
- Amazon SQS FIFO 佇列用戶會處理訊息，並在可見性逾時到期之前將訊息從佇列中刪除。
- Amazon SNS 訂閱主題沒有[訊息篩選](#)。當您設定訊息篩選時，Amazon SNS FIFO 主題支援最多一次傳遞，因為訊息可以根據您的訂閱篩選政策篩選出來。
- 沒有網路中斷會阻止確認訊息傳遞。

Note

當主題屬性 `FifoThroughputScope` 設定為 `時`，訊息重複資料刪除會套用至整個 Amazon SNS FIFO 主題 `Topic`。當主題屬性 `FifoThroughputScope` 設定為 `時` `MessageGroup`，訊息重複資料刪除會套用至每個個別[訊息群組](#)。

當您發布訊息到 Amazon SNS FIFO 主題，訊息必須包含重複資料刪除 ID。此 ID 包含在 Amazon SNS FIFO 主題傳遞至訂閱的 Amazon SQS FIFO 佇列的訊息中。

如果具有特定重複資料刪除 ID 的訊息成功發佈至 Amazon SNS FIFO 主題，則任何以相同重複資料刪除 ID 發佈的訊息都會在五分鐘的重複資料刪除間隔內接受，但不會傳送。Amazon SNS FIFO 主題會在主題屬性設定的重複資料刪除範圍內繼續追蹤訊息重複資料刪除 ID `FifoThroughputScope`，即使訊息已交付至訂閱的端點。

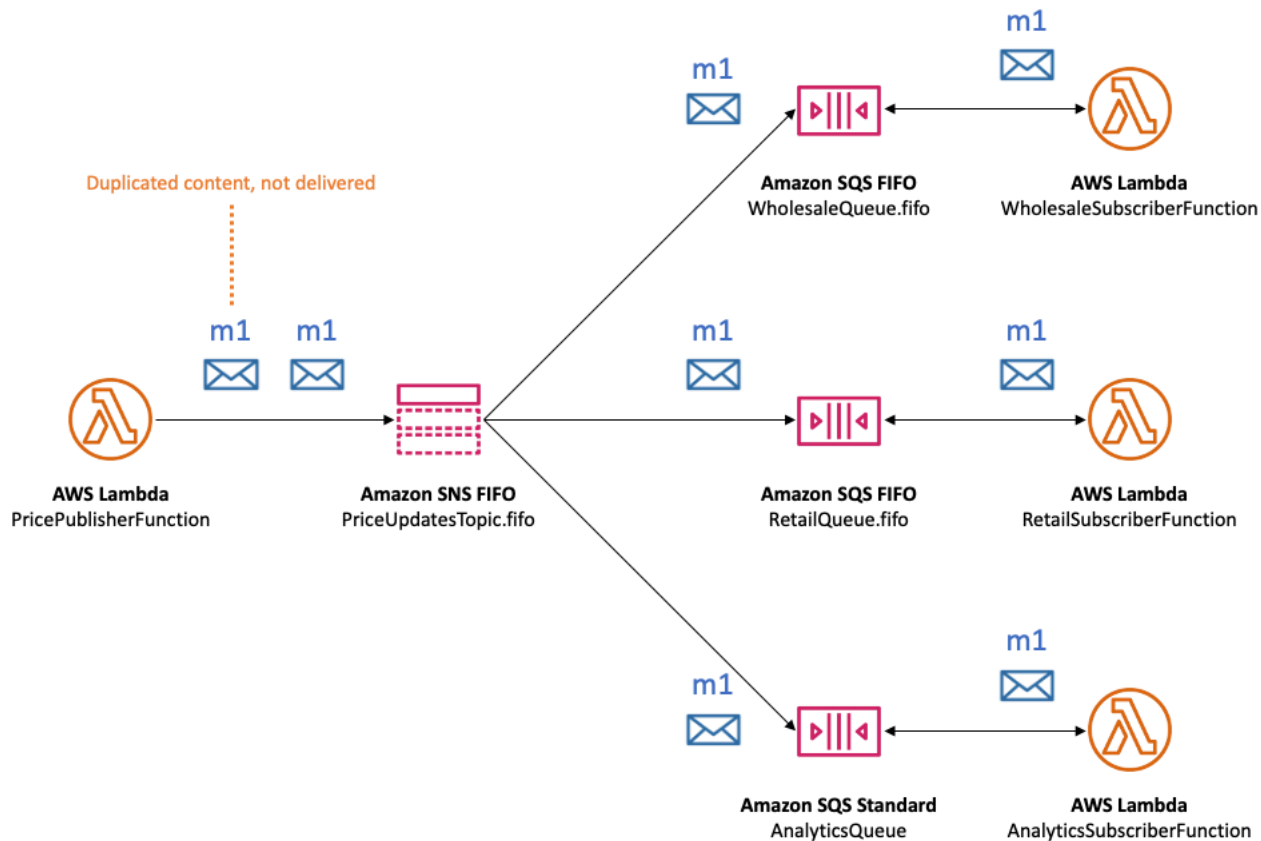
如果確保每封已發布訊息的郵件本文都是唯一的，您可以針對 Amazon SNS FIFO 主題和訂閱的 Amazon SQS FIFO 佇列啟用內容型重複資料刪除功能。Amazon SNS 使用訊息主體產生唯一的雜湊值，用作每封訊息的重複資料刪除 ID，因此您在傳送每封訊息時不需要設定重複資料刪除 ID。

Note

訊息屬性不包含在雜湊計算中。

為 Amazon SNS FIFO 主題啟用內容型重複資料刪除功能，並使用重複資料刪除 ID 發佈訊息時，發佈的重複資料刪除 ID 會覆寫產生的內容型重複資料刪除 ID。

在[汽車零件價格管理示範使用案例](#)中，公司必須為每次價格更新設定一個通用唯一的重複資料刪除 ID。這是因為即使批發和零售的訊息屬性不同，訊息內文也可以是相同的。不過，如果公司將業務類型(批發或零售)新增至訊息主體，則可以在 Amazon SNS FIFO 主題和訂閱的 Amazon SQS FIFO 佇列中啟用內容型重複。



除了訊息排序和重複資料刪除之外，Amazon SNS FIFO 主題還支援具有 AWS KMS 金鑰的訊息伺服器端加密 (SSE)，以及透過具有的 VPC 端點進行訊息隱私權 AWS PrivateLink。如需詳細資訊，請參閱[FIFO 主題的 Amazon SNS 訊息安全性](#)。

FIFO 主題的 Amazon SNS 訊息安全性

您可以使用 [AWS Key Management Service \(AWS KMS\) 客戶主金鑰 \(CMKs\)](#) 啟用 Amazon SNS FIFO 主題和 Amazon SQS FIFO 佇列的加密。

- 您可以建立新的加密 FIFO 主題和佇列，或為現有主題啟用加密。
- 只有訊息內文會加密。訊息屬性、資源中繼資料和資源指標保持未加密。

 Note

將加密新增至現有的 FIFO 主題或佇列並不會加密任何積存的訊息，而且從主題或佇列移除加密會使積存的訊息加密。

SNS FIFO 主題會立即解密訊息，然後再將訊息傳遞至訂閱的端點。SQS FIFO 佇列在將訊息回傳給消費者應用程式之前解密消息。如需詳細資訊，請參閱 AWS 運算部落格上的 [Amazon SNS 資料加密與透過加密發佈到 Amazon SNS 的訊息 AWS KMS](#) 文章。

此外，SNS FIFO 主題和 SQS FIFO 佇列支援由 AWS PrivateLink 驅動的 [介面 VPC 端點](#) 訊息隱私。使用介面端點，您可以從 Amazon Virtual Private Cloud (Amazon VPC) 子網路傳送訊息到 FIFO 主題和佇列，而無需遍歷公用網際網路。此模型會將您的訊息保留在 AWS 基礎設施和網路中，以增強應用程式的整體安全性。使用時 AWS PrivateLink，您不需要設定網際網路閘道、網路位址轉譯 (NAT) 或虛擬私有網路 (VPN)。如需詳細資訊，請參閱 AWS 安全性部落格上的 [使用 VPC 端點保護 Amazon SNS 流量與透過 AWS PrivateLink 保護發佈到 Amazon SNS 的訊息](#) 文章。

SNS FIFO 主題也支援跨可用區域的無效字母佇列和訊息儲存。如需詳細資訊，請參閱 [FIFO 主題的 Amazon SNS 訊息持久性](#)。

FIFO 主題的 Amazon SNS 訊息持久性

Amazon SNS FIFO 主題和 Amazon SQS 佇列是耐用的。這兩種資源類型會在多個可用區域間冗餘儲存郵件，並提供無效字母佇列以處理例外情況。

在 Amazon SNS 中，當因用戶端或伺服器端錯誤而造成 Amazon SNS 主題無法存取訂閱的 Amazon SQS 佇列時，訊息傳遞就會失敗：

- 當 Amazon SNS FIFO 主題有過時的訂閱中繼資料時，就會發生用戶端錯誤。用戶端錯誤的兩個常見原因是 Amazon SQS 佇列擁有者執行下列其中一項作業時：
 - 刪除佇列。
 - 以防止 Amazon SNS 服務主體傳遞訊息的方式變更佇列政策。

Amazon SNS 不會重試傳遞因用戶端錯誤而失敗的訊息。

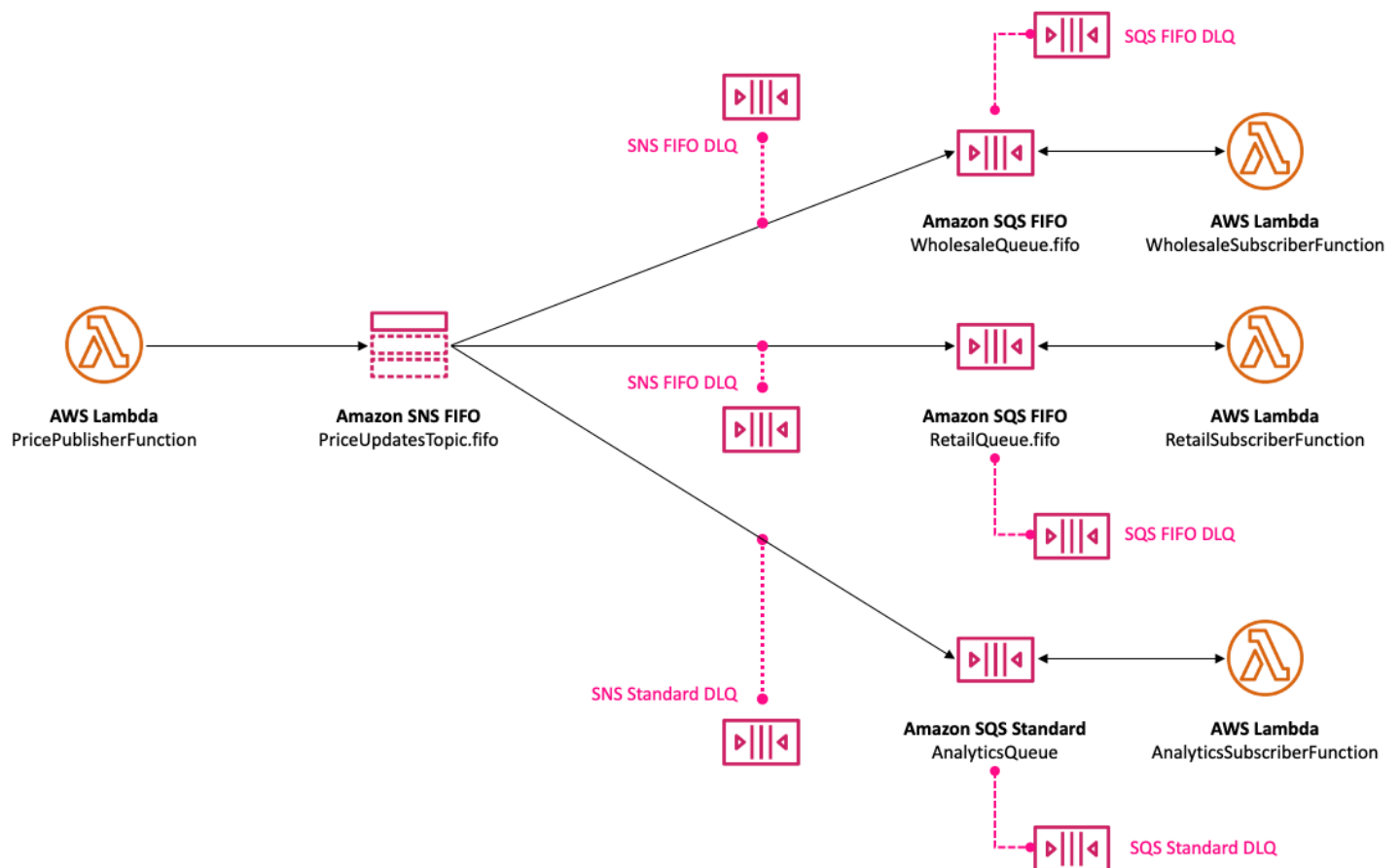
- 伺服器端錯誤可能會發生在下列情況：
 - Amazon SQS 服務無法使用。
 - Amazon SQS 無法處理來自 Amazon SNS 服務的有效請求。

發生伺服器端錯誤時，Amazon SNS FIFO 主題會在 23 天內最多重試失敗的傳遞 100,015 次。如需詳細資訊，請參閱[Amazon SNS 訊息傳遞重試](#)。

對於任何類型的錯誤，Amazon SNS 可以將訊息旁邊傳送到 Amazon SQS 無效字母佇列，因此資料不會遺失。

在 Amazon SQS 中，當消費者應用程式無法接收訊息、處理訊息並從佇列中刪除訊息時，訊息處理會失敗。當接收請求數目上限失敗時，Amazon SQS 可以將訊息旁邊傳送至無效字母佇列，因此資料不會遺失。

在[汽車零件價格管理範例使用案例](#)中，公司可以為每個 Amazon SNS FIFO 主題訂閱指派 Amazon SQS 無效字母佇列 (DLQ)，以及每個訂閱的 Amazon SQS 佇列。這可以保護公司免受任何價格更新損失。



Amazon SNS 訂閱相關聯的無效字母佇列必須是與訂閱佇列類型相同的 Amazon SQS 佇列。例如，Amazon SQS FIFO 佇列的 Amazon SNS FIFO 訂閱必須將 Amazon SQS FIFO 佇列作為無效字母佇列。同樣，Amazon SQS 標準佇列的 Amazon SNS FIFO 訂閱必須將 Amazon SQS 標準佇列作

為其無效字母佇列。如需詳細資訊，請參閱 AWS 運算部落格上的 [Amazon SNS 無效字母佇列 與為 Amazon SNS、Amazon SQS、AWS Lambda 使用 DLQ 設計耐用的無伺服器應用程式](#) 文章。

為了延長耐久性以協助從下游失敗中復原，主題擁有者也可以使用 FIFO 主題來封存訊息長達 365 天。主題訂閱用戶接著就可以對訂閱的端點重播這些訊息，以復原因下游應用程式失敗而遺失的訊息，或複寫現有應用程式的狀態。如需更多資訊，請參閱 [FIFO 主題的 Amazon SNS 訊息封存和重播](#)。

FIFO 主題的 Amazon SNS 訊息封存和重播

什麼是訊息封存與重播功能？

Amazon SNS 提供無程式碼訊息封存和重播功能，專為 FIFO First-In-First-Out 主題而設計。此功能可讓主題擁有者直接在主題封存中存放訊息最多 365 天，並在需要時重新將其傳送給訂閱者。訊息封存和重播對於復原遺失的訊息，以及透過複寫狀態跨區域或系統同步應用程式至關重要。

您可以透過 AWS API、SDK AWS CloudFormation 和 存取此功能 AWS Management Console。

關鍵使用案例

- 訊息復原 – 透過將訊息重播至訂閱用戶的端點，復原因下游應用程式失敗而遺失的訊息。
- 狀態複寫 – 透過從特定時間戳記開始重播訊息，複寫新環境中現有系統的狀態。
- 錯誤修正 – 在中斷期間重新傳送遺漏的訊息，以確保正確處理所有事件。

訊息封存和重播的元件

管理 Amazon SNS FIFO 主題的訊息封存和重播，包括設定保留期、使用 CloudWatch 監控封存的訊息、透過訂閱屬性啟動重播，以及了解修改和啟動重播所需的許可。

訊息封存

- 主題擁有者會啟用封存功能，並設定訊息保留期，最長可達 365 天。如需詳細資訊，請參閱 [FIFO 主題擁有者的 Amazon SNS 訊息封存](#)。
- CloudWatch 指標有助於監控封存的訊息。

訊息重播

- 訂閱者會啟動重播，選取訊息要重新處理到訂閱端點的時間範圍。如需更多資訊，請參閱 [FIFO 主題訂閱者的 Amazon SNS 訊息重播](#)。

- 您可以使用 `ReplayPolicy` 功能，透過訂閱屬性管理重播。

相關許可

- **`SetSubscriptionAttributes`** – 使用訂閱上的 `ReplayPolicy` 屬性來設定或修改重播設定時需要。
- **`Subscribe`** – 連接新訂閱並啟動重播的必要項目。
- **`GetTopicAttributes`** – 允許檢視主題的屬性，但重播啟動主要涉及訂閱管理。

FIFO 主題擁有者的 Amazon SNS 訊息封存

訊息封存可讓您封存發佈至主題之所有訊息的單一副本。您可以對主題啟用訊息封存政策，以便將發佈的訊息儲存在主題內，如此就能對連結至該主題的所有訂閱啟用訊息封存。訊息可封存最短一天，最長 365 天。

設定封存政策需支付額外的費用。如需定價資訊，請參閱 [Amazon SNS 定價](#)。

使用 建立訊息封存政策 AWS Management Console

使用此選項即可使用 AWS Management Console 來建立新的訊息封存政策。

1. 登入 [Amazon SNS 主控台](#)。
2. 選擇主題或建立新主題。若要進一步了解如何建立主題，請參閱 [建立 Amazon SNS 主題](#)。

Note

Amazon SNS 訊息封存與重播功能只適用於應用程式對應用程式 (A2A) FIFO 主題。

3. 在編輯主題頁面上，展開封存政策區段。
4. 啟用封存政策功能，然後輸入要在主題中封存訊息的天數。
5. 選擇 `Save changes` (儲存變更)。

若要檢視、編輯和停用郵件封存主題政策

- 在主題詳細資訊頁面上，保留政策會顯示封存政策的狀態，包括為其設定的天數。選取封存政策索引標籤，以檢視下列訊息封存詳細資訊：

- 狀態 - 套用封存政策時，封存與重播功能狀態會顯示為作用中。當封存政策設定為空的 JSON 物件時，封存與重播功能狀態會顯示為非作用中。
- 訊息保留期間 - 指定的訊息保留天數。
- 封存開始日期 - 訂閱用戶可以開始重播訊息的日期。
- JSON 預覽 - 封存政策的 JSON 預覽。
- (選用) 若要編輯封存政策，請前往主題摘要頁面，然後選擇編輯。
- (選用) 若要停用封存政策，請前往主題摘要頁面，然後選擇編輯。停用封存政策，然後選擇儲存變更。
- (選用) 若要刪除具有封存政策的主題，您必須先如前述停用封存政策。

Important

為了避免意外刪除訊息，您無法刪除具有作用中訊息封存政策的主題。您必須先停用主題的訊息封存政策，才能刪除主題。當您停用訊息封存政策時，Amazon SNS 會刪除所有已封存的訊息。刪除主題時，訂閱也會一併移除，且傳輸中的任何訊息都可能無法傳遞。

使用 API 建立訊息封存政策

若要使用 API 建立訊息封存政策，您需要將屬性 `ArchivePolicy` 新增至您的主題。您可以使用 API 動作 `CreateTopic` 和 `SetTopicAttributes` 設定 `ArchivePolicy`。`ArchivePolicy` 具有單一值 `MessageRetentionPeriod`，代表 Amazon SNS 保留訊息的天數。若要啟用主題的訊息封存，請將 `MessageRetentionPeriod` 設定為大於零的整數值。例如，若要將訊息保留在封存中 30 天，請將 `ArchivePolicy` 設定為：

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

若要停用主題的訊息封存並清除封存，請取消設定 `ArchivePolicy`，如下所示：

```
{}
```

使用 SDK 建立訊息封存政策

若要使用 AWS 開發套件，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 SDK [config](#)和[工具參考指南中的共用](#) 和 [credentials](#) 檔案。 AWS SDKs

下列程式碼範例將示範如何設定 Amazon SNS 主題的 ArchivePolicy，以將發佈至主題的所有訊息保留 30 天。

```
// 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);
```

使用 建立訊息封存政策 AWS CloudFormation

若要使用 建立封存政策，AWS CloudFormation 請參閱AWS CloudFormation 《使用者指南[AWS::SNS::Topic](#)》中的 。

授權存取加密的封存

您必須先完成下列步驟，訂閱用戶才能開始重播來自加密主題的訊息。由於會重播過去的訊息，因此必須為 Amazon SNS 佈建 KMS 金鑰的 Decrypt 存取權，以使用該金鑰來加密封存中的訊息。

1. 當您使用 KMS 金鑰加密訊息並將訊息儲存在主題內時，您必須授權讓 Amazon SNS 能夠透過金鑰政策解密這些訊息。如需更多資訊，請參閱 [對 Amazon SNS 授予解密許可](#)。
2. AWS KMS 為 Amazon SNS 啟用。如需更多資訊，請參閱 [設定 AWS KMS 許可](#)。

⚠ Important

當您為 KMS 金鑰政策新增區段時，請勿變更政策中任何現有的區段。若已啟用主題的加密功能，但 KMS 金鑰已停用或刪除，或者 Amazon SNS 的 KMS 金鑰政策設定不正確，則 Amazon SNS 不會對訂閱用戶重播訊息。

對 Amazon SNS 授予解密許可

若要讓 Amazon SNS 從主題的封存內存取加密的訊息，並對訂閱的端點重播訊息，您必須啟用 Amazon SNS 服務原則來解密這些訊息。

以下是允許 Amazon SNS 服務主體在重播主題內的歷史訊息期間，解密儲存的訊息所需的範例政策。

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

使用 Amazon CloudWatch 監控訊息封存指標

您可以使用下列指標與 Amazon CloudWatch 監控封存的訊息。若要收到工作負載異常的通知並協助避免受到影響，您可以對這些指標設定 Amazon CloudWatch 警示。如需詳細資訊，請參閱[在 Amazon SNS 中記錄和監控](#)。

指標	描述
ApproximateNumberOfMessagesArchived	以 60 分鐘的解析度，為主題擁有者提供主題封存中封存的彙總訊息數。
ApproximateNumberOfBytesArchived	以 60 分鐘的解析度，為主題擁有者提供主題封存中封存的所有訊息的彙總位元組數。

指標	描述
NumberOfMessagesArchiveProcessing	以 1 分鐘解析度，為主題擁有者提供間隔期間儲存至主題封存的訊息數。
NumberOfBytesArchiveProcessing	以 1 分鐘解析度，為主題擁有者提供間隔期間儲存至主題封存的彙總位元組數。

GetTopicAttributes API 具有 BeginningArchiveTime 屬性，代表訂閱用戶可以開始重播的最早時間戳記。以下代表此 API 動作的範例回應：

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

FIFO 主題訂閱者的 Amazon SNS 訊息重播

Amazon SNS 重播可讓主題訂閱者從主題資料存放區擷取封存的訊息，並重新傳遞至訂閱的端點。

- 訂閱建立後，即可立即重播訊息。
- 重播的訊息會保留Timestamp與原始訊息相同的內容 MessageId和。
- 訊息包含Replayed屬性，以表示它是重播的訊息。
- 若要僅重播特定訊息，請將篩選政策套用至您的訂閱。

如需更多篩選訊息的資訊，請參閱 [篩選重播的訊息](#)。

使用 建立訊息重播政策 AWS Management Console

使用此選項即可使用 AWS Management Console來建立新的重播政策。

1. 登入 [Amazon SNS 主控台](#)。
2. 選擇主題訂閱或建立新的主題訂閱。若要進一步了解如何建立訂閱，請參閱 [建立 Amazon SNS 主題的訂閱](#)。

3. 若要啟動訊息重播，請前往重播下拉式清單，然後選擇開始重新播放。
4. 從重新播放時間互動視窗進行下列選擇：
 - a. 選擇重播開始日期和時間 - 選擇要開始重播封存訊息的日期 (YYYY/MM/DD 格式) 和時間 (24 小時 hh:mm:ss 格式)。開始時間應晚於大約的封存時間開頭。
 - b. (選用) 選擇重播結束日期和時間 - 選擇要停止重播封存訊息的日期 (YYYY/MM/DD 格式) 和時間 (24 小時 hh:mm:ss 格式)。
 - c. 選擇開始重新播放。
5. (選用) 若要停止訊息重播，請前往訂閱詳細資訊頁面，然後從重新播放下拉式清單中選擇停止重新播放。
6. (選用) 若要使用 CloudWatch 在此工作流程內監控訊息重播指標，請參閱 [使用 Amazon CloudWatch 監控訊息重播指標](#)。

若要檢視和編輯訊息重播政策

您可以從訂閱詳細資訊頁面執行下列動作：

- 若要檢視訊息重播狀態，重新播放狀態欄位會顯示下列值：
 - 已完成 - 重播已成功重新傳遞所有訊息，且現在正在傳遞新發佈的訊息。
 - 進行中 - 重播目前正在重播選取的訊息。
 - 失敗 - 重播無法完成。
 - 待定 - 重播啟動時的預設狀態。
- (選用) 若要修改訊息重播政策，請前往訂閱詳細資訊頁面，然後從重新播放下拉式清單中選擇開始重新播放。開始重播將會取代現有的重播。

使用 API 新增重播政策至訂閱

若要重播封存的訊息，請使用屬性 `ReplayPolicy`。`ReplayPolicy` 可搭配 `Subscribe` 和 `SetSubscriptionAttributes` API 動作使用。此政策包括下列值：

- **StartingPoint** (必要) - 指出從何處開始重播訊息。
- **EndingPoint** (選用) - 指出何時停止重播訊息。如果省略 `EndingPoint`，則重播將繼續進行，直到趕上目前時間為止。
- **PointType** (必要) - 設定起點和終點的類型。目前 `PointType` 支援的值為 `Timestamp`。

例如，若要在 2023 年 10 月 1 日從下游失敗復原，並且重新傳送 2 小時期間的所有訊息，請使用 `SetSubscriptionAttributes` API 動作來設定 `ReplayPolicy`，如下所示：

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

若要重播 2023 年 10 月 1 日起傳送至該主題的所有訊息，並繼續接收所有新發佈至您的主題的訊息，請使用 `SetSubscriptionAttributes` API 動作在您的訂閱上設定 `ReplayPolicy`，如下所示：

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

為了確認訊息是否已重播，會將布林值屬性 `Replayed` 新增至每一則重播的訊息。

使用 SDK 新增重播政策至訂閱

若要使用 AWS 開發套件，您必須使用登入資料進行設定。如需詳細資訊，請參閱 SDK [config 和工具參考指南中的共用和 credentials 檔案](#)。AWS SDKs

下列程式碼範例示範如何在訂閱上設定 `ReplayPolicy`，以便在 2023 年 10 月 1 日從 Amazon SNS FIFO 主題的封存重新傳遞 2 小時時段的訊息。

```
// 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);
```

了解 EndingPoint

當您ReplayPolicy將套用至 Amazon SNS 訂閱時，該EndingPoint值為選用。如果未EndingPoint提供任何，則重播會從指定的開始StartingPoint，並持續到目前為止，包括處理任何新發佈的訊息。趕上進度後，訂閱將做為一般訂閱運作，並在發佈新訊息時接收新訊息。

如果指定 EndingPoint，服務將從重播訊息，StartingPoint直到 EndingPoint，然後停止。此動作會有效地暫停訂閱。當訂閱暫停時，新發佈的訊息將不會傳遞到訂閱的端點。

若要繼續訊息傳遞，請套用新的 ReplayPolicy而不提供 EndingPoint，並將 StartingPoint設定為要繼續接收訊息的所需時間點。例如，若要從先前重播完成的位置繼續訂閱，請將新 StartingPoint 設定為先前提供的 EndingPoint。

篩選重播的訊息

Amazon SNS 訊息篩選可讓您控制 Amazon SNS 對訂閱用戶端點重播的訊息。訊息篩選和訊息封存同時啟用時，Amazon SNS 會先從主題的資料存放區擷取訊息，再對訂閱的 FilterPolicy 套用訊息。若有相符項目，訊息就會傳遞至訂閱的端點，否則訊息會被篩選掉。如需詳細資訊，請參閱[Amazon SNS 訂閱篩選政策](#)。

使用 Amazon CloudWatch 監控訊息重播指標

您可以使用下列指標與 Amazon CloudWatch 監控重播訊息。若要收到工作負載異常的通知並協助避免受到影響，您可以對這些指標設定 Amazon CloudWatch 警示。如需詳細資訊，請參閱[在 Amazon SNS 中記錄和監控](#)。

指標	描述
NumberOfReplayedNotificationsDelivered	以 1 分鐘解析度，為訂閱用戶提供從主題封存重播的彙總訊息數。
NumberOfReplayedNotificationsFailed	以 1 分鐘解析度，為訂閱用戶提供從主題封存傳遞失敗的重播彙總訊息數。

FIFO 主題的 Amazon SNS 程式碼範例

使用下列程式碼範例，將[自動組件價格管理範例使用案例](#)與 Amazon SNS FIFO 主題和 Amazon SQS FIFO 佇列或標準佇列整合。

使用 AWS SDK

使用 AWS SDK，您可以透過將其 `FifoTopic` 屬性設定為 `true` 來建立 Amazon SNS FIFO 主題。您可以透過將其 `FifoQueue` 屬性設定為 `true` 建立一個 Amazon SQS FIFO 佇列。此外，您必須將 `.fifo` 後綴新增到每個 FIFO 資源的名稱。建立 FIFO 主題或佇列之後，您無法將其轉換為標準主題或佇列。

下列代碼範例會建立這些 FIFO 與標準佇列資源：

- 分配價格更新的 Amazon SNS FIFO 主題
- Amazon SQS FIFO 佇列提供這些更新給批發和零售應用程式
- 用於儲存記錄的分析應用程式的 Amazon SQS 標準佇列，可查詢商業智慧 (BI)
- 將三個佇列連線到主題的 Amazon SNS FIFO 訂閱

此範例設定訂閱的[篩選政策](#)。如果您藉由將訊息發佈到主題來測試範例，請確定發佈具有 `business` 屬性的訊息。指定 `retail` 或 `wholesale` 做為屬性值。否則，會篩選出訊息，而不會傳遞至訂閱佇列。如需詳細資訊，請參閱[FIFO 主題的 Amazon SNS 訊息篩選](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

此範例

- 建立一個 Amazon SNS FIFO 主題、兩個 Amazon SQS FIFO 佇列和一個標準佇列。
- 訂閱佇列到主題並向該主題發布訊息。

[測試](#)可驗證每個佇列的訊息接收狀況。[完整範例](#)也會顯示存取政策的新增，並在最後刪除資源。

```
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);
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
 - [CreateTopic](#)
 - [發布](#)
 - [Subscribe](#)

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

建立 Amazon SNS FIFO 主題，訂閱 Amazon SQS FIFO 和標準佇列到該主題，並向該主題發布訊息。

```
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

```

- 如需 API 的詳細資訊，請參閱 AWS SDK for Python (Boto3) API Reference 中的下列主題。
- [CreateTopic](#)
- [發布](#)
- [Subscribe](#)

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

建立 FIFO 主題、將 Amazon SQS FIFO 佇列訂閱至主題，然後將訊息發佈至 Amazon SNS 主題。

```
" 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_snsstopiclimitexcdex.
        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.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的下列主題。
 - [CreateTopic](#)
 - [發布](#)
 - [Subscribe](#)

接收來自 FIFO 訂閱的訊息

您現在可以在三個已訂閱的應用程式中接收價格更新。如所示 [the section called “FIFO 主題使用案例”](#)，每個取用者應用程式的進入點都是 Amazon SQS 佇列，其對應的 AWS Lambda 函數可以自動輪詢。當 Amazon SQS 佇列是 Lambda 函數的事件來源時，Lambda 會視需要調整其輪詢器機群，以有效率地使用訊息。

如需詳細資訊，請參閱《AWS Lambda 開發人員指南》中的 [搭配使用 AWS Lambda 與 Amazon SQS](#)。如需撰寫自己佇列輪詢程式的相關資訊，請參閱 Amazon Simple Queue Service 開發者指南中的 [Amazon SQS 標準和 FIFO 佇列的建議](#) 和 Amazon Simple Queue Service API 參考中的 [ReceiveMessage](#)。

使用 AWS CloudFormation

AWS CloudFormation 可讓您使用範本檔案，將 AWS 資源集合建立並設定為單一單位。本節含有可建立下列項目的範例範本：

- 分配價格更新的 Amazon SNS FIFO 主題
- Amazon SQS FIFO 佇列提供這些更新給批發和零售應用程式
- 用於儲存記錄的分析應用程式的 Amazon SQS 標準佇列，可查詢商業智慧 (BI)
- 將三個佇列連線到主題的 Amazon SNS FIFO 訂閱

- 指定訂閱者應用程式只接收所需價格更新的[篩選政策](#)

Note

如果您藉由將訊息發佈至主題來測試此程式碼範例，請確定發佈具有 `business` 屬性的訊息。指定 `retail` 或 `wholesale` 做為屬性值。否則，會篩選出訊息，而不會傳遞至訂閱佇列。

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

如需使用 AWS CloudFormation 範本部署 AWS 資源的詳細資訊，請參閱AWS CloudFormation 《使用者指南》[中的入門](#)。

Amazon SNS 訊息篩選

根據預設，Amazon SNS 主題訂閱者會接收發佈到主題的每個訊息。若要僅接收一部分的訊息，訂閱者必須將篩選政策指派給主題訂閱。

篩選政策是包含屬性的 JSON 物件，這些屬性會定義訂閱者接收的訊息。Amazon SNS 支援對訊息屬性或內文採取行動的政策，該政策係根據您為訂閱設定的篩選政策範圍。訊息內文的篩選政策假定訊息承載是格式正確的 JSON 物件。

如果訂閱沒有篩選政策，訂閱者會收到發佈到主題的每個訊息。當您發佈訊息至具有篩選政策的主題時，Amazon SNS 會針對每一個主題訂閱，比較訊息屬性或內文與篩選政策中的屬性。如果所有訊息屬性或訊息內文屬性都符合篩選條件政策中指定的條件，Amazon SNS 會將訊息傳送給訂閱者。否則，Amazon SNS 不會傳送訊息給訂閱者。

如需詳細資訊，請參閱 [Filter Messages Published to Topics](#) (篩選發佈至主題的訊息)。

Amazon SNS 訂閱篩選政策範圍

FilterPolicyScope 訂閱屬性可讓您透過設定下列其中一個值來定義篩選範圍：

- MessageAttributes – 將篩選條件政策套用至訊息屬性（預設設定）。
- MessageBody – 將篩選條件政策套用至訊息內文。

Note

如果未針對現有篩選政策定義篩選政策範圍，則範圍預設為 MessageAttributes。

Amazon SNS 訂閱篩選政策

訂閱篩選政策可讓您指定屬性名稱，並將值清單指派給每個屬性名稱。如需詳細資訊，請參閱[Amazon SNS 訊息篩選](#)。

當 Amazon SNS 根據訂閱篩選政策來評估訊息屬性或訊息內文屬性時，它會忽略政策中未指定的部份。

 Important

AWS IAM 和 Amazon SNS 等服務會使用稱為最終一致性的分散式運算模型。對訂閱篩選原則進行新增或變更，最多需要 15 分鐘才能完全生效。

在下列情況下，訂閱會接受訊息：

- 當篩選政策範圍為 `MessageAttributes`，篩選政策中每個屬性名稱都符合訊息屬性名稱。針對篩選政策中的每個相符屬性名稱，至少有一個屬性值符合訊息屬性值。
- 當篩選政策範圍為 `MessageBody`，篩選政策中每個屬性名稱都符合訊息內文屬性名稱。針對篩選政策中的每個相符屬性名稱，至少有一個屬性值符合訊息內文屬性值。

Amazon SNS 目前支援下列篩選運算子：

- [AND 邏輯](#)
- [OR 邏輯](#)
- [OR 運算子](#)
- [索引鍵比對](#)
- [數值完全相符](#)
- [支援任何數值，但不會比對](#)
- [數值範圍比對](#)
- [字串值完全相符](#)
- [支援任何字串值，但不會比對](#)
- [支援任何使用前綴的字串值，但不會比對](#)
- [字串值相同，但忽略大小寫](#)
- [字串值 IP 地址比對](#)
- [字串值前綴比對](#)
- [字串值後綴比對](#)

Amazon SNS 範例篩選政策

下列範例顯示處理客戶交易的 Amazon SNS 主題所傳送的訊息酬載。

第一個範例包含 MessageAttributes 欄位，其中包含描述交易的屬性：

- 客戶的利益
- 商店名稱
- 事件狀態
- 購買價格 (以美元為單位)

由於此訊息包含 MessageAttributes 欄位，所以只要訂閱中的 FilterPolicyScope 皆設為 MessageAttributes，設為 FilterPolicy 的主題訂閱皆可選擇性地接受或拒絕訊息。如需將屬性套用到訊息的資訊，請參閱[Amazon SNS 訊息屬性](#)。

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


下列範例顯示 Message 欄位中的相同屬性，也稱為訊息承載或訊息內文。只要訂閱中的 FilterPolicyScope 皆設為 MessageBody，任何包含 FilterPolicy 的主題訂閱皆可選擇性地接受或拒絕訊息。

```
{
  "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"
}
```

以下篩選政策根據訊息的屬性名稱和值來接受或拒絕訊息。

接受範例訊息政策

以下訂閱篩選政策中的屬性符合指派給範例訊息的屬性。請注意，不論其設為 MessageAttributes 或 MessageBody，相同的篩選政策皆適用於 FilterPolicyScope。每個訂閱者會根據從主題接收到的訊息，選擇其篩選範圍。

如果此政策中的任何單一屬性不符指派給該訊息的屬性，政策將拒絕訊息。

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

拒絕範例訊息政策

以下訂閱篩選政策在其屬性和指派給範例訊息的屬性之間有諸多不符。例如，由於 `encrypted` 屬性名稱沒有出現在訊息屬性中，不論指派的值為何，此政策屬性都會導致訊息遭到拒絕。

如果有任何不符，政策會拒絕訊息。

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

Amazon SNS 中的篩選政策限制

當您在 Amazon SNS 中設定篩選政策時，請注意幾個重要的規則。這些規則有助於確保有效套用篩選政策，同時維持系統效能和相容性。

常見政策限制條件

在 Amazon SNS 中設定篩選政策時，請遵循下列重要規則，以確保它們有效運作，同時維持系統效能和相容性：

- 字串比對 – 對於篩選政策中的字串比對，比較區分大小寫。
- 數值比對 – 對於數值比對，值的範圍可以從 -10^9 到 10^9 (-10 億到 10 億)，在小數點後有五位數的準確性。
- 篩選政策複雜性 – 篩選政策中值的總組合不得超過 150。若要計算總組合，請將篩選政策中每個陣列中的值數量相乘。
- 金鑰數量限制 – 篩選政策最多可有五個金鑰。

其他考量事項

- 篩選政策的 JSON 可以包含下列項目：
 - 用引號括住的字串

- 數字
- 關鍵字 true、false 和 null (不含引號)
- 使用 Amazon SNS API 時，您必須將篩選條件政策的 JSON 做為有效的 UTF-8 字串傳遞。
- 篩選條件政策的大小上限為 256 KB。
- 根據預設，每個主題最多可有 200 個篩選政策，每個 AWS 帳戶最多可有 10,000 個篩選政策。

此政策限制不會阻止使用 [Subscribe](#) API 建立 Amazon SQS 佇列訂閱。但是，當您在 [Subscribe](#) API 呼叫 (或 [SetSubscriptionAttributes](#) API 呼叫) 中連接篩選條件政策時，它將會失敗。

若要提升配額，請使用 [AWS Service Quotas](#)。

以屬性為基礎的篩選政策限制條件

以屬性為基礎的篩選是預設選項。[FilterPolicyScope](#) 在訂閱中設定為 MessageAttributes。

- Amazon SNS 不接受以屬性為基礎的巢狀篩選政策。
- Amazon SNS 只將政策屬性與具有下列資料類型的訊息屬性做比較：
 - String
 - String.Array

Important

在 Amazon SNS 中使用屬性型篩選時，您必須雙逸出某些特殊字元，特別是：

- 雙引號 (")
- 反斜線 (\)

如果無法雙逸出這些字元，將導致篩選條件政策與已發佈訊息的屬性不相符，且通知將不會傳送。

其他考量事項

- 不建議在陣列中傳遞物件，因為它可能會因為屬性型篩選不支援的巢狀而產生非預期的結果。為巢狀政策使用以承載為基礎的篩選條件。

- Number 數值屬性值支援。
- Amazon SNS 會使用二進位資料類型忽略訊息屬性。

複雜性政策範例：

在下列政策範例中，第一個索引鍵有三個相符運算子，第二個則有一個相符運算子，第三個則有兩個相符運算子。

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

總組合的計算方式為篩選政策中每個索引鍵的相符運算子數目乘積：

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

以承載為基礎的篩選政策限制條件

若要從以屬性為基礎的篩選 (預設) 切換為以承載為基礎的篩選，您必須在訂閱中將 [FilterPolicyScope](#) 設為 `MessageBody`。

- Amazon SNS 接受以承載為基礎的巢狀篩選政策。
- 對於巢狀政策，只有分葉索引鍵會計入五個索引鍵限制。

金鑰限制的政策範例：

在下列政策範例中：

- 有兩種分葉索引鍵：key_c和 key_e。
- key_c 有四個巢狀層級為 3 的相符運算子，key_e有三個巢狀層級為 2 的相符運算子。

```
{
```

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

總組合的計算方式為篩選條件政策中每個索引鍵的相符運算子數量和巢狀層級的乘積：

```
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
```

AND/OR 邏輯

在篩選政策中使用 AND/OR 邏輯，以符合 Amazon SNS 中的訊息屬性或訊息內文屬性。這可實現更精確且靈活的訊息篩選。

AND 邏輯

您可以使用多個屬性名稱來套用 AND 邏輯。

舉例下列政策：

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

它符合 `customer_interests` 的值設為 `rugby` 且 `price_usd` 的值設為大於 100 數字的任何訊息屬性或訊息內文屬性。

Note

您無法將 AND 邏輯套用至相同屬性的值。

OR 邏輯

您可以將多個值指派給屬性名稱來套用 OR 邏輯。

舉例下列政策：

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

它符合 `customer_interests` 的值設為 `rugby`、`football` 或 `baseball` 的任何訊息屬性或訊息內文屬性。

OR 運算子

您可以使用 `"$or"` 運算子明確定義篩選政策，以表示政策中多個屬性之間的 OR 關係。

Amazon SNS 只會在政策符合下列所有條件時辨識 `"$or"` 關係。當所有這些條件都不符合時，`"$or"` 會被視為一般屬性名稱，與政策中的任何其他字串相同。

- 例如 `"$or" : []`，規則中有一個 `"$or"` 欄位屬性，其後面有一個陣列。
- `"$or"` 陣列中至少有兩個物件：`"$or": [{}, {}]`。
- `"$or"` 陣列中沒有任何物件具有保留關鍵字的欄位名稱。

否則，`"$or"` 會被視為一般屬性名稱，與政策中的其他字串相同。

下列政策不會剖析為 OR 關係，因為數字和前置詞是保留的關鍵字。

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

OR 運算子範例

標準OR：

```
{
  "source": [ "aws.cloudwatch" ],
  "$or": [
    { "metricName": [ "CPUUtilization" ] },
```

```
{ "namespace": [ "AWS/EC2" ] }  
]  
}
```

此政策的篩選邏輯為：

```
"source" && ("metricName" || "namespace")
```

它符合以下任一組訊息屬性：

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},  
"metricName": {"Type": "String", "Value": "CPUUtilization"}
```

或

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},  
"namespace": {"Type": "String", "Value": "AWS/EC2"}
```

它也符合以下任一訊息內文：

```
{  
  "source": "aws.cloudwatch",  
  "metricName": "CPUUtilization"  
}
```

或

```
{  
  "source": "aws.cloudwatch",  
  "namespace": "AWS/EC2"  
}
```

包含 **OR** 關係的政策限制

舉例下列政策：

```
{  
  "source": [ "aws.cloudwatch" ],  
  "$or": [  
    { "metricName": [ "CPUUtilization", "ReadLatency" ] },  
    { "metricName": [ "CPUUtilization", "ReadLatency" ] },  
  ],  
}
```

```
{
  "metricType": [ "MetricType" ] ,
  "$or" : [
    { "metricId": [ 1234, 4321 ] },
    { "spaceId": [ 1000, 2000, 3000 ] }
  ]
}
```

此政策的邏輯也可以簡化為：

```
("source" AND "metricName")
OR
("source" AND "metricType" AND "metricId")
OR
("source" AND "metricType" AND "spaceId")
```

具有 OR 關係的政策，複雜度計算可以簡化為每個 OR 陳述式的組合複雜性總和。

組合總計的計算如下：

```
(source * metricName) + (source * metricType * metricId) + (source * metricType *
spaceId)
= (1 * 2) + (1 * 1 * 2) + (1 * 1 * 3)
= 7
```

source 有一個值、metricName 有兩個值、metricType 有一個值、metricId 有兩個值和 spaceId 有三個值。

請考慮下列巢狀篩選政策：

```
{
  "$or": [
    { "metricName": [ "CPUUtilization", "ReadLatency" ] },
    { "namespace": [ "AWS/EC2", "AWS/ES" ] }
  ],
  "detail" : {
    "scope" : [ "Service" ],
    "$or": [
      { "source": [ "aws.cloudwatch" ] },
      { "type": [ "CloudWatch Alarm State Change" ] }
    ]
  }
}
```



```
    ]
  }
}
```

此政策的邏輯可簡化為：

```
("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"))
```

對於非巢狀政策，組合總計的計算相同，除非我們需要考慮關鍵的巢狀等級。

組合總計的計算如下：

```
(2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) = 32
```

`metricName` 有兩個值、`namespace` 有兩個值、`scope` 具有兩個巢狀索引鍵與一個值，`source` 具有兩個巢狀索引鍵與一個值，並且 `type` 具有兩個巢狀索引鍵與一個值。

索引鍵比對

在篩選條件政策中使用 `exists` 運算子，根據特定屬性是否存在，來比對傳入的訊息。

- `exists` 僅適用於分葉節點（結構中的最終屬性）。
- 它不適用於巢狀 JSON 結構中的中繼節點。
- 使用 `"exists": true` 配對包含指定屬性的傳入訊息。該鍵必須具有非空值和非空值。

例如，下列政策屬性會使用 `exists` 運算子及為 `true` 的值：

```
"store": [{"exists": true}]
```

它符合具有 `store` 屬性索引鍵的任何訊息屬性，如下所示：

```
"store": {"Type": "String", "Value": "fans"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball\"]"}
```

它也符合以下任一訊息內文：

```
{  
  "store": "fans"  
  "customer_interests": ["baseball", "basketball"]  
}
```

不過，不符合不具有 store 屬性索引鍵的任何訊息屬性，如下所示：

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball\"]"}
```

它也不符合以下訊息內文：

```
{  
  "customer_interests": ["baseball", "basketball"]  
}
```

- 使用 "exists": false 配對不包含指定屬性的傳入訊息。

Note

"exists": false 只有在至少有一個屬性存在時才相符。一組空白的屬性會導致篩選條件不相符。

例如，下列政策屬性會使用 exists 運算子及為 false 的值：

```
"store": [{"exists": false}]
```

它不符合具有 store 屬性索引鍵的任何訊息，如下所示：

```
"store": {"Type": "String", "Value": "fans"}  
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball\"]"}
```

它也不符合以下訊息內文：

```
{
  "store": "fans"
  "customer_interests": ["baseball", "basketball"]
}
```

不過，它符合不具有 store 屬性索引鍵的任何訊息屬性，如下所示：

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball\\\"]"}
```

它也符合以下訊息內文：

```
{
  "customer_interests": ["baseball", "basketball"]
}
```

數值比對

透過將數值與訊息屬性值或訊息內文屬性值相符來篩選訊息。在 JSON 政策中，數值不是以雙引號括住。您可以使用以下數值操作來篩選。

Note

僅在字串比對時支援字首。

完全符合

當政策屬性值包含 numeric 關鍵字和 = 運算子時，它符合任何具有相同名稱和相等數值的訊息屬性或訊息內文屬性值。

舉例下列政策屬性：

```
"price_usd": [{"numeric": ["=", 301.5]}]
```

它符合以下任一訊息屬性：

```
"price_usd": {"Type": "Number", "Value": 301.5}
```

```
"price_usd": {"Type": "Number", "Value": 3.015e2}
```

它也符合以下任一訊息內文：

```
{  
  "price_usd": 301.5  
}
```

```
{  
  "price_usd": 3.015e2  
}
```

除外相符

當政策屬性值包含關鍵字 `anything-but`，就會比對不包含任何政策屬性值的任何訊息屬性或訊息內文屬性值。

舉例下列政策屬性：

```
"price": [{"anything-but": [100, 500]}]
```

它符合以下任一訊息屬性：

```
"price": {"Type": "Number", "Value": 101}
```

```
"price": {"Type": "Number", "Value": 100.1}
```

它也符合以下任一訊息內文：

```
{  
  "price": 101  
}
```

```
{  
  "price": 100.1  
}
```

除此之外，它符合以下訊息屬性 (因為它包含的值不是 100 或 500)：

```
"price": {"Type": "Number.Array", "Value": "[100, 50]"}
```

它也符合以下訊息內文 (因為它包含的值不是 100 或 500)：

```
{  
  "price": [100, 50]  
}
```

但不符合以下訊息屬性：

```
"price": {"Type": "Number", "Value": 100}
```

它也不符合以下訊息內文：

```
{  
  "price": 100  
}
```

值範圍相符

除了 = 運算子，數值政策屬性還可包括下列運算子：<、<=、> 和 >=。

舉例下列政策屬性：

```
"price_usd": [{"numeric": ["<", 0]}]
```

它符合具有負數值的任何訊息屬性或訊息內文屬性。

舉例另一個訊息屬性：

```
"price_usd": [{"numeric": [ ">", 0, "<=", 150 ]}]
```

它符合正數最大為 150 (含 150) 的任何訊息屬性或訊息內文屬性。

字串值比對

透過將字串值與訊息屬性值或訊息內文屬性值相符來篩選訊息。在 JSON 政策中，字串值是以雙引號括住。您可以使用下列字串操作來比對訊息屬性或訊息內文屬性：

完全符合

政策屬性值符合一或多個訊息屬性值時，即完全相符。對於String.Array類型屬性，陣列中的每個元素都會被視為個別字串，以用於比對目的。

舉例下列政策屬性：

```
"customer_interests": ["rugby", "tennis"]
```

它符合以下訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String", "Value": "tennis"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"tennis\"]"}
```

它也符合以下訊息內文：

```
{  
  "customer_interests": "rugby"  
}
```

```
{  
  "customer_interests": "tennis"  
}
```

不過，它不符合下列訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\"]"}
```

它也不符合以下訊息內文：

```
{  
  "customer_interests": "baseball"  
}
```

除外相符

當政策屬性值包含關鍵字 `anything-but`，就會比對不包含任何政策屬性值的任何訊息屬性或訊息內文值。`anything-but` 可以與 `"exists": false` 結合。對於 `String.Array` 類型屬性，如果政策屬性中未列出任何陣列元素，則會比對。

舉例下列政策屬性：

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

它符合下列任何訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

它也符合以下任一訊息內文：

```
{  
  "customer_interests": "baseball"  
}
```

```
{  
  "customer_interests": "football"  
}
```

除此之外，它符合以下訊息屬性 (因為它包含的值不是 `rugby` 或 `tennis`)：

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

它也符合以下訊息內文 (因為它包含的值不是 `rugby` 或 `tennis`)：

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

不過，它不符合下列訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\"]"}
```

它也不符合以下訊息內文：

```
{  
  "customer_interests": ["rugby"]  
}
```

使用具 **anything-but** 運算子的字首

針對字串比對，您也可以使用具 **anything-but** 運算子的字首。例如，下列政策屬性會拒絕 **order-**字首：

```
"event": [{"anything-but": {"prefix": "order-"}}]
```

它符合以下任一屬性：

```
"event": {"Type": "String", "Value": "data-entry"}
```

```
"event": {"Type": "String", "Value": "order_number"}
```

它也符合以下任一訊息內文：

```
{  
  "event": "data-entry"  
}
```

```
{  
  "event": "order_number"  
}
```

但不符合以下訊息屬性：

```
"event": {"Type": "String", "Value": "order-cancelled"}
```


它也不符合以下訊息內文：

```
{
  "event": "order-cancelled"
}
```

等於忽略大小寫比對

當政策屬性包含關鍵字 `equals-ignore-case` 時，將對任何訊息屬性或內文屬性值執行不分大小寫的比對。

舉例下列政策屬性：

```
"customer_interests": [{"equals-ignore-case": "tennis"}]
```

它符合以下任一訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "TENNIS"}
```

```
"customer_interests": {"Type": "String", "Value": "Tennis"}
```

它也符合以下任一訊息內文：

```
{
  "customer_interests": "TENNIS"
}
```

```
{
  "customer_interests": "teNnis"
}
```

IP 地址比對

您可以使用 `cidr` 運算子來檢查傳入訊息是否來自特定 IP 地址或子網路。

舉例下列政策屬性：

```
"source_ip": [{"cidr": "10.0.0.0/24"}]
```

它符合以下任一訊息屬性：

```
"source_ip": {"Type": "String", "Value": "10.0.0.0"}
```

```
"source_ip": {"Type": "String", "Value": "10.0.0.255"}
```

它也符合以下任一訊息內文：

```
{  
  "source_ip": "10.0.0.0"  
}
```

```
{  
  "source_ip": "10.0.0.255"  
}
```

但不符合以下訊息屬性：

```
"source_ip": {"Type": "String", "Value": "10.1.1.0"}
```

它也不符合以下訊息內文：

```
{  
  "source_ip": "10.1.1.0"  
}
```

前綴相符

當政策屬性包含 `prefix` 關鍵字時，它符合任何以特定字元為開頭的訊息屬性或內文屬性值。

舉例下列政策屬性：

```
"customer_interests": [{"prefix": "bas"}]
```

它符合以下任一訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

它也符合以下任一訊息內文：

```
{  
  "customer_interests": "baseball"  
}
```

```
{  
  "customer_interests": "basketball"  
}
```

但不符合以下訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

它也不符合以下訊息內文：

```
{  
  "customer_interests": "rugby"  
}
```

後綴相符

當政策屬性包含 `suffix` 關鍵字時，它符合任何以特定字元為結束的訊息屬性或內文屬性值。

舉例下列政策屬性：

```
"customer_interests": [{"suffix": "ball"}]
```

它符合以下任一訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

它也符合以下任一訊息內文：

```
{
  "customer_interests": "baseball"
}
```

```
{
  "customer_interests": "basketball"
}
```

但不符合以下訊息屬性：

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

它也不符合以下訊息內文：

```
{
  "customer_interests": "rugby"
}
```

在 Amazon SNS 中套用訂閱篩選條件政策

Amazon SNS 中的訊息篩選可讓您根據篩選政策，選擇性地將訊息傳遞給訂閱者。這些政策定義訊息必須符合的條件，才能交付至訂閱。雖然原始訊息傳遞是可能影響訊息處理的選項，但訂閱篩選條件不需要運作。

您可以使用 Amazon SNS 主控台將篩選政策套用到 Amazon SNS 訂閱。或者，若要以程式設計方式套用政策，您可以使用 Amazon SNS API、AWS Command Line Interface (AWS CLI) 或任何支援 Amazon SNS 的 AWS SDK。您也可以使用 AWS CloudFormation。

啟用原始訊息傳遞

原始訊息傳遞可確保訊息承載依原狀交付給訂閱者，而不需要任何額外的編碼或轉換。當訂閱者需要原始訊息格式進行處理時，這很有用。不過，原始訊息傳遞與訂閱篩選條件的功能不直接相關。

套用訂閱篩選條件

若要將訊息篩選條件套用到訂閱，您可以使用 JSON 語法定義篩選條件政策。此政策指定訊息必須符合的條件，才能交付至訂閱。篩選條件可以根據訊息屬性，例如訊息屬性、訊息結構，甚至訊息內容。

原始訊息交付與訂閱篩選條件之間的關係

雖然啟用原始訊息傳遞可能會影響訂閱者交付和處理訊息的方式，但這不是使用訂閱篩選條件的先決條件。不過，在訂閱者需要原始訊息格式而不進行任何修改的情況下，啟用原始訊息傳遞可能比訂閱篩選條件更有益。

有效篩選的考量事項

實作訊息篩選時，請考慮應用程式和訂閱者的特定需求。定義準確符合訊息傳遞條件的篩選政策，以確保有效率且目標性的訊息分發。

Important

AWS IAM 和 Amazon SNS 等服務會使用稱為最終一致性的分散式運算模型。對訂閱篩選原則進行新增或變更，最多需要 15 分鐘才能完全生效。

AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Subscriptions (訂閱)。
3. 選取訂閱，然後選擇 Edit (編輯)。
4. 在 Edit (編輯) 頁面上，展開 Subscription filter policy (訂閱篩選政策) 區段。
5. 選擇以屬性為基礎的篩選或以承載為基礎的篩選。
6. 在 JSON editor (JSON 編輯器) 欄位中，提供您篩選政策的 JSON 內文。
7. 選擇 Save changes (儲存變更)。

Amazon SNS 套用您的篩選政策到訂閱。

AWS CLI

若要使用 AWS Command Line Interface (AWS CLI) 套用篩選條件政策，請使用 [set-subscription-attributes](#) 命令，如下列範例所示。對於 --attribute-name 選項，指定 FilterPolicy。針對 --attribute-value，請指定 JSON 政策。

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

若要為您的政策提供有效的 JSON，請使用雙引號括住屬性名稱和值。您也必須用引號括住整個政策引數。若要避免逸出引號，您可以如上方範例所示，使用單引號括住政策，並用雙引號括住 JSON 名稱和值。

若您想要從以屬性為基礎 (預設) 切換到以承載為基礎的訊息篩選，也可以使用 [set-subscription-attributes](#) 命令。對於 `--attribute-name` 選項，指定 `FilterPolicyScope`。對於 `--attribute-value`，請指定 `MessageBody`。

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicyScope --attribute-value MessageBody
```

如果要確認是否已套用篩選政策，請使用 `get-subscription-attributes` 命令。如以下範例所示，內部輸出中的屬性應顯示您的 `FilterPolicy` 索引鍵的篩選政策：

```
$ 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

下列程式碼範例示範如何使用 `SetSubscriptionAttributes`。

Important

如果您使用適用於 Java 2.x 的 SDK 範例，則類別 `SNSMessageFilterPolicy` 非可立即運作。如需如何安裝此類別的指示，請參閱 GitHub 網站的 [範例](#)。

CLI

AWS CLI

設定訂閱屬性

下列 `set-subscription-attributes` 範例會將 `RawMessageDelivery` 屬性設定為 SQS 訂閱。

```
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
```

此命令不會產生輸出。

下列 `set-subscription-attributes` 範例會將 `FilterPolicy` 屬性設定為 SQS 訂閱。

```
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\"] }"
```

此命令不會產生輸出。

下列 `set-subscription-attributes` 範例會從 SQS 訂閱移除 `FilterPolicy` 屬性。

```
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 "{}"
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetSubscriptionAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [SetSubscriptionAttributes](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [SetSubscriptionAttributes](#)。

Amazon SNS API

若要使用 Amazon SNS API 套用篩選政策，請提出 [SetSubscriptionAttributes](#) 動作的請求。將 `AttributeName` 參數設定為 `FilterPolicy`，然後將 `AttributeValue` 參數設定為您的篩選政策 JSON。

若您想要從以屬性為基礎 (預設) 切換到以承載為基礎的訊息篩選，也可以使用 [SetSubscriptionAttributes](#) 動作。將 `AttributeName` 參數設定為 `FilterPolicyScope`，並將 `AttributeValue` 參數設定為 `MessageBody`。

AWS CloudFormation

若要使用 套用篩選政策 AWS CloudFormation，請使用 JSON 或 YAML 範本來建立 AWS CloudFormation 堆疊。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 `AWS::SNS::Subscription` 資源的 [FilterPolicy 屬性](#)和[範例 AWS CloudFormation 範本](#)。

1. 登入 [AWS CloudFormation 主控台](#)。
2. 選擇 Create Stack (建立堆疊)。
3. 在 Select Template (選擇範本) 頁面中，選擇 Upload a template to Amazon S3 (上傳範本到 Amazon S3)、選擇檔案，並選擇 Next (下一步)。
4. 在 Specify Details (指定詳細資訊) 頁面上，執行下列作業：
 - a. 在 Stack Name (堆疊名稱) 中輸入 `MyFilterPolicyStack`。
 - b. 對於 `myHttpEndpoint`，輸入要訂閱到主題的 HTTP 端點。

 Tip

若您沒有 HTTP 端點，請建立一個。

5. 在 Options (選項) 頁面上，選擇 Next (下一步)。
6. 在 Review (檢閱) 頁面上，選擇 Create (建立)。

在 Amazon SNS 中移除訂閱篩選條件政策

若要停止篩選傳送到訂閱的訊息，請以空的 JSON 內文覆寫訂閱的篩選政策來移除政策。移除政策後，訂閱會接受發佈給它的每個訊息。

使用 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Subscriptions (訂閱)。
3. 選取訂閱，然後選擇 Edit (編輯)。
4. 在 Edit **EXAMPLE1-23bc-4567-d890-ef12g3hij456** (EXAMPLE1-23bc-4567-d890-ef12g3hij456) 頁面上，展開 Subscription filter policy (訂閱篩選政策) 區段。
5. 在 JSON editor (JSON 編輯器) 欄位中，提供您篩選政策的空白 JSON 內文：{ }。
6. 選擇 Save changes (儲存變更)。

Amazon SNS 套用您的篩選政策到訂閱。

使用 AWS CLI

若要使用 移除篩選條件政策 AWS CLI，請使用 [set-subscription-attributes](#) 命令，並提供引 --attribute-value 數的空白 JSON 內文：

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

使用 Amazon SNS API

若要使用 Amazon SNS API 移除篩選政策，請提出 [SetSubscriptionAttributes](#) 動作的請求。將 `AttributeName` 參數設定為 `FilterPolicy`，並為 `AttributeValue` 參數提供空的 JSON 內文。

Amazon SNS 中的訊息資料保護

什麼是訊息資料保護？

訊息資料保護會使用[資料保護政策](#)來稽核、遮罩、修訂或封鎖在應用程式 AWS 或服務之間移動的敏感資訊，以保護發佈至 Amazon SNS 主題的資料。

訊息資料保護會使用資料識別符來掃描動態資料中是否有個人身分識別資訊 (PII) 和受保護醫療資訊 (PHI)。您可以選擇使用[預先定義](#) (或 Amazon SNS 受管) 的資料識別符 (例如姓名、地址、信用卡號碼和處方藥代碼)，也可以建立[自訂](#)資料識別符 (您的業務使用案例專屬)。使用掃描的資訊，訊息資料保護可提供詳細的稽核日誌，並可讓您採取行動來保護該資料。

訊息資料保護支援下列動作，以協助保護敏感的客戶資訊：

- [稽核](#) – 可稽核高達 99% 發佈到 Amazon SNS 主題的資料。然後，您可以選擇將調查結果傳送至 [Amazon CloudWatch](#)、[Amazon S3](#) 或 [Amazon Data Firehose](#)。
- [去識別化](#) – 遮罩或修改敏感資料，而不中斷訊息發布或提交。
- [拒絕](#) – 如果承載中存在敏感資料，則封鎖應用程式 AWS 和資源之間的資料傳輸。

Note

Amazon SNS 僅支援 Amazon SNS 標準主題的訊息資料保護。

為什麼要使用訊息資料保護？

透過將訊息資料保護引入您的控管、風險管理和法規遵循計畫中，您可以實作資料保護政策，協助您識別並防止資料外洩。這為您的團隊提供了一些工具，可以透過遵守 HIPAA，GDPR，PCI 和 FedRAMP 等隱私權法規來幫助降低財務、法律和法規風險。它還可讓您的開發人員免受與建置和管理自己工具以保護敏感資料相關的營運開銷。

例如，您可以使用訊息資料保護來建立稽核政策，用以判斷任何系統是否意外傳送或接收敏感資料。如果稽核結果顯示系統正在傳送信用卡資訊至不需要此資訊的系統，您可以使用封鎖政策來防止資料的傳遞。

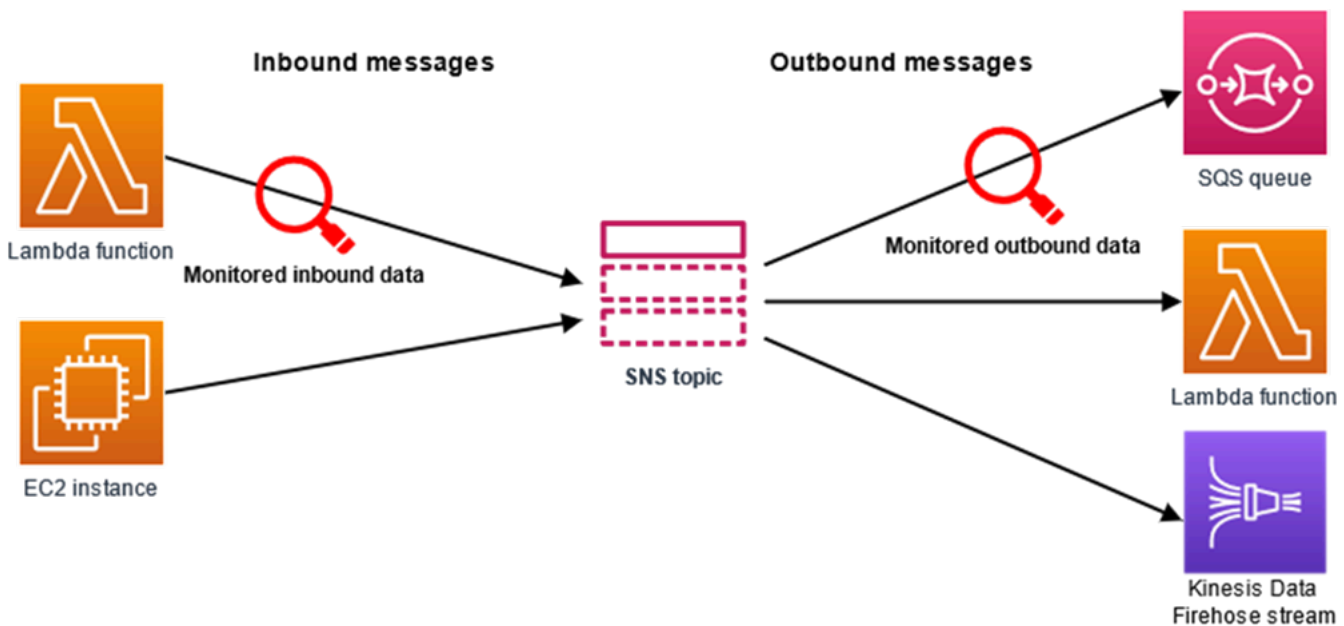
Note

Amazon SNS 僅支援 Amazon SNS 標準主題的訊息資料保護。

了解 Amazon SNS 資料保護政策

什麼是資料保護政策？

Amazon SNS 使用資料保護政策來選取您要掃描的敏感資料，以及您想要採取的動作以防止 Amazon SNS 主題交換該資料。若要選擇感興趣的敏感資料，請使用[資料識別符](#)。然後，Amazon SNS 訊息資料保護會使用機器學習和模式比對來偵測敏感資料。若要根據找到的資料識別符採取行動，您可以定義稽核、去識別化或拒絕操作。這些操作可讓您記錄找到 (或找不到) 的敏感資料、遮罩或修訂敏感資料，或拒絕訊息傳遞。

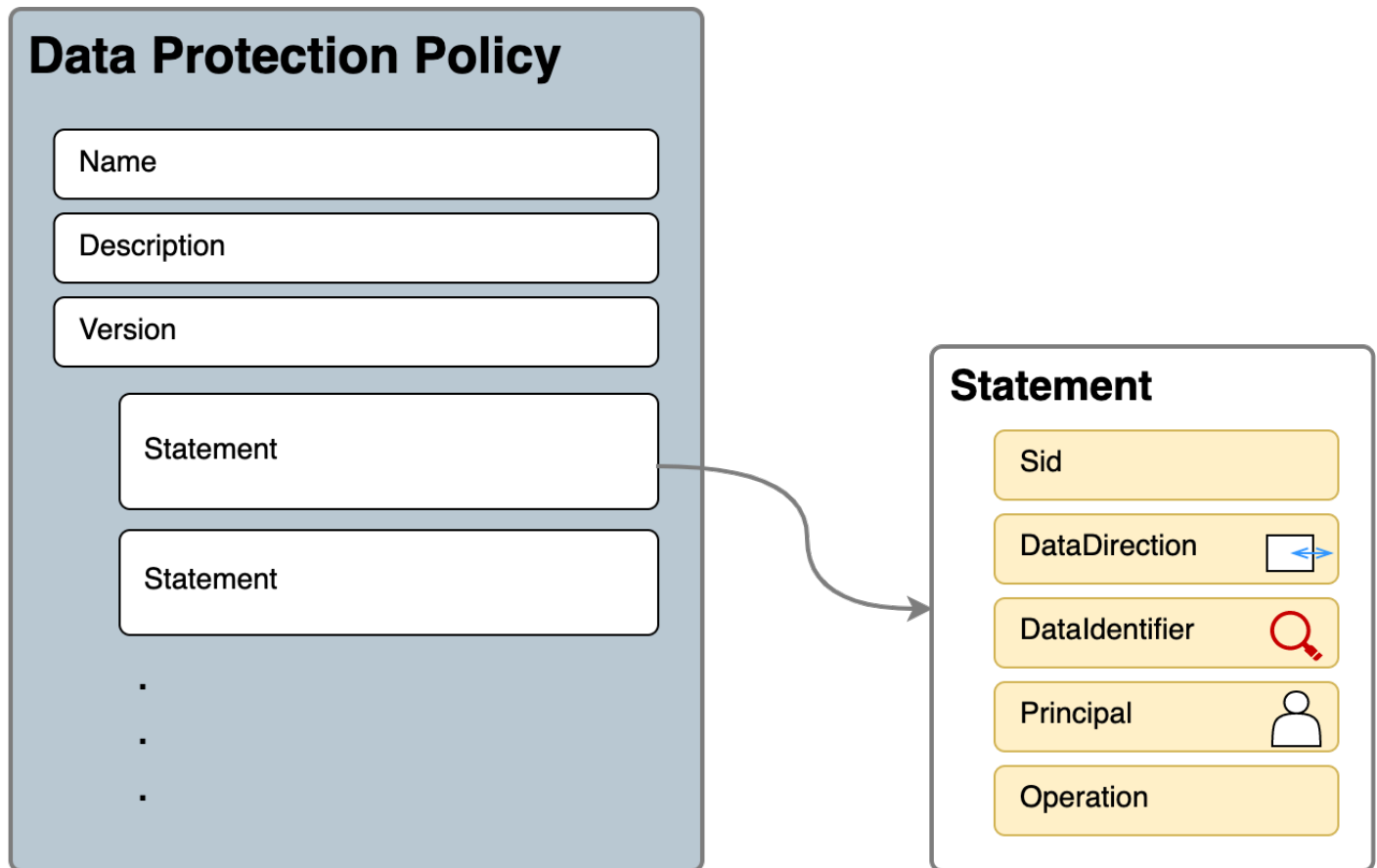


資料保護政策的結構如何？

如下圖所示，資料保護政策文件包含以下元素：

- 在文件最上方選用的整體政策資訊
- 一或多個個別的陳述式

每個陳述式包含關於單一許可的資訊。



每個 Amazon SNS 主題只能定義一個資料保護政策。資料保護政策可以有一或多個拒絕或去識別化陳述式，但只能有一個稽核陳述式。

資料保護政策的 JSON 屬性

資料保護政策需要下列基本政策資訊才能識別：

- Name - 政策名稱。
- Description (選用) - 政策描述。
- Version - 政策語言版本。目前版本是 2021-06-01。
- Statement - 指定資料保護政策動作的陳述式清單。

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
    ...
  ]
}
```



```
]
}
```

政策陳述式的 JSON 屬性

政策陳述式會設定資料保護操作的偵測內容。

- Sid (選用) - 陳述式識別符。
- DataDirection - 與 Amazon SNS 主題相關的傳入 (針對發佈 API 請求) 或傳出 (用於通知交付)。
- DataIdentifier - Amazon SNS 主題應掃描的敏感資料。例如，姓名、地址或電話號碼。
- Principal - 發布至主題的 IAM 主體，或訂閱主題的 IAM 主體。
- 操作 - Amazon SNS 主題找到敏感資料後會執行的後續動作：Audit (稽核)、De-identify (去識別化) (遮罩或修改) 或 Deny (拒絕) (封鎖)。

```
{
  "Sid": "basicPII-inbound-protection",
  "DataDirection": "Inbound",
  "Principal": ["*"],
  "DataIdentifier": [
    "arn:aws:dataprotection::aws:data-identifier/Name",
    "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
  ],
  "Operation": {
    ...
  }
}
```

政策陳述式操作的 JSON 屬性

政策陳述式會設定下列其中一項資料保護操作。

- [Audit](#) - 發出指標和發現結果日誌，但不中斷訊息發佈或傳遞。
- [去識別化](#) - 遮罩或修改敏感資料，而不中斷訊息發佈。
- [Deny](#) - 封鎖 Amazon SNS 發佈請求或讓訊息傳遞失敗。

如何判斷資料保護政策的 IAM 主體？

訊息資料保護使用兩個與 Amazon SNS 互動的 IAM 主體。

1. 發佈 API 主體 (傳入) - 已驗證的 IAM 主體，呼叫 Amazon SNS Publish API。
2. 訂閱主體 (傳出) - 已驗證的 IAM 主體，在建立訂閱期間呼叫 Subscribe API。

SubscriptionPrincipal 是公開可用的 Amazon SNS 訂閱屬性，可從 GetSubscriptionAttributes API 擷取。

```
{
  "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"
  }
}
```

Amazon SNS 中的資料保護政策操作

以下是您可以用來稽核和拒絕敏感資料的資料保護政策的範例。如需包含範例應用程式的完整教學課程，請參閱 [Introducing message data protection for Amazon SNS](#) (Amazon SNS 的訊息資料保護簡介) 部落格文章。

Audit 操作

稽核操作會範例傳入訊息主題，並記錄 AWS 目的地中的敏感資料問題清單。取樣率可以是介於 0-99 之間的整數。此操作需要下列其中一種記錄目的地類型：

1. FindingsDestination - Amazon SNS 主題在承載中找到敏感資料時的記錄目的地。
2. NoFindingsDestination - Amazon SNS 主題在承載中找不到敏感資料時的記錄目的地。

您可以在下列每種日誌目的地類型 AWS 服務 中使用下列項目：

- Amazon CloudWatch Logs (選用) - LogGroup 必須在主題區域中，且名稱開頭必須是 /aws/vendedlogs/。

- Amazon Data Firehose (選用) – DeliveryStream 必須位於主題區域，並以 Direct PUT 做為交付串流的來源。如需其他詳細資訊，請參閱《Amazon Data Firehose 開發人員指南》中的[來源、目的地和名稱](#)。
- Amazon S3 (選用) - Amazon S3 儲存貯體名稱。[在啟用 SSE-KMS 加密的情況下使用 Amazon S3 儲存貯體，需要採取額外動作](#)。

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

指定日誌目的地時所需的許可

當您在資料保護政策中指定記錄目的地時，必須使用 `--data-protection-policy` 參數，將下列許可新增至呼叫 Amazon SNS PutDataProtectionPolicy API 或 CreateTopic API 之 IAM 主體的 IAM 身分政策。

稽核目的地	IAM 許可
預設	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs>DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy 在啟用 SSE-KMS 加密的情況下使用 Amazon S3 儲存貯體，需要採取額外動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs>DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ]
    }
  ],
}
```

```

    "Resource": [
      "*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "logs:PutResourcePolicy",
      "logs:DescribeResourcePolicies",
      "logs:DescribeLogGroups"
    ],
    "Resource": [
      "arn:aws:logs:region:account-id:SampleLogGroupName:*:*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "iam:CreateServiceLinkedRole",
      "firehose:TagDeliveryStream"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutBucketPolicy",
      "s3:GetBucketPolicy"
    ],
    "Resource": [
      "arn:aws:s3:::bucket-name"
    ]
  }
]
}

```

搭配 SSE-KMS 使用的必要金鑰政策

如果您使用 Amazon S3 儲存貯體做為日誌目的地，您可以透過啟用伺服器端加密搭配 Amazon S3-Managed 管金鑰 (SSE-S3) 或伺服器端加密搭配 AWS KMS keys (SSE-KMS) 來保護儲存貯體中的資料。如需詳細資訊，請參閱《Amazon S3 使用者指南》中的[使用伺服器端加密保護資料](#)。

如果您選擇 SSE-S3，則不需要其他組態。Amazon S3 會處理加密金鑰。

若您選擇 SSE-KMS，則必須使用客戶受管金鑰。您必須更新客戶受管金鑰的金鑰政策，以讓日誌傳遞帳戶能夠寫入您的 S3 儲存貯體。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的 [Amazon S3 儲存貯體伺服器端加密](#)。

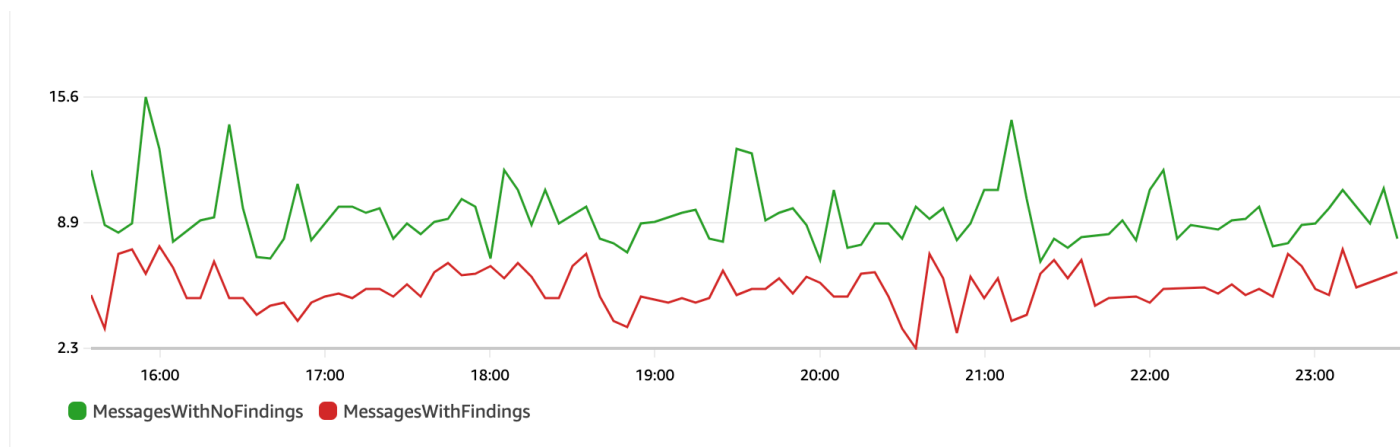
稽核目的地日誌範例

在以下範例中，`callerPrincipal` 用於識別敏感內容來源，`messageID` 用作檢查 Publish API 回應的參考。

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

Audit 操作指標

當 audit 操作已指定 FindingsDestination 或 NoFindingsDestination 屬性時，主題擁有者也會收到 CloudWatch MessagesWithFindings 和 MessagesWithNoFindings 指標。



去識別化操作

去識別化操作會遮罩或修訂已發布或交付訊息中的敏感資料。此操作適用於傳入與傳出訊息，且需要下列其中一種設定類型：

- MaskConfig - 使用下表中支援的字元進行遮罩。例如，ssn:123-45-6789 變成 ssn:#####。

```
{
  "Operation": {
    "Deidentify": {
      "MaskConfig": {
        "MaskWithCharacter": "#"
      }
    }
  }
}
```

受支援的遮罩字元	名稱
*	星號
A-Z、a-z 和 0-9	英數字元
	空格

受支援的遮罩字元	名稱
!	驚嘆號
\$	貨幣符號
%	百分比符號
&	& 符號
()	括號
+	加號
,	逗號
-	連字號
.	期間
/	斜線、反斜線
#	井字號
:	冒號
;	分號
=, <>	等於、小於或大於
@	@ 符號
[]	括弧
^	插入符號
_	底線
`	反引號
	垂直分隔號

受支援的遮罩字元	名稱
~	波浪符號

- RedactConfig - 完全刪除資料以進行修訂。例如，ssn:123-45-6789 變成 ssn: 。

```
{
  "Operation": {
    "Deidentify": {
      "RedactConfig": {}
    }
  }
}
```

在傳入訊息上，會在稽核操作後去識別化敏感資料，且當整則訊息皆敏感時，SNS:Publish API 呼叫者會收到下列無效參數錯誤。

Error code: AuthorizationError ...

Deny 操作

如果訊息包含敏感資料，Deny 操作會中斷 Publish API 請求或訊息傳遞。Deny 操作物件是空的，因為它不需要額外的組態。

```
"Operation": {
  "Deny": {}
}
```

在傳入訊息上，SNS:Publish API 呼叫者會收到授權錯誤。

Error code: AuthorizationError ...

在傳出訊息上，Amazon SNS 主題不會將訊息傳遞至訂閱。若要追蹤未經授權的傳遞，請啟用主題的[傳遞狀態記錄](#)。下列為傳遞狀態日誌的範例：

```
{
  "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"
}
```

Amazon SNS 資料保護政策範例

以下範例是您可以用來稽核和拒絕敏感資料的資料保護政策。如需包含範例應用程式的完整教學課程，請參閱 [Introducing message data protection for Amazon SNS](#) (Amazon SNS 的訊息資料保護簡介) 部落格文章。

稽核的範例政策

稽核政策可讓您稽核高達 99% 的傳入訊息，並將調查結果傳送至 [Amazon CloudWatch](#)、[Amazon Data Firehose](#) 和 [Amazon S3](#)。

例如，您可以建立稽核政策來評估是否有任何系統意外傳送或接收敏感資料。如果稽核結果顯示系統正在傳送信用卡資訊至不需要此資訊的系統，您可以使用封鎖政策來防止資料的傳遞。

下列範例會尋找信用卡號碼並將問題清單傳送至 CloudWatch Logs、Firehose 和 Amazon S3，藉此稽核 99% 通過主題的訊息。

資料保護政策：

```
{
  "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>"
          }
        }
      }
    }
  }
}

```

稽核結果格式範例：

```

{
  "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"
}

```

政策傳入去識別化遮罩陳述式範例

下列範例會遮罩訊息內容中的敏感資料，防止使用者將訊息發佈至含有 CreditCardNumber 的主題。

```
{
  "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": "#"
          }
        }
      }
    }
  ]
}
```

傳入去識別化修改結果範例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is #####
```

政策傳入去識別化修改陳述式範例

下列範例會修訂訊息內容中的敏感資料，防止使用者將訊息發佈至含有 CreditCardNumber 的主題。

```
{
  "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": {}
    }
  }
}
```

傳入去識別化修訂結果範例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

政策傳出去識別化遮罩陳述式範例

下列範例會遮罩訊息內容中的敏感資料，防止使用者接收含有 CreditCardNumber 的訊息。

```
{
  "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": "-"
        }
      }
    }
  }
}
]
}

```

傳出去識別化遮罩結果範例：

```

// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is -----

```

政策傳出去識別化修改陳述式範例

下列範例會修改訊息內容中的敏感資料，防止使用者接收含有 CreditCardNumber 的訊息。

```

{
  "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": {}
        }
      }
    }
  ]
}

```

```
}
```

傳出去識別化修訂結果範例：

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

政策傳入拒絕陳述式範例

下列範例會防止使用者將訊息內容中有 CreditCardNumber 的訊息發佈至主題。API 回應中遭拒的承載狀態碼為「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": {}
      }
    }
  ]
}
```

傳出拒絕陳述式的政策範例

下列範例會封鎖 AWS 帳戶接收包含 的訊息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": {}
    }
  }
]
}

```

Amazon CloudWatch 中記錄的傳出拒絕結果範例：

```

{
  "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"
}

```

在 Amazon SNS 中建立資料保護政策

[資料保護政策](#)透過稽核、去識別化 (遮罩或修訂) 和拒絕 (封鎖) 在應用程式或 AWS 服務服務之間移動的敏感資料，來保護發佈到 Amazon SNS 主題的資料。您可以使用 AWS API、AWS CLI、AWS

CloudFormation 或 AWS Management Console 在 Amazon SNS 中建立資料保護政策。每個 Amazon SNS 主題只能定義一個政策。每個資料保護政策可以有一或多個去識別化和拒絕陳述式，但只能有一個稽核陳述式。

主題

- [使用 API 在 Amazon SNS 中建立資料保護政策](#)
- [使用 CLI 在 Amazon SNS 中建立資料保護政策](#)
- [使用 CloudFormation 在 Amazon SNS 中建立資料保護政策](#)
- [使用主控台在 Amazon SNS 中建立資料保護政策](#)
- [使用 SDK 建立 Amazon SNS 資料保護政策以保護訊息資料](#)

使用 API 在 Amazon SNS 中建立資料保護政策

AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱 [Amazon Simple Notification Service 端點和配額](#)。

使用 API 建立資料保護政策

使用 AWS API 建立 Amazon SNS 資料保護政策。

若要與 Amazon SNS 主題一起建立資料保護政策 (AWS API)

使用標準 Amazon SNS 主題的 DataProtectionPolicy 屬性：

- [CreateTopic](#)

若要擷取或建立現有 Amazon SNS 主題的資料保護政策 (AWS API)

呼叫以下其中一項操作：

- [GetDataProtectionPolicy](#)
- [PutDataProtectionPolicy](#)

使用 CLI 在 Amazon SNS 中建立資料保護政策

AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱 [Amazon Simple Notification Service 端點和配額](#)。

使用 建立資料保護政策 AWS CLI

使用 建立 Amazon SNS 資料保護政策 AWS Command Line Interface。

若要與 Amazon SNS 主題一起建立資料保護政策 (AWS CLI)

使用此選項，與標準 Amazon SNS 主題一起建立新的資料保護政策：

- [create-topic](#)

若要建立或擷取現有 Amazon SNS 主題的資料保護政策 (AWS CLI)

呼叫以下其中一項操作：

- [get-data-protection-policy](#)
- [put-data-protection-policy](#)

使用 CloudFormation 在 Amazon SNS 中建立資料保護政策

AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱 [Amazon Simple Notification Service 端點和配額](#)。

建立資料保護政策 (CloudFormation)

使用 建立 Amazon SNS 資料保護政策 AWS CloudFormation。

若要與 Amazon SNS 主題一起建立資料保護政策 (CloudFormation)

使用此選項，與標準 Amazon SNS 主題一起建立新的資料保護政策：

- [AWS::SNS::Topic](#)

使用主控台在 Amazon SNS 中建立資料保護政策

AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱 [Amazon Simple Notification Service 端點和配額](#)。

若要與 Amazon SNS 主題一起建立資料保護政策 (主控台)

使用此選項可同時建立新的資料保護政策與標準 Amazon SNS 主題。

1. 登入 [Amazon SNS 主控台](#)。
2. 選擇主題或建立新主題。如需建立主題的詳細資訊，請參閱[建立 Amazon SNS 主題](#)。
3. 在 Create topic (建立主題) 頁面的 Details (詳細資訊) 區段中，選擇 Standard (標準)。
 - a. 輸入新主題的 名稱 (Name)。
 - b. (選用) 為主題輸入 Display name (顯示名稱)。
4. 展開 Data protection policy (資料保護政策)。
5. 選擇 Configuration mode (組態模式) :
 - Basic (基本) - 使用簡單的功能表定義資料保護政策。
 - Advanced (進階) - 使用 JSON 定義自訂資料保護政策。
6. (選用) 若要建立自己的自訂資料識別符，請展開自訂資料識別符組態區段，並執行下列動作：
 - a. 輸入自訂資料識別符的唯一名稱。自訂資料識別符的名稱可支援英數字、底線 (_) 和連字號 (-) 字元。最多可支援 128 個字元。此名稱不可與[受管資料識別符](#)的名稱相同。如需自訂資料識別符限制的完整清單，請參閱 [自訂資料識別符的限制](#)。
 - b. 輸入自訂資料識別符的規則運算式 (RegEx)。RegEx 可支援英數字元、RegEx 保留字元和符號。RegEx 的長度上限為 200 個字元。如果 RegEx 太過複雜，Amazon SNS 會讓 API 呼叫失敗。如需 RegEx 限制的完整清單，請參閱 [自訂資料識別符的限制](#)。
 - c. (選用) 選擇新增自訂資料識別符，視需要新增其他資料識別符。每個資料保護政策最多可支援 10 個自訂資料識別符。
7. 選擇您要新增至資料保護政策的陳述式。您可以將audit (稽核)、de-identify (去識別化) (遮罩或修訂) 和deny (拒絕) (封鎖) 陳述式類型新增至相同的資料保護政策。
 - a. Add audit statement (新增稽核陳述式) - 設定要稽核的敏感資料、要稽核該資料的訊息百分比，以及將稽核日誌傳送到何處。

 Note

每個資料保護政策或主題僅允許一個稽核陳述式。

- i. 選取 data identifiers (資料識別符) 定義您要稽核的敏感資料。
- ii. 對於 Audit sample rate (稽核採樣率)，輸入要稽核機密資訊的訊息百分比，上限為 99%。

- iii. 針對稽核目的地，選取要 AWS 服務 傳送稽核調查結果的結果，然後針對 AWS 服務 您使用的每個項目輸入目的地名稱。您可以從以下 Amazon Web Services 中進行選擇：
 - Amazon CloudWatch - CloudWatch Logs 是 AWS 標準記錄解決方案。使用 CloudWatch Logs，您可以使用 Logs Insights 執行日誌分析 ([在這裡查看範例](#)) 並建立指標和警示。CloudWatch Logs 是許多服務發佈日誌的地方，這可以簡化使用單一解決方案彙總所有日誌。如需 Amazon CloudWatch 的詳細資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。
 - Amazon Data Firehose – Firehose 滿足即時串流至 Splunk、OpenSearch 和 Amazon Redshift 的需求，以進行進一步的日誌分析。如需 Amazon Data Firehose 的相關資訊，請參閱《[Amazon Data Firehose 使用者指南](#)》。
 - Amazon Simple Storage Service - Amazon S3 是用於存檔的經濟型日誌目的地。您可能需要保留日誌一段時間。在此情況下，您可以將日誌放入 Amazon S3 中以節省成本。如需 Amazon Simple Storage Service 的相關資訊，請參閱 [Amazon Simple Storage Service 使用者指南](#)。
- b. Add a de-identify statement (新增去識別化陳述式) - 設定您要在訊息中去識別化 (無論遮罩或修訂) 的敏感資料，以及停止傳遞該資料的帳戶。
 - i. 針對 Data identifiers (資料識別符)，請選取您要去識別化的敏感資料。
 - ii. 針對定義此去識別化陳述式，選取套用此去識別化陳述式 AWS 的帳戶或 IAM 主體。您可以將其套用至所有 AWS 帳戶，或使用帳戶 ID 或 IAM 實體 ARN 的特定 AWS 帳戶或 IAM 實體 (帳戶根目錄、角色或使用者)。使用逗號 (,) 分隔多個 ID 或 ARN。

以下是支援的 [IAM](#) 主體：

- IAM account principals (IAM 帳戶主體) - 例如 `arn:aws:iam::AWS-account-ID:root`。
 - IAM role principals (IAM 角色主體) - 例如 `arn:aws:iam::AWS-account-ID:role/role-name`。
 - IAM user principals (IAM 使用者主體) - 例如 `arn:aws:iam::AWS-account-ID:user/user-name`。
- iii. 針對 De-identify Option (去識別化選項)，請選取您要如何去識別化敏感資料。支援下列選項：
 - Redact (修訂) - 完全移除資料。例如，電子郵件：`classified@amazon.com` 變成電子郵件：。

- Mask (遮罩) - 以單一字元取代資料。例如，電子郵件：classified@amazon.com 變成電子郵件：*****。
- iv. (選用) 視需要繼續新增去識別化陳述式。
- c. Add deny statement (新增拒絕陳述式) - 設定要防止在您的主題中移動的敏感資料，以及要防止哪些主參與者傳送該資料。
 - i. 針對 data direction (資料方向)，請選擇拒絕陳述式的訊息方向：
 - Inbound messages (傳入訊息) - 將此拒絕陳述式套用至傳送至主題的訊息。
 - Outbound messages (傳出訊息) - 將此拒絕陳述式套用至主題傳遞給訂閱端點的訊息。
 - ii. 選擇 data identifiers (資料識別符) 定義您要拒絕的敏感資料。
 - iii. 選擇此拒絕陳述式套用的 IAM principals (IAM 主體)。您可以將其套用至使用 AWS 帳戶 ID 或 IAM 實體 ARN 的所有帳戶、特定 AWS 帳戶或 IAM 實體 (例如帳戶根目錄、角色或使用者)。IDs ARNs 使用逗號 (,) 分隔多個 ID 或 ARN。以下是支援的 [IAM](#) 主體：
 - IAM account principals (IAM 帳戶主體) - 例如 arn:aws:iam::AWS-account-ID:root。
 - IAM role principals (IAM 角色主體) - 例如 arn:aws:iam::AWS-account-ID:role/role-name。
 - IAM user principals (IAM 使用者主體) - 例如 arn:aws:iam::AWS-account-ID:user/user-name。
 - iv. (選用) 視需要繼續新增拒絕陳述式。

使用 SDK 建立 Amazon SNS 資料保護政策以保護訊息資料

AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱 [Amazon Simple Notification Service 端點和配額](#)。

使用 SDK AWS 建立資料保護政策

使用 SDK 建立 Amazon SNS 資料保護政策 AWS。

若要與 Amazon SNS 主題一起建立資料保護政策 (AWS SDK)

使用下列選項，與標準 Amazon SNS 主題一起建立新的資料保護政策：

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();
```

若要建立或擷取現有 Amazon SNS 主題的資料保護政策 (AWS SDK)

使用下列選項，與標準 Amazon SNS 主題一起建立或擷取新的資料保護政策：

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();
```

刪除 Amazon SNS 中的資料保護政策

您可以使用 AWS API AWS CLI AWS CloudFormation、或刪除 Amazon SNS 資料保護政策 AWS Management Console。

如需 Amazon SNS 資料保護政策的一般資訊，請參閱 [了解 Amazon SNS 資料保護政策](#)。

AWS 帳戶中的 Amazon SNS 資料保護政策資源之數量和大小均有所限制。如需詳細資訊，請參閱 AWS 一般參考中的 [Amazon SNS API 限流](#)。

使用主控台刪除資料保護政策

使用主控台刪除受管資料保護政策

1. 登入 [Amazon SNS 主控台](#)。
2. 選擇包含您要刪除之資料保護政策的主題。
3. 選擇編輯。
4. 展開 Data protection policy (資料保護政策) 區段。
5. 選擇您要移除之資料保護政策陳述式旁的 Remove (移除)。
6. 選擇 Save changes (儲存變更)。

使用空的 JSON 字串刪除資料保護政策

您可以將資料保護政策更新為空的 JSON 字串來刪除資料保護政策。

使用 刪除資料保護政策 AWS CLI

您可使用 AWS CLI刪除資料保護政策。

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-  
protection-policy ""
```

Amazon SNS 資料識別符

Amazon SNS 使用標準和技術 (包括機器學習和模式比對) 的組合來偵測敏感資料。這些準則和技術統稱為資料識別符，可偵測許多國家和地區大量且不斷增加的敏感資料類型。Amazon SNS 受管資料識別符提供預先設定的資料類型，可用於保護財務資料、個人醫療資訊 (PHI) 和個人身分識別資訊 (PII)。您還可以使用自訂資料識別符，為您的特定使用案例建立獨有的資料識別符。

在 Amazon SNS 中使用受管資料識別碼

受管資料識別符是什麼？

Amazon SNS 受管資料識別符旨在偵測特定類型的敏感資料，例如信用卡號碼、AWS 私密存取金鑰或特定國家或地區的護照號碼。建立資料保護政策時，您可以設定 Amazon SNS 使用這些識別符來分析經歷該主題的訊息，並在偵測到訊息時採取動作。

Amazon SNS 可以使用受管資料識別符偵測下列類別的敏感資料：

- 登入資料，例如私有金鑰或 AWS 私密存取金鑰
- 裝置識別符，例如 IP 地址或 MAC 地址
- 財務資訊，例如信用卡號碼。
- 醫療資訊，像是健康保險或醫療識別號碼等 PHI。
- 個人資訊，像是駕照或社會安全號碼等 PII。

在每個類別中，Amazon SNS 都可以偵測多種類型的敏感資料。本節中的主題列出並說明每種類型以及偵測它的任何相關需求。對於每種類型，它們也會指出專為偵測資料而設計之受管理資料識別符的唯一識別符 (ID)。建立資料保護原則時，您可以使用此 ID 包含受管理的資料識別符，以便偵測到郵件資料保護。

關鍵字要求

為了偵測特定類型的敏感資料，Amazon SNS 會掃描資料附近的關鍵字。如果特定類型的資料是這種情況，本節的後續主題會指出該資料的特定關鍵字需求。

關鍵字不區分大小寫。此外，如果關鍵字包含空格，Amazon SNS 會自動比對不包含空格，或包含底線 (_) 或連字號 (-) 而非空格的關鍵字變體。在某些情況下，Amazon SNS 還會擴展或縮寫關鍵字，以解決關鍵字的常見變化。

適用於敏感資料類型的 Amazon SNS 受管資料識別符

下表列出並說明 Amazon SNS 可以使用受管資料識別符偵測的憑證、裝置、財務、醫療和個人健康資訊 (PHI) 類型。這些是某些資料類型的個人身分識別資訊 (PII) 等資料。

與區域相關的資料識別符需要帶有破折號的識別符名稱，以及兩個字母 (ISO 3166-1 alpha-2) 代碼。例如，DriversLicense-US。

識別符	類別	國家/地區/語言
BankAccountNumber	金融	DE、ES、FR、GB、IT
CepCode	個人	BR
Cnpj	個人	BR
CpfCode	個人	BR
DriversLicense	個人	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	醫療保健	US
ElectoralRollNumber	個人	GB
HealthInsuranceCardNumber	醫療保健	歐盟
HealthInsuranceClaimNumber	醫療保健	US
HealthInsuranceNumber	醫療保健	法國
HealthcareProcedureCode	醫療保健	US

識別符	類別	國家/地區/語言
IndividualTaxIdentificationNumber	個人	美國
InseeCode	個人	法國
MedicareBeneficiaryNumber	醫療保健	US
NationalDrugCode	醫療保健	US
NationalIdentificationNumber	個人	DE、ES、IT
NationalInsuranceNumber	個人	GB
NationalProviderId	醫療保健	US
NhsNumber	醫療保健	GB
Nienumber	個人	ES
NifNumber	個人	ES
PassportNumber	個人	CA、DE、ES、FR、GB、IT、US
PermanentResidenceNumber	個人	CA
PersonalHealthNumber	醫療保健	CA
PhoneNumber	個人	BR、DE、ES、FR、GB、IT、US
PostalCode	個人	CA
RgNumber	個人	BR
SocialInsuranceNumber	個人	CA
Ssn	個人	ES、US
TaxId	個人	DE、ES、FR、GB

識別符	類別	國家/地區/語言
ZipCode	個人	美國

與語言/區域無關的支援識別符

識別符	類別
Address	個人
AWSSecretKey	登入資料
CreditCardExpiration	金融
CreditCardNumber	金融
CreditCardSecurityCode	金融
EmailAddress	個人
IpAddress	個人
LatLong	個人
Name	個人
OpenSshPrivateKey	登入資料
PgpPrivateKey	登入資料
PkcsPrivateKey	登入資料
PuttyPrivateKey	登入資料
VehicleIdentificationNumber	個人

Amazon SNS 敏感資料類型：登入資料

下表列出並說明 Amazon SNS 可使用受管資料識別符偵測的憑證類型。

偵測類型	受管資料識別符 ID	必要的關鍵字	國家和地區
AWS 私密存取金鑰	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	任何
OpenSSH 私密金鑰	OpenSshPrivateKey	否	任何
PGP 私密金鑰	PgpPrivateKey	否	任何
公有金鑰密碼編譯標準 (PKCS) 私密金鑰	PkcsPrivateKey	否	任何
PuTTY 私密金鑰	PuttyPrivateKey	否	任何

憑證資料類型的資料識別符 ARN

以下列出您可新增至資料保護政策的資料識別符 Amazon Resource Name (ARN)。

憑證資料識別符 ARN

```
arn:aws:dataprotection::aws:data-identifier/AwsSecretKey
```

```
arn:aws:dataprotection::aws:data-identifier/OpenSshPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PgpPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PkcsPrivateKey
```

```
arn:aws:dataprotection::aws:data-identifier/PuttyPrivateKey
```

Amazon SNS 敏感資料類型：裝置

下表列出並說明 Amazon SNS 可使用受管資料識別符偵測到的裝置識別符類型。

偵測類型	受管資料識別符 ID	必要的關鍵字	國家和地區
IP Address (IP 地址)	IpAddress	否	任何

裝置資料類型的資料識別符 ARN

以下列出您可新增至資料保護政策的資料識別符 Amazon Resource Name (ARN)。

裝置資料識別符 ARN

```
arn:aws:dataprotection::aws:data-identifier/IpAddress
```

Amazon SNS 敏感資料類型：金融

下表列出並說明 Amazon SNS 可使用受管資料識別符偵測的財務資訊類型。

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
銀行帳戶號碼	BankAccountNumber BankAccountNumber-US	是，請參閱 銀行帳戶號碼的關鍵字 。	這包括：國際銀行帳戶號碼 (IBAN)，最多由 34 個英數字元組成，包括國家/地區代碼等元素。	法國、德國、義大利、西班牙、英國
信用卡到期日	CreditCardExpiration	exp d、exp m、exp y、expiration、expiry	–	任何
信用卡磁條資料	CreditCardMagneticStripe	是的，包括：信用卡資料、iso7813、mag、mag	這包括第 1 軌和第 2 軌。	任何

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
		stripe、stripe、swipe。		
信用卡號碼	CreditCardNumber	account number、american express、amex、bank card、card、card num、card number、cc #、ccn、check card、credit、credit card#、debit kort、debit t、debit card、diners club、discover、electron、electron verification code、japanese card bureau、jcb、mastercard、mc、pan、payment account number、payment card number、pcn、union pay、visa	偵測要求資料是符合 Luhn 檢查公式的 13-19 位數序列，並針對以下任何類型的信用卡使用標準卡號字首：American Express、Dankort、Diner's Club、Discover、Electron、Japanese Card Bureau (JCB)、Mastercard、UnionPay 和 Visa (1 下面的上標連結)。	任何

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
信用卡驗證碼	CreditCardSecurityCode	card id、card identification code、card identification number、card security code、card validation code、card validation number、card verification data、card verification value、cvc、cvc2、cvv、cvv2、elo verification code	–	任何

1. Amazon SNS 不會報告出現的以下序列，信用卡發卡機構已保留這些序列供公開測試使用：

1220000000000003、2222405343248877、2222990905257051、2223007648726984、22235771200176 和 76009244561。

銀行帳戶號碼的關鍵字

使用下列關鍵字來偵測最多由 34 個英數字元組成的國際銀行帳戶號碼 (IBAN)，包括國家/地區代碼等元素。

國家/地區或區域	關鍵字			
法國	account code, account number,			

國家/地區或區域	關鍵字			
	accountno #, accountnu mber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
德國	account code, account number, accountno #, accountnu mber#, bankleitz ahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartenum mer, kontonumm er, kreditkar tennummer, sepa			

國家/地區或區域	關鍵字			
義大利	account code, account number, accountno #, accountnu mber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			
西班牙	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			

國家/地區或區域	關鍵字			
英國	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			
美國	銀行帳戶、銀行 帳戶、支票帳 戶、支票帳戶、 存款帳戶、存款 帳戶、儲蓄帳 戶、儲蓄帳戶、 支票帳戶、支票 帳戶			

財務資料類型的資料識別符 ARN

以下列出您可新增至資料保護政策的資料識別符 Amazon Resource Name (ARN)。

財務資料識別符 ARN
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-DE
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-ES
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-FR
arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-GB

財務資料識別符 ARN

arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-IT

arn:aws:dataprotection::aws:data-identifier/BankAccountNumber-US

arn:aws:dataprotection::aws:data-identifier/CreditCardExpiration

arn:aws:dataprotection::aws:data-identifier/CreditCardNumber

arn:aws:dataprotection::aws:data-identifier/CreditCardSecurityCode

Amazon SNS 敏感資料類型：受保護的健康資訊 (PHI)

下表列出並說明 Amazon SNS 可使用受管資料識別符偵測到的受保護醫療資訊 (PHI) 類型。

偵測類型	受管資料識別符 ID	必要的關鍵字	國家和地區
緝毒署 (DEA) 註冊號碼	DrugEnforcementAgencyNumber	dea number, dea registration	美國
健康保險卡號碼 (EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero、carta assicurazione numero、carte d'assurance maladie、carte européenne d'assurance maladie、ceam、ehic、ehic#、finlandehicnumber#、gesundheitskarte、hälsokort、health card、health card number、health insurance card、health insurance number、in	歐盟

偵測類型	受管資料識別符 ID	必要的關鍵字	國家和地區
		urance card number、krankenvers icherungskarte、kra nkenversicherungs nummer、med ical account number、numero conto medico、nu méro d'assurance maladie、numéro de carte d'assuran ce、numéro de compte medical、n úmero de cuenta médica、número de seguro de salud、número de tarjeta de seguro、sa iraanhoitokortin、s airausvakuutuskort ti、sairausvakuutus numero、sjukförsäkr ing nummer、sj ukförsäkringskort、 suomi ehic-nume ro、tarjeta de salud、terveyskortt i、tessera sanitaria assicurazione numero、versicherun gsnummer	

偵測類型	受管資料識別符 ID	必要的關鍵字	國家和地區
健康保險索償編碼 (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	美國
健康保險或醫療識別號碼	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	法國
醫療保健通用程序編碼系統 (HCPCS) 代碼	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	美國
聯邦醫療保險受益人號碼 (MBN)	MedicareBeneficiaryNumber	mbi, medicare beneficiary	美國
國家藥物法規 (NDC)	NationalDrugCode	national drug code, ndc	美國
國家提供者識別符 (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npa	美國
國民保健署 (NHS) 號碼	NhsNumber	national health service, NHS	GB
個人健康號碼 (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

健康保險和醫療識別號碼的關鍵字

為了偵測各種類型的健康保險和醫療識別號碼，Amazon SNS 要求關鍵字與號碼相鄰。這包括歐洲健康保險卡號碼 (歐盟、芬蘭)、健康保險號碼 (法國)、聯邦醫療保險受益人識別碼 (美國)、國民保險號碼 (英國)、NHS 號碼 (英國) 和個人健康號碼 (加拿大)。

下表列出 Amazon SNS 在特定國家和地區辨識的關鍵字。

國家/地區或區域	關鍵字
加拿大	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
歐盟	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, krankenversicherungnummer, medical account number, numero conto medico, numéro d'assurance maladie, numéro de carte d'assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausva kuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
芬蘭	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance

國家/地區或區域	關鍵字
	number, sairaanhoitokortin, sairaanhoitokortin , sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
法國	carte d'assuré social, carte vitale, insurance card
英國	國民保健署 , NHS
美國	mbi, medicare beneficiary

受保護醫療資訊 (PHI) 資料類型的資料識別符 ARN

以下列出可用於 PHI 資料保護政策的資料識別符 Amazon Resource Name (ARN)。

PHI 資料識別符 ARN

```
arn:aws:dataprotection::aws:data-identifier/DrugEnforcementAgencyNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthcareProcedureCode-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceCardNumber-EU
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceClaimNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/HealthInsuranceNumber-FR
```

```
arn:aws:dataprotection::aws:data-identifier/MedicareBeneficiaryNumber-US
```

```
arn:aws:dataprotection::aws:data-identifier/NationalDrugCode-US
```

```
arn:aws:dataprotection::aws:data-identifier/NationalInsuranceNumber-GB
```

```
arn:aws:dataprotection::aws:data-identifier/NationalProviderId-US
```

```
arn:aws:dataprotection::aws:data-identifier/NhsNumber-GB
```

PHI 資料識別符 ARN

```
arn:aws:dataprotection::aws:data-identifier/PersonalHealthNumber-CA
```

Amazon SNS 敏感資料類型：個人身分識別資訊 (PII)

下表列出並說明 Amazon SNS 可使用受管資料識別符偵測的個人身分識別資訊 (PII) 類型。

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
出生日期	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	支援大多數日期格式，例如所有數字以及數字和月份名稱的組合。您可以用空格、斜線 (/) 或連字號 (-) 分隔日期組成部分。	任何
Código de Endereçamento Postal (CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	巴西
Cadastro Nacional da Pessoa Jurídica (CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	巴西
Cadastro de Pessoas Físicas (CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de	–	巴西

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
		pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf		
駕照識別號碼	DriversLicense	是，請參閱 駕照識別號碼的關鍵字。	–	澳洲、奧地利、 比利時、保加利亞、 加拿大、克羅埃西亞、 賽普勒斯、捷克、 丹麥、愛沙尼亞、 芬蘭、法國、德國、 希臘、匈牙利、 愛爾蘭、義大利、 拉脫維亞、立陶宛、 盧森堡、馬爾他、 荷蘭、波蘭、葡萄牙、 羅馬尼亞、斯洛伐克、 斯洛維尼亞、西班牙、 瑞典、英國、美國

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
選民名冊號碼	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	英國
個人納稅識別號碼	IndividualTaxIdentification Number	是，請參閱 納稅識別號碼及參考號碼的關鍵字 。	–	美國
國家統計和經濟研究所 (INSEE)	InseeCode	是，請參閱 國民身分證號碼的關鍵字 。	–	法國
國家身分證號碼	NationalIdentificationNumber	是，請參閱 國民身分證號碼的關鍵字 。	這包括 Documento Nacional de Identidad (DNI) 識別符 (西班牙)、Codice fiscale codes (義大利) 和國民身分證號碼 (德國)。	德國、義大利、西班牙

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
國民保險號碼 (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, national insurance#, national insurancenumber, nin, nino	–	英國
Número de identidad de extranjero (NIE)	NieNumber	是，請參閱 納稅識別號碼及參考號碼的關鍵字 。	–	西班牙
Número de Identificación Fiscal (NIF)	NifNumber	是，請參閱 納稅識別號碼及參考號碼的關鍵字 。	–	西班牙
護照號碼	PassportNumber	是，請參閱 護照號碼的關鍵字 。	–	加拿大、法國、德國、義大利、西班牙、英國、美國

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
永久居留號碼	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	–	加拿大

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
電話號碼	PhoneNumber	巴西：關鍵字還包括：cel、celular、fone、móvel、número residencial、numero residencial、telefone 其他：cell、contact、fax、fax number、mobile、phone、phone number、tel、telephone、telephone number	這包括美國免付費電話號碼和傳真號碼。如果關鍵字與資料相鄰，則該號碼不必包含國家/地區代碼。如果關鍵字不在資料附近，則該數字必須包含國家/地區代碼。	巴西、加拿大、法國、德國、義大利、西班牙、英國、美國
郵遞區號	PostalCode	No	—	加拿大
Registro Geral (RG)	RgNumber	是，請參閱 國民身分證號碼的關鍵字 。	—	巴西
社會保險號碼 (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	—	加拿大

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
社會安全號碼 (SSN)	Ssn	Spain – número de la seguridad social, social security no., social security no. número de la seguridad social, social security number, socialsecurityno#, ssn, ssn# 美國 - social security、ss#、ssn	–	西班牙、美國
納稅識別號碼或參考號碼	TaxId	是，請參閱 納稅識別號碼及參考號碼的關鍵字 。	這包括 TIN (法國)；Steueridentifikationsnummer (德國)；CIF (西班牙)；以及 TRN、UTR (英國)。	法國、德國、西班牙、英國
美國郵遞區號	ZipCode	zip code, zip+4	–	美國
郵寄地址	Address	No	雖然不需要使用關鍵字，但偵測地址需要包含城市或地點的名稱以及郵遞區號。	澳洲、加拿大、法國、德國、義大利、西班牙、英國、美國

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
電子郵件地址	EmailAddress	電子郵件，電子郵件地址，電子郵件，電子郵件地址	–	任何
全球定位系統 (GPS) 座標	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	如果緯度和經度座標以一對形式儲存，且使用十進位度 (DD) 格式 (例如 41.948614,-87.655311)，Amazon SNS 就可以偵測 GPS 座標。支援不包括度數十進位分鐘 (DDM) 格式的座標 (例如 41°56.9168'N 87°39.3187'W) 或度、分、秒 (DMS) 格式 (例如 41°56'55.0104"N 87°39'19.1196"W)。	任何
全名	Name	No	Amazon SNS 只能偵測全名。支援僅限於拉丁字元集。	任何

偵測類型	受管資料識別符 ID	必要的關鍵字	其他資訊	國家和地區
車輛識別符 (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 可 以偵測包含 17 個字元序列並 符合 ISO 3779 和 3780 標準的 VIN。這些標準是 專為全球使用而 設計的。	任何

駕照識別號碼的關鍵字

為了偵測各種類型的駕照識別號碼，Amazon SNS 要求關鍵字與這些號碼相鄰。下表列出 Amazon SNS 針對特定國家和地區辨識的關鍵字。

國家/地區或區域	關鍵字
澳洲	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

國家/地區或區域	關鍵字
奧地利	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
比利時	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
保加利亞	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
加拿大	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
克羅埃西亞	vozačka dozvola
賽普勒斯	άδεια οδήγησης
捷克	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský průkaz, řidičský průkaz
丹麥	kørekort, kørekortnummer

國家/地區或區域	關鍵字
愛沙尼亞	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
芬蘭	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
法國	permis de conduire
德國	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, fuhrerschein, fuhrerschein- nr, fuhrerschein- nr, fuhrersch einnummer, fuhrerscheinnummer
希臘	δεια οδήγησης, adeia odigisis
匈牙利	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
愛爾蘭	ceadúnas tiomána
義大利	patente di guida, patente di guida numero, patente guida, patente guida numero
拉脫維亞	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
立陶宛	vairuotojo pažymėjimas
盧森堡	fahrerlaubnis, fuhrerschäin
馬爾他	liċenzja tas-sewqan
荷蘭	permis de conduire, rijbewijs, rijbewijsnummer

國家/地區或區域	關鍵字
波蘭	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
葡萄牙	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
羅馬尼亞	numărul permisului de conducere, permis de conducere
斯洛伐克	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
斯洛維尼亞	vozniško dovoljenje
西班牙	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
瑞典	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

國家/地區或區域	關鍵字
英國	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
美國	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

國民身分證號碼的關鍵字

若要偵測各種類型的國民身分證號碼，Amazon SNS 要求關鍵字與數字相鄰。這包括 Documento Nacional de Identidad (DNI) 識別符 (西班牙)、法國國家統計和經濟研究所 (INSEE) 代碼、德國國民身分證號碼和 Registro Geral (RG) 號碼 (巴西)。

下表列出 Amazon SNS 針對特定國家和地區辨識的關鍵字。

國家/地區或區域	關鍵字
巴西	registro geral, rg
法國	assurance sociale, carte nationale d'identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité

國家/地區或區域	關鍵字
	sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
德國	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
義大利	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
西班牙	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#

護照號碼的關鍵字

若要偵測各種類型的護照號碼，Amazon SNS 要求關鍵字與號碼相鄰。下表列出 Amazon SNS 針對特定國家和地區辨識的關鍵字。

國家/地區或區域	關鍵字
加拿大	pasport, pasport#, passport, passport#, passportno, passportno#
法國	numéro de pasport, pasport, pasport #, pasport #, pasportn °, pasport n °, pasportNon, pasport non

國家/地區或區域	關鍵字
德國	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass-nr, reisepassnummer
義大利	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
西班牙	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
英國	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
美國	passport, travel document

納稅識別號碼及參考號碼的關鍵字

為了偵測各種類型的納稅識別號碼和參考號碼，Amazon SNS 要求關鍵字與這些號碼相鄰。下表列出 Amazon SNS 針對特定國家和地區辨識的關鍵字。

國家/地區或區域	關鍵字
巴西	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 jurídica, cnpj, cpf
法國	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#

國家/地區或區域	關鍵字
德國	identifikationsnummer, steuer id, steueridentifikationsnummer, steuernummer, tax id, tax identification number, tax number
西班牙	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#
英國	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
美國	個人納稅人識別號碼, itin, i.t.i.n.

個人身分識別資訊 (PII) 的資料識別符 ARN

下表列出您可新增至資料保護政策的資料識別符 Amazon Resource Name (ARN)。

PII 資料識別符 ARN

arn:aws:dataprotection::aws:data-identifier/Address

arn:aws:dataprotection::aws:data-identifier/CepCode-BR

arn:aws:dataprotection::aws:data-identifier/Cnpj-BR

arn:aws:dataprotection::aws:data-identifier/CpfCode-BR

arn:aws:dataprotection::aws:data-identifier/DateOfBirth

arn:aws:dataprotection::aws:data-identifier/DriversLicense-AT

PII 資料識別符 ARN

arn:aws:dataprotection::aws:data-identifier/DriversLicense-AU

arn:aws:dataprotection::aws:data-identifier/DriversLicense-BE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-BG

arn:aws:dataprotection::aws:data-identifier/DriversLicense-CA

arn:aws:dataprotection::aws:data-identifier/DriversLicense-CY

arn:aws:dataprotection::aws:data-identifier/DriversLicense-CZ

arn:aws:dataprotection::aws:data-identifier/DriversLicense-DE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-DK

arn:aws:dataprotection::aws:data-identifier/DriversLicense-EE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-ES

arn:aws:dataprotection::aws:data-identifier/DriversLicense-FI

arn:aws:dataprotection::aws:data-identifier/DriversLicense-FR

arn:aws:dataprotection::aws:data-identifier/DriversLicense-GB

arn:aws:dataprotection::aws:data-identifier/DriversLicense-GR

arn:aws:dataprotection::aws:data-identifier/DriversLicense-HR

arn:aws:dataprotection::aws:data-identifier/DriversLicense-HU

arn:aws:dataprotection::aws:data-identifier/DriversLicense-IE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-IT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-LT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-LU

PII 資料識別符 ARN

arn:aws:dataprotection::aws:data-identifier/DriversLicense-LV

arn:aws:dataprotection::aws:data-identifier/DriversLicense-MT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-NL

arn:aws:dataprotection::aws:data-identifier/DriversLicense-PL

arn:aws:dataprotection::aws:data-identifier/DriversLicense-PT

arn:aws:dataprotection::aws:data-identifier/DriversLicense-RO

arn:aws:dataprotection::aws:data-identifier/DriversLicense-SE

arn:aws:dataprotection::aws:data-identifier/DriversLicense-SI

arn:aws:dataprotection::aws:data-identifier/DriversLicense-SK

arn:aws:dataprotection::aws:data-identifier/DriversLicense-US

arn:aws:dataprotection::aws:data-identifier/ElectoralRollNumber-GB

arn:aws:dataprotection::aws:data-identifier/EmailAddress

arn:aws:dataprotection::aws:data-identifier/IndividualTaxIdentificationNumber-US

arn:aws:dataprotection::aws:data-identifier/InseeCode-FR

arn:aws:dataprotection::aws:data-identifier/LatLong

arn:aws:dataprotection::aws:data-identifier/Name

arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-DE

arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-ES

arn:aws:dataprotection::aws:data-identifier/NationalIdentificationNumber-IT

arn:aws:dataprotection::aws:data-identifier/NieNumber-ES

PII 資料識別符 ARN

arn:aws:dataprotection::aws:data-identifier/NifNumber-ES

arn:aws:dataprotection::aws:data-identifier/PassportNumber-CA

arn:aws:dataprotection::aws:data-identifier/PassportNumber-DE

arn:aws:dataprotection::aws:data-identifier/PassportNumber-ES

arn:aws:dataprotection::aws:data-identifier/PassportNumber-FR

arn:aws:dataprotection::aws:data-identifier/PassportNumber-GB

arn:aws:dataprotection::aws:data-identifier/PassportNumber-IT

arn:aws:dataprotection::aws:data-identifier/PassportNumber-US

arn:aws:dataprotection::aws:data-identifier/PermanentResidenceNumber-CA

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-BR

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-DE

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-ES

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-FR

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-GB

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-IT

arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US

arn:aws:dataprotection::aws:data-identifier/PostalCode-CA

arn:aws:dataprotection::aws:data-identifier/RgNumber-BR

arn:aws:dataprotection::aws:data-identifier/SocialInsuranceNumber-CA

arn:aws:dataprotection::aws:data-identifier/Ssn-ES

PII 資料識別符 ARN

```
arn:aws:dataprotection::aws:data-identifier/Ssn-US
```

```
arn:aws:dataprotection::aws:data-identifier/TaxId-DE
```

```
arn:aws:dataprotection::aws:data-identifier/TaxId-ES
```

```
arn:aws:dataprotection::aws:data-identifier/TaxId-FR
```

```
arn:aws:dataprotection::aws:data-identifier/TaxId-GB
```

```
arn:aws:dataprotection::aws:data-identifier/VehicleIdentificationNumber
```

```
arn:aws:dataprotection::aws:data-identifier/ZipCode-US
```

在 Amazon SNS 中使用自訂資料識別符

自訂資料識別符 (CDI) 可讓您定義自訂的規則運算式，以用於資料保護政策。使用自訂資料識別符就可以鎖定[受管資料識別符](#)無法提供的企業特定個人身分識別資訊 (PII) 使用案例。例如，您可以使用自訂資料識別符來尋找公司專屬員工 ID。自訂資料識別符可與受管資料識別符搭配使用。

什麼是 SNS 自訂資料識別符？

自訂資料識別符 (CDI) 可讓您定義自訂的規則運算式，以用於資料保護政策。使用自訂資料識別符就可以鎖定[受管資料識別符](#)無法提供的企業特定個人身分識別資訊 (PII) 使用案例。例如，您可以使用自訂資料識別符來尋找公司專屬員工 ID。自訂資料識別符可與受管資料識別符搭配使用。

在您的資料保護政策中使用自訂資料識別符

下列資料保護政策會指示 Amazon SNS 主題偵測內含公司專屬員工 ID 的承載，然後使用雜湊符號 (#) 為這些 ID 加上遮罩。

1. 在您的資料保護政策內建立 Configuration 區塊。
2. 輸入自訂資料識別符的 Name。例如：**EmployeeId**。
3. 輸入自訂資料識別符的 Regex。例如：**EID-\d{9}-US**。
4. 請參閱政策聲明中的下列自訂資料識別符。

```
{
```

```

    "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. (選用) 視需要繼續將其他自訂資料識別符新增至 Configuration 區塊。資料保護政策目前最多可支援 10 個自訂資料識別符。

自訂資料識別符的限制

Amazon SNS 自訂資料識別符設有下列限制：

- 每個資料保護政策最多可支援 10 個自訂資料識別符。
- 自訂資料識別符名稱的長度上限為 128 個字元。支援的字元如下：
 - 英數字：(a-zA-Z0-9)
 - 符號：('_' | '-')
- RegEx 的長度上限為 200 個字元。支援的字元如下：
 - 英數字：(a-zA-Z0-9)
 - 符號：('_' | '#' | '=' | '@' | '/' | ';' | ':' | '-' | '|')

- RegEx 保留字元 : ('^'|'\$' | '?' | '[' | ']' | '{' | '}' | '|' | '\' | '*' | '+' | '.')
- 自訂資料識別符的名稱不可與受管資料識別符的名稱相同。
- 每個 Amazon SNS 主題的每個資料保護政策中，都必須指定自訂資料識別符。

Amazon SNS 訊息交付

本主題說明 Amazon SNS 如何處理各種案例的訊息傳遞。您將了解原始訊息傳遞，其中 Amazon SNS 會以原始、未修改的格式將訊息傳遞至端點。您也將探索如何從 Amazon SNS 主題傳送訊息到不同 Amazon SQS 佇列 AWS 帳戶，提供跨帳戶訊息的深入見解。

本主題提供將 Amazon SNS 訊息交付至不同 Amazon SQS 佇列或 Lambda 函數 AWS 區域的資訊、跨區域交付的運作方式，以及涉及的考量。

此外，您將了解如何監控和解釋訊息傳遞狀態，這可提供訊息是否成功傳遞或遇到問題的關鍵資訊。在訊息傳遞失敗的情況下，您將了解訊息傳遞重試程序，包括 Amazon SNS 如何自動嘗試重新傳遞訊息，以確保訊息到達其預期目的地。本主題也討論使用無效字母佇列來擷取多次嘗試後無法傳遞的訊息，讓您有效地分析和疑難排解這些失敗。

Amazon SNS 原始訊息交付

為了避免 [Amazon Data Firehose](#)、[Amazon SQS](#) 和 [HTTP/S](#) 端點處理訊息的 JSON 格式，Amazon SNS 允許原始訊息傳遞：

- 當您為 Amazon Data Firehose 或 Amazon SQS 端點啟用原始訊息交付時，任何 Amazon SNS 中繼資料都會從已發佈的訊息中分割，並依原狀傳送訊息。
- 當您啟用 HTTP/S 端點的原始訊息交付時，值設為 `x-amz-sns-rawdelivery` 的 HTTP 標頭 `true` 會加入至訊息中，表示該訊息已在沒有 JSON 格式設定的情況下發佈。
- 當您啟用 HTTP/S 端點的原始訊息交付時，會交付訊息本文、用戶端 IP 和所需的標頭。當您指定訊息屬性時，系統不會傳送該訊息。
- 當您為 Firehose 端點啟用原始訊息交付時，訊息內文即會交付。當您指定訊息屬性時，系統不會傳送該訊息。

若要使用 AWS SDK 啟用原始訊息傳遞，您必須使用 `SetSubscriptionAttribute` API 動作，並將 `RawMessageDelivery` 屬性的值設定為 `true`。

使用 AWS Management Console 啟用原始訊息交付

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。

3. 在主題頁面上，選擇訂閱 Firehose、Amazon SQS 或 HTTP/S 端點的主題。
4. 在 **MyTopic (####)** 頁面上，於 Subscription (訂閱) 區段內，選擇訂閱並選擇 Edit (編輯)。
5. 在 Edit **EXAMPLE1-23bc-4567-d890-ef12g3hij456** (編輯範例1-23bc-4567-d890-ef12g3hij456) 頁面上，於 Details (詳細資訊) 區段內，選擇 Enable raw message delivery (啟用原始訊息交付)。
6. 選擇 Save changes (儲存變更)。

訊息格式範例

在下列範例中，相同的訊息會傳送到相同的 Amazon SQS 佇列兩次。唯一的差異是第一封郵件的原始訊息交付會停用，第二封郵件則會啟用。

- 原始訊息交付停用

```
{
  "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/
    zLTnMyIxEzmi2X1sh0BWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAIkP9jstZzDRm
    +bcVs8+T0yaLiEGLrIIIL4esi111hIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
    rd5071X1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNk1VyPfy
    +7Ta1MD0lzmJu0rExtnSIbZew3foxgx8GT+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"
}
```

- 原始訊息交付啟用

```
This is a test message.
```

Amazon SQS 訂閱的訊息屬性和原始訊息傳遞

Amazon SNS 支援訊息屬性的傳遞，可讓您提供有關訊息的結構化中繼資料項目，例如時間戳記、地理空間資料、簽章和識別符。對於啟用原始訊息交付的 Amazon SQS 訂閱，最多可傳送 10 個訊息屬性。若要傳送超過 10 個訊息屬性，您必須停用原始訊息傳遞。不過，Amazon SNS 會捨棄具有超過 10 個訊息屬性的訊息，這些屬性導向已啟用原始訊息交付的 Amazon SQS 訂閱，並將它們視為用戶端錯誤。

傳送 Amazon SNS 訊息至其他帳戶中的 Amazon SQS 佇列

本文件說明如何將通知發佈至其他帳戶中具有一個或多個 Amazon SQS 佇列訂閱的 Amazon SNS 主題。如果主題和佇列在相同帳戶中，您要以相同方式設定主題和佇列 (請參閱[將 Amazon SNS 通知發送至 Amazon SQS 佇列以進行非同步處理](#))。主要差異是您處理訂閱確認的方式，而這取決於您如何將佇列訂閱至主題。

最佳實務是在可能的情況下按照[佇列擁有者建立訂閱](#)一節中所述的步驟進行，因為當佇列擁有者建立訂閱時會自動確認。

Note

如果 Amazon SQS 佇列具有大量訊息，建議佇列擁有者建立訂閱。

佇列擁有者建立訂閱

建立 Amazon SQS 佇列的帳戶是佇列擁有者。當佇列擁有者建立訂閱，訂閱並不需要確認。Subscribe 動作一完成，佇列就會開始接收來自主題的通知。若要讓佇列擁有者訂閱主題擁有者的主題，主題擁有者必須提供佇列擁有者許可，以呼叫主題的 Subscribe 動作。

步驟 1：使用 AWS Management Console 設定主題政策

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。
3. 選取主題，然後選擇 Edit (編輯)。
4. 在 Edit **MyTopic** (編輯 MyTopic) 頁面上，展開存取政策區段。
5. 輸入下列政策：

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

此政策授予帳戶 111122223333 呼叫帳戶 123456789012 中 MyTopic 上之 sns:Subscribe 的許可。

擁有帳戶 111122223333 憑證的使用者可以訂閱 MyTopic。這項許可允許帳戶 ID 將許可委派給其 IAM 使用者/角色。唯有根帳戶或管理員使用者可以呼叫 sns:Subscribe。IAM 使用者/角色也必須擁有 sns:subscribe 才能允許其佇列訂閱。

6. 選擇 Save changes (儲存變更)。

擁有帳戶 111122223333 的登入資料的使用者可以訂閱 MyTopic。

步驟 2：AWS 帳戶 使用 將 Amazon SQS 佇列訂閱新增至另一個 中的主題 AWS Management Console

開始之前，請確定您有主題和佇列的 ARN 且您已[授予主題可傳送訊息至佇列的許可](#)。

1. 請登入 [Amazon SQS 主控台](#)。
2. 在導覽面板中選擇 Queues (佇列)。
3. 在佇列清單中選擇要訂閱 Amazon SNS 主題的 佇列。
4. 選擇 Subscribe to Amazon SNS topic (訂閱 Amazon SNS 主題)。
5. 從 Specify an Amazon SNS topic available for this queue menu (為此佇列選單指定可用的 Amazon SNS 主題) 中，為您的佇列選擇 Amazon SNS topic (Amazon SNS 主題)。
6. 選擇 Enter Amazon SNS topic ARN (輸入 Amazon SNS 主題 ARN)，然後輸入主題的 Amazon Resource Name (ARN)。
7. 選擇 Save (儲存)。

 Note

- 若要能夠與服務進行通訊，佇列必須擁有 Amazon SNS 的許可。
- 如果您是該佇列的擁有者，則無須確認訂閱。

沒有擁有佇列的使用者建立訂閱

建立訂閱但不是佇列擁有者的任何使用者皆須確認訂閱。

當您使用 `Subscribe` 動作時，Amazon SNS 會傳送訂閱確認至佇列。訂閱會在 Amazon SNS 主控台中顯示，且其訂閱 ID 設為 `Pending Confirmation` (訂閱確認)。

若要確認訂閱，具備可讀取佇列中訊息之許可的使用者必須擷取訂閱確認 URL，且訂閱擁有者必須使用訂閱確認 URL 來確認訂閱。直到確認訂閱完成，都不會有發佈至主題的通知傳送至佇列。若要確認訂閱，您可以使用 Amazon SQS 主控台或 [ReceiveMessage](#) 動作。

 Note

讓端點訂閱主題之前，請設定佇列的 `sqs:SendMessage` 許可，以確定佇列可以接收主題的訊息。如需詳細資訊，請參閱[步驟 2：將許可提供給 Amazon SNS 主題，以將訊息傳送至 Amazon SQS 佇列。](#)。

步驟 1：AWS 帳戶 使用 將 Amazon SQS 佇列訂閱新增至另一個 中的主題 AWS Management Console

開始之前，請確定您有主題和佇列的 ARN 且您已[授予主題可傳送訊息至佇列的許可](#)。

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 `Subscriptions` (訂閱)。
3. 在訂閱頁面，選擇建立訂閱。
4. 在建立訂閱頁面上，於詳細資訊區段中，執行以下作業：
 - a. 在 `Topic ARN` (主題 ARN) 中，輸入主題的 ARN。
 - b. 在 `Protocol` (通訊協定) 中，選擇 Amazon SQS。

- c. 針對 Endpoint (端點)，輸入佇列的 ARN。
- d. 選擇 Create subscription (建立訂閱)。

 Note

- 若要能夠與服務進行通訊，佇列必須擁有 Amazon SNS 的許可。

以下範例政策陳述式允許 Amazon SNS 主題將訊息傳送至 Amazon SQS 佇列。

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

步驟 2：使用 確認訂閱 AWS Management Console

1. 請登入 [Amazon SQS 主控台](#)。
2. 選取具有待訂閱主題的佇列。
3. 選擇 Send and receive messages (傳送和接收訊息)，然後選擇 Poll for messages (訊息輪詢)。

有訂閱確認的訊息會在佇列中收到。

4. 在 Body (本文) 欄位中，執行下列操作：
 - a. 選擇 More Details (更多詳細資訊)。
 - b. 在 Message Details (訊息詳細資訊) 對話方塊中，找到並記下 SubscribeURL 值。這是您的訂閱連結 (如下方範例)。有關 API 字符驗證的其他詳細資料，請參閱《Amazon SNS API 參考》中的 [ConfirmSubscription](#)。

```
https://sns.us-west-2.amazonaws.com/?  
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-  
east-2:123456789012:MyTopic&Token=2336412f37fb...
```

- c. 記下訂閱確認連結。URL 必須從佇列擁有者傳遞給訂閱擁有者。訂閱擁有者必須將 URL 輸入到 [Amazon SNS 主控台](#)。
5. 以訂閱擁有者身分登入 [Amazon SNS 主控台](#)。訂閱擁有者執行確認。
6. 選擇相關主題。
7. 選擇主題訂閱清單表格中相關的訂閱。標記為「Pending confirmation (待確認)」。
8. 選擇確認訂閱 (Confirm subscription)。
9. 出現一個模式，提示訂閱確認連結。貼上訂閱確認連結。
10. 選取模式中的 Confirm subscription (確認訂閱)。

XML 回應隨即顯示，例如：

```
<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>
```

訂閱的佇列已準備好接收來自主題的訊息。

11. 如果您在 Amazon SNS 主控台中檢視主題訂閱，您現在將會看到該訂閱 ARN 取代 Subscription ID (訂閱 ID) 欄位中的 Pending Confirmation (待確認中) 訊息。

如何強制訂閱以要求對取消訂閱請求進行身分驗證？

訂閱擁有者必須在訂閱確認時將 `AuthenticateOnUnsubscribe` 標記設定為 `true`。

- 當佇列擁有者建立訂閱時，`AuthenticateOnUnsubscribe` 將自動設定為 `True`。
- 當瀏覽至訂閱確認連結而不進行身分驗證時，無法將 `AuthenticateOnUnsubscribe` 設定為 `True`。

將 Amazon SNS 訊息傳送至 Amazon SQS 佇列或位於不同區域的 AWS Lambda 函數

Amazon SNS 支援跨區域交付，適用於預設啟用的區域和[加入區域](#)。如需 Amazon SNS 支援的 AWS 區域 (包括選擇加入區域) 的目前清單，請參閱 Amazon Web Services 一般參考 中的 [Amazon Simple Notification Service 端點和配額](#)。

Amazon SNS 支援跨區域傳送通知至 Amazon SQS 佇列，以及 AWS Lambda 函數。當其中一個區域是選擇加入區域時，您必須在訂閱資源的政策中指定不同的 Amazon SNS 服務主體。

Amazon SNS 訂閱命令必須在 Amazon SNS 託管所在的區域，在對應的區域中執行。例如，如果 Amazon SNS 在 us-east-1 區域的帳戶「A」中，而 Lambda 函數在 us-east-2 區域的帳戶「B」中，則必須在 us-east-1 區域的帳戶「A」中執行訂閱 CLI 命令。

選擇加入區域

Amazon SNS 支援下列選擇加入區域：

區域名稱	區域
非洲 (開普敦) 區域	af-south-1
亞太區域 (香港) 區域	ap-east-1
亞太區域 (海德拉巴)	ap-south-2
亞太區域 (雅加達)	ap-southeast-3
亞太區域 (墨爾本) 區域	ap-southeast-4
Europe (Milan) Region	eu-south-1
歐洲 (西班牙) 區域	eu-south-2
歐洲 (蘇黎世) 區域	eu-central-2
以色列 (特拉維夫) 區域	il-central-1
Middle East (Bahrain) Region	me-south-1

區域名稱	區域
中東 (阿拉伯聯合大公國) 區域	me-central-1

如需啟用選擇加入區域的資訊，請參閱《》中的[管理 AWS 區域](#) Amazon Web Services 一般參考。

當您使用 Amazon SNS 將訊息從選擇加入區域傳遞至預設啟用的區域時，您必須變更為佇列建立的資源政策。將委託人 `sns.amazonaws.com` 取代為 `sns.<opt-in-region>.amazonaws.com`。例如：

- 例如，如果您想要讓美國東部 (維吉尼亞北部) 的 Amazon SQS 佇列訂閱亞太區域 (香港) 的 Amazon SNS 主題，請將佇列政策中的委託人變更為 `sns.ap-east-1.amazonaws.com`。選擇加入區域包括 2019 年 3 月 20 日後推出的任何區域，其中包括亞太區域 (香港)、亞太區域 (雅加達)、中東 (巴林)、歐洲 (米蘭) 和非洲 (開普敦)。2019 年 3 月 20 日之前推出的區域為預設啟用。

跨區域遞送支援至 Amazon SQS

跨區域遞送	支援/不支援
預設啟用區域至選擇加入區域	支援在佇列的服務主體使用 <code>sns.<opt-in-region>.amazonaws.com</code>
選擇加入區域至預設啟用區域	支援在佇列的服務主體使用 <code>sns.<opt-in-region>.amazonaws.com</code>
選擇加入區域至選擇加入區域	不支援

以下的存取政策陳述式範例允許選擇加入區域 (af-south-1) 中的 Amazon SNS 主題遞送至預設啟用區域 (us-east-1) 中的 Amazon SQS 佇列。它在路徑 `Statement/Principal/Service` 下方包含必要的區域化服務主體組態。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
```



```

    "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.af-south-1.amazonaws.com"
    },
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
      }
    }
  },
  ...
]
}

```

- 若要訂閱美國東部（維吉尼亞北部）的 AWS Lambda 函數至亞太區域（香港）的 Amazon SNS 主題，請將 AWS Lambda 函數政策中的主體變更為 `sns.ap-east-1.amazonaws.com`。選擇加入區域包括 2019 年 3 月 20 日後推出的任何區域，其中包括亞太區域 (香港)、亞太區域 (雅加達)、中東 (巴林)、歐洲 (米蘭) 和非洲 (開普敦)。2019 年 3 月 20 日之前推出的區域為預設啟用。

跨區域交付支援至 AWS Lambda

跨區域遞送	支援/不支援	
預設啟用區域至選擇加入區域	不支援	
選擇加入區域至預設啟用區域	支援在 Lambda 函數的服務主體使用 <code>sns.<opt-in-region>.amazonaws.com</code>	
選擇加入區域至選擇加入區域	不支援	

Amazon SNS 訊息傳遞狀態

Amazon SNS 支援使用下列 Amazon SNS 端點記錄傳送至主題的通知訊息交付狀態：

- Amazon Data Firehose

- Amazon Simple Queue Service
- AWS Lambda
- HTTPS
- 平台應用程式端點

交付狀態日誌會傳送至 Amazon CloudWatch Logs，提供訊息交付操作的洞見。這些日誌可協助您：

- 判斷訊息是否已成功交付至端點。
- 識別從端點到 Amazon SNS 的回應。
- 測量訊息駐留時間（發佈時間戳記和遞交至端點之間的時間）。

您可以使用 AWS Management Console、AWS SDKs、查詢 API 或來設定交付狀態記錄 AWS CloudFormation。

交付狀態記錄的先決條件

本主題概述讓 Amazon SNS 將交付日誌寫入 CloudWatch 的必要 IAM 許可，並說明預設日誌群組命名慣例。這可確保您有正確的設定和存取，以監控和分析 CloudWatch 日誌中的訊息傳遞日誌。

必要的 IAM 許可

附加用於交付狀態記錄的 IAM 角色必須包含下列許可，才能讓 Amazon SNS 寫入 CloudWatch Logs。您可以使用具有這些許可的現有角色，或在設定期間建立新的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

日誌群組命名慣例

根據預設，Amazon SNS 會使用下列命名慣例，為交付狀態日誌建立 CloudWatch 日誌群組。此群組內的日誌串流對應至端點通訊協定（例如 Lambda、Amazon SQS）。請確定您有在 CloudWatch Logs 主控台中檢視這些日誌的許可。

```
sns/<region>/<account-id>/<topic-name>
```

使用 AWS Management Console 設定交付狀態記錄日誌

本主題說明如何啟用 Amazon SNS 主題的訊息交付狀態記錄，包括設定記錄設定、指派 IAM 角色，以及驗證 CloudWatch Logs 擷取交付日誌以進行監控和故障診斷。

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。
3. 選取所需的主題，然後選擇編輯。
4. 展開交付狀態記錄區段。
5. 選擇您要啟用記錄的通訊協定（例如 HTTP、Lambda、Amazon SQS）。
6. 輸入成功範例速率，這是您要接收 CloudWatch Logs 之成功訊息的百分比。
7. 在 IAM 角色區段中，您必須為成功和失敗記錄設定角色：
 - 使用現有的服務角色 – 選取具有 Amazon SNS 將日誌寫入 CloudWatch 所需許可的現有 IAM 角色。
 - 建立新的服務角色 – 選擇建立新的角色，在 IAM 主控台中定義成功和失敗交付的 IAM 角色。如需許可詳細資訊，請參閱 [交付狀態記錄的先決條件](#)。
8. 選擇儲存變更。

啟用記錄後，您可以檢視和剖析包含訊息傳遞狀態的 CloudWatch Logs。如需使用 CloudWatch 的詳細資訊，請參閱 [CloudWatch 文件](#)。

驗證日誌設定

1. 登入 CloudWatch Logs 主控台。
2. 找到名為 `sns/<region>/<account-id>/<topic-name>` 的日誌群組。

3. 確保已設定的端點通訊協定存在日誌串流。
4. 傳送測試訊息到您的主題，並確認日誌項目出現，表示傳送成功或失敗。

使用 AWS SDKs 設定交付狀態記錄

AWS SDKs 提供多種語言 APIs，以設定訊息傳遞狀態記錄的主題屬性。例如，使用 [SetTopicAttributes](#) API 來設定：

- `LambdaSuccessFeedbackRoleArn` – IAM 角色，用於成功將訊息傳遞至 Lambda 端點。
- `LambdaSuccessFeedbackSampleRate` – 成功將訊息傳送至 Lambda 端點的取樣率。
- `LambdaFailureFeedbackRoleArn` – 用於將失敗訊息傳遞至 Lambda 端點的 IAM 角色。

範例 AWS CLI 命令

```
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
```

主題屬性

針對訊息傳遞狀態使用下列主題屬性名稱值：

HTTP

- `HTTPSuccessFeedbackRoleArn` – 訂閱 HTTP 端點之 Amazon SNS 主題的訊息傳遞狀態成功。
- `HTTPSuccessFeedbackSampleRate` – 訂閱 HTTP 端點的 Amazon SNS 主題範例成功訊息的百分比。
- `HTTPFailureFeedbackRoleArn` – 訂閱 HTTP 端點之 Amazon SNS 主題的訊息傳遞狀態失敗。

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn` – 訂閱 Amazon Kinesis Data Firehose 端點的 Amazon SNS 主題成功訊息傳遞狀態。 Amazon Kinesis Data Firehose
- `FirehoseSuccessFeedbackSampleRate` – 訂閱 Amazon Kinesis Data Firehose 端點之 Amazon SNS 主題的取樣成功訊息百分比。 Amazon Kinesis Data Firehose

- `FirehoseFailureFeedbackRoleArn` – 訂閱 Amazon Kinesis Data Firehose 端點之 Amazon SNS 主題的訊息傳遞狀態失敗。 Amazon Kinesis Data Firehose

AWS Lambda

- `LambdaSuccessFeedbackRoleArn` – 訂閱 Lambda 端點的 Amazon SNS 主題成功訊息傳遞狀態。
- `LambdaSuccessFeedbackSampleRate` – 訂閱 Lambda 端點之 Amazon SNS 主題的成功訊息範例百分比。
- `LambdaFailureFeedbackRoleArn` – 訂閱 Lambda 端點之 Amazon SNS 主題的訊息傳遞狀態失敗。

平台應用程式端點

- `ApplicationSuccessFeedbackRoleArn` – 訂閱 AWS 應用程式端點之 Amazon SNS 主題的訊息傳遞狀態成功。
- `ApplicationSuccessFeedbackSampleRate` – 訂閱 AWS 應用程式端點之 Amazon SNS 主題的成功訊息範例百分比。
- `ApplicationFailureFeedbackRoleArn` – 訂閱 AWS 應用程式端點之 Amazon SNS 主題的訊息傳遞狀態失敗。

Note

此外，您可以設定應用程式屬性直接記錄交付狀態，以推送通知服務。如需詳細資訊，請參閱[使用訊息傳遞狀態的 Amazon SNS 應用程式屬性](#)。

Amazon SQS

- `SQSSuccessFeedbackRoleArn` – 訂閱 Amazon SQS 端點的 Amazon SNS 主題成功訊息傳遞狀態。 Amazon SQS
- `SQSSuccessFeedbackSampleRate` – 訂閱 Amazon SQS 端點的 Amazon SNS 主題範例成功訊息的百分比。 Amazon SQS
- `SQSFailureFeedbackRoleArn` – 訂閱 Amazon SQS 端點之 Amazon SNS 主題的訊息傳遞狀態失敗。 Amazon SQS

平台應用程式端點的日誌會寫入與其他端點相同的 CloudWatch Logs 群組。

 Note

<ENDPOINT>SuccessFeedbackRoleArn 和 <ENDPOINT>FailureFeedbackRoleArn 屬性是用來提供 Amazon SNS 寫入存取權，以代表您使用 CloudWatch Logs。<ENDPOINT>SuccessFeedbackSampleRate 屬性用於指定成功傳送訊息的取樣率百分比 (0-100)。在您設定 <ENDPOINT>FailureFeedbackRoleArn 屬性後，則所有傳送失敗的訊息都會產生 CloudWatch Logs。

AWS 用於設定主題屬性的 SDK 範例

下列程式碼範例示範如何使用 SetTopicAttributes。

CLI

AWS CLI

設定主題的屬性

下列 set-topic-attributes 範例會設定指定主題的 DisplayName 屬性。

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name DisplayName \  
  --attribute-value MyTopicDisplayName
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetTopicAttributes](#)。

Java

SDK for Java 2.x

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [SetTopicAttributes](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。


```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。

- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [SetTopicAttributes](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun setTopAttr(
    attribute: String?,
    topicArnVal: String?,
    value: String?,
) {
    val request =
        SetTopicAttributesRequest {
            attributeName = attribute
            attributeValue = value
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 SetTopicAttributes。

PHP

SDK for PHP

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());  
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [SetTopicAttributes](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
# 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}.")  
  end  
end
```

```

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

# 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

```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-enable-resource.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 [SetTopicAttributes](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [SetTopicAttributes](#)。

使用 AWS CloudFormation 設定交付狀態記錄日誌

若要 DeliveryStatusLogging 使用 設定 AWS CloudFormation，請使用 JSON 或 YAML 範本來建立 AWS CloudFormation 堆疊。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 `AWS::SNS::Topic` 資源的 DeliveryStatusLogging 屬性。以下是 JSON 和 YAML 中建立新主題或更新具有 Amazon SQS 通訊協定所有 DeliveryStatusLogging 屬性之現有主題的 AWS CloudFormation 範本範例。

確保 中參考SuccessFeedbackRoleArn的 IAM 角色FailureFeedbackRoleArn具有所需的 CloudWatch Logs 許可。

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
```

Amazon SNS 訊息傳遞重試

Amazon SNS 會為每個傳遞通訊協定定義傳遞政策。傳遞政策會定義在伺服器端發生錯誤時，Amazon SNS 重試傳遞訊息的方式 (在託管訂閱端點的系統無法使用時)。當傳遞政策用盡時，除非已將無效字母佇列附加至訂閱，否則 Amazon SNS 會停止重試傳遞並捨棄訊息。如需詳細資訊，請參閱[Amazon SNS 無效字母佇列](#)。

傳遞通訊協定和政策

Note

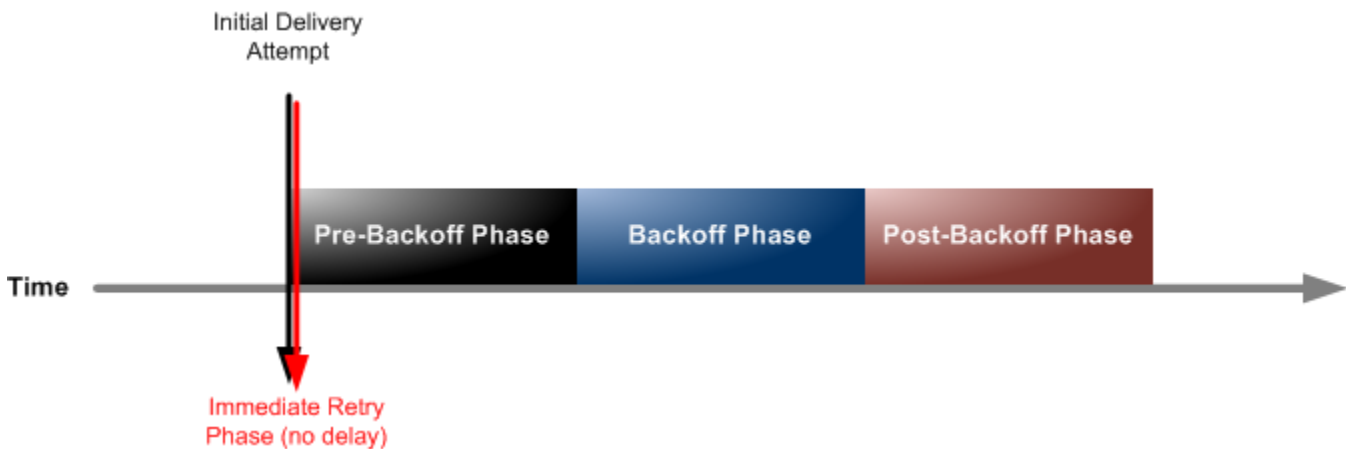
- 您無法變更 Amazon SNS 定義的傳遞政策，但 HTTP/S 除外。只有 HTTP/S 支援自訂政策。請參閱 [建立 HTTP/S 傳遞政策](#)。
- Amazon SNS 會將抖動套用到傳遞重試。如需詳細資訊，請參閱 AWS 架構部落格中的[指數退避和抖動](#)貼文。
- HTTP/S 端點的總政策重試時間不能大於 3,600 秒。這是硬性限制，無法再增加。

端點類型	傳遞通訊協定	立即重試 (無延遲) 階段	預先輪詢階段	輪詢階段	事後輪詢階段	嘗試總次數
AWS 受管端點	Amazon Data Firehose1	3 次，毫無延遲	2 次，相隔 1 秒	10 次，含指數退避，從 1 秒到 20 秒	100,000 次，相隔 20 秒	100,015 次，在 23 天內
	AWS Lambda					
	Amazon SQS					
客戶管理的端點	SMTP	0 次，毫無延遲	2 次，相隔 10 秒	10 次，含指數退避，從 10 秒到 600 秒 (10 分鐘)	38 次，相隔 600 秒 (10 分鐘)	50 次嘗試，在 6 小時內
	SMS					
	行動推播					

1 對於 Firehose 通訊協定的限流錯誤，Amazon SNS 使用與客戶受管端點相同的交付政策。

傳遞政策階段

下圖顯示傳遞政策的各個階段。



每個傳遞政策都是由四個階段組成。

1. 立即重試階段（無延遲） – 此階段會在初次交付嘗試後立即發生。在此階段各次重試之間無延遲。
2. 預退避階段 – 此階段遵循立即重試階段。Amazon SNS 會先使用此階段來嘗試一組重試動作，然後再套用輪詢函數。此階段會指定重試次數和各次重試之間的延遲量。
3. 退避階段 – 此階段使用重試退避函數控制重試之間的延遲。此階段會設定最小延遲、最大延遲，以及重試輪詢函數，以定義延遲多快從最小增加到最大延遲。輪詢函數可以是算術、指數、幾何或線性類型。
4. 後退避階段 – 此階段遵循後退階段。它會指定重試次數和各次重試之間的延遲量。此為最後一個階段。

建立 HTTP/S 傳遞政策

您可以使用四個階段的傳遞政策來定義 Amazon SNS 如何重試將訊息傳遞至 HTTP/S 端點：無延遲、預退避、退避和後退避。此政策可讓您覆寫預設重試設定，並自訂以符合 HTTP 伺服器的容量。

您可以在主題或訂閱層級將 HTTP/S 交付政策定義為 JSON 物件：

- 主題層級政策 – 適用於連結至主題的所有 HTTP/S 訂閱。使用 [CreateTopic](#) 或 [SetTopicAttributes](#) API 動作來設定此政策。
- 訂閱層級政策 – 僅適用於特定訂閱。使用 [Subscribe](#) 或 [SetSubscriptionAttributes](#) API 動作來設定此政策。

或者，您也可以可以在 AWS CloudFormation 範本中使用 [AWS::SNS::Subscription](#) 資源。

您應該根據 HTTP/S 伺服器的容量自訂交付政策：

- 所有訂閱的單一伺服器 – 如果主題中的所有 HTTP/S 訂閱都使用相同的伺服器，請將交付政策設定為主題屬性，以確保所有訂閱的一致性。
- 訂閱的不同伺服器 – 如果訂閱以不同的伺服器為目標，請為每個訂閱建立唯一的交付政策，根據特定伺服器的容量量身打造。

您也可以可以在請求政策中設定 Content-Type 標頭，以指定通知的媒體類型。根據預設，Amazon SNS 會將所有通知傳送至內容類型設為的 HTTP/S 端點 `text/plain; charset=UTF-8`。不過，您可以使用請求政策中的 [headerContentType](#) 欄位覆寫此預設值。

下列 JSON 物件定義交付政策，其重試分為四個階段：

1. 無延遲階段 – 立即重試 3 次。
2. 預退避階段 – 以 1 秒間隔重試 2 次。
3. 退避階段 – 重試 10 次，指數延遲範圍為 1 到 60 秒。
4. 後退避階段 – 以固定的 60 秒間隔重試 35 次。

Amazon SNS 在捨棄訊息之前，總共會嘗試傳遞訊息 50 次。若要保留所有重試後無法傳遞的訊息，請設定訂閱將無法傳遞的訊息移至無效字母佇列 (DLQ)。如需詳細資訊，請參閱 [Amazon SNS 無效字母佇列](#)。

Amazon SNS 會將所有 5XX 錯誤和 429（傳送的請求太多）錯誤視為可重試。這些錯誤受交付政策約束。所有其他錯誤都視為永久失敗，不會嘗試重試。

Note

此交付政策使用 `maxReceivesPerSecond` 屬性，將交付流量調節為每個訂閱平均每秒 10 則訊息。雖然此機制有助於防止 HTTP/S 端點因高流量而不堪負荷，但其設計目的是維持平均交付速率，並且不會強制執行嚴格的上限。偶爾可能會發生超過指定限制的交付流量峰值，特別是當您的發佈速率明顯高於限流限制時。

當發佈（傳入）流量超過交付（傳出）速率時，可能會導致訊息積壓和更高的交付延遲。若要避免此類問題，請確定 `maxReceivesPerSecond` 值符合您 HTTP/S 伺服器的容量和工作負載需求。

下列交付政策範例会覆寫 HTTP/S 通知的預設內容類型給 application/json。

```
{
  "healthyRetryPolicy": {
    "minDelayTarget": 1,
    "maxDelayTarget": 60,
    "numRetries": 50,
    "numNoDelayRetries": 3,
    "numMinDelayRetries": 2,
    "numMaxDelayRetries": 35,
    "backoffFunction": "exponential"
  },
  "throttlePolicy": {
    "maxReceivesPerSecond": 10
  },
  "requestPolicy": {
    "headerContentType": "application/json"
  }
}
```

交付政策由重試政策、調節政策和請求政策組成。交付政策中總共有 9 個屬性。

政策	描述	限制條件
minDelayTarget	重試的最小延遲。 單位：秒	1 到最大延遲 預設：20
maxDelayTarget	重試的最大延遲。 單位：秒	最小延遲到 3,600 預設：20
numRetries	重試總數，包括立即、預先輪詢、輪詢和事後輪詢重試。	0 到 100 預設：3
numNoDelayRetries	要立即完成的重試次數，且各次重試之間毫無延遲。	0 或以上 預設：0

政策	描述	限制條件
numMinDelayRetries	預先輪詢階段中的重試次數，且各次重試之間有指定的最小延遲。	0 或以上 預設：0
numMaxDelayRetries	事後輪詢階段中的重試次數，且各次重試之間有最大延遲。	0 或以上 預設：0
backoffFunction	重試之間的輪詢模式。	下列四個選項之一： • 算術 • 指數 • 幾何 • 線性 預設值：線性
maxReceivesPerSecond	每個訂閱每秒的平均訊息傳遞次數上限。	1 或以上 預設：無限流（傳遞速率無限制）

政策	描述	限制條件
headerContentType	傳送至 HTTP/S 端點之通知的內容類型。	如果未定義請求政策，則內容類型預設為 text/plain; charset=UTF-8 。 當訂閱停用原始訊息傳遞時 (預設)，或在主題層級定義傳遞政策時，支援的標頭內容類型為 application/json 和 text/plain 。 啟用訂閱的原始訊息傳遞時，支援下列內容類型： • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • application/x-www-form-urlencoded • application/xhtml+xml • application/xml

Amazon SNS 會使用下列公式計算輪詢階段中的重試次數：

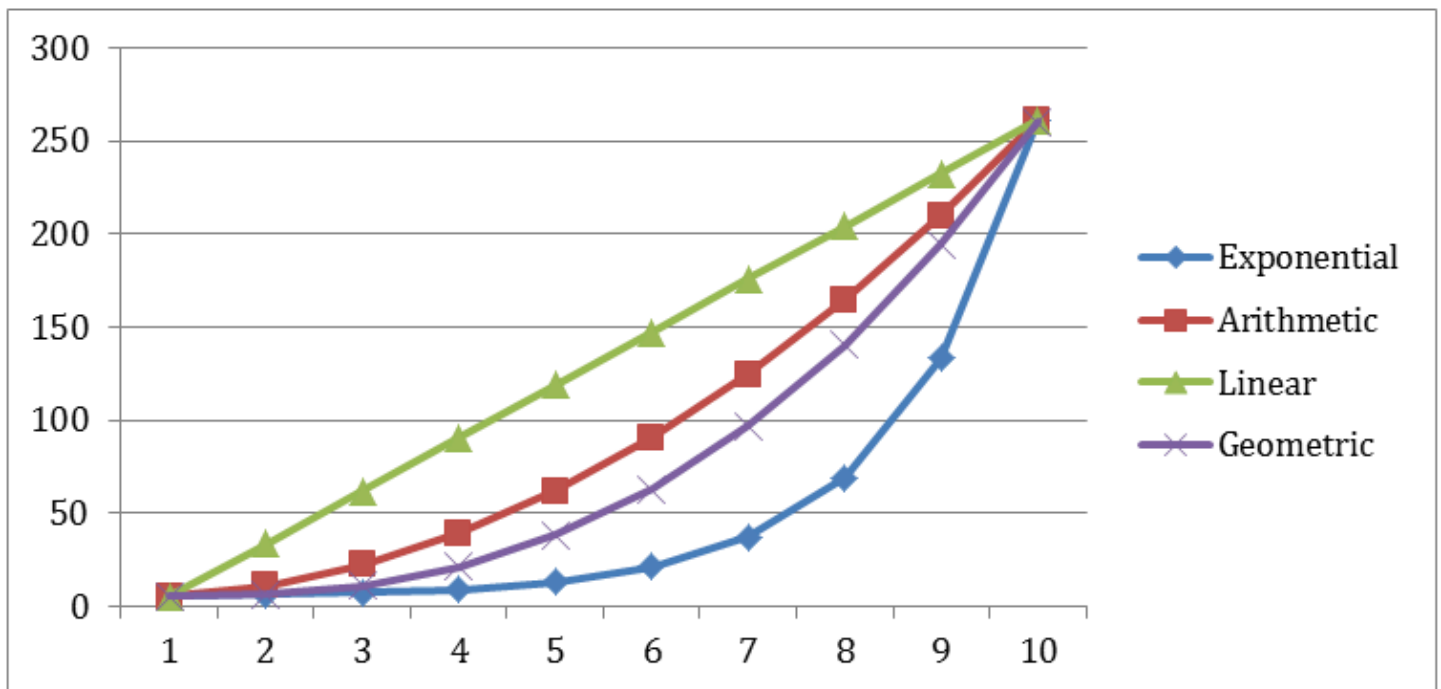
```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

您可以使用三個參數來控制退避階段期間的重試頻率：

- **minDelayTarget** – 設定退避階段中第一次重試嘗試的延遲。
- **maxDelayTarget** – 設定退避階段中最後一次重試嘗試的延遲。
- **backoffFunction** – 決定 Amazon SNS 用來計算第一次和最後一次重試之間所有重試嘗試的延遲演算法。您可以從四個可用的重試退避函數中進行選擇。

下圖說明不同的重試退避函數如何影響退避階段期間重試之間的延遲。用於此範例的交付政策包括下列設定：10 次重試、最短延遲 5 秒，以及最長延遲 260 秒。

- 垂直軸會顯示每次重試嘗試的延遲（以秒為單位）。
- 水平軸代表重試序列，範圍從第一次到第十次嘗試。



Amazon SNS 無效字母佇列

無效字母佇列是一種 Amazon SQS 佇列，Amazon SNS 將它作為無法成功傳遞給訂閱者訊息的目標。由於用戶端錯誤或伺服器錯誤而無法傳遞的訊息，會保留在無效字母佇列，以供進一步分析或重新處理。如需詳細資訊，請參閱 [設定訂閱的 Amazon SNS 無效字母佇列](#) 和 [Amazon SNS 訊息傳遞重試](#)。

Note

- Amazon SNS 訂閱和 Amazon SQS 佇列必須位於相同的 AWS 帳戶和區域下。

- 對於 [FIFO 主題](#)，您可以使用 Amazon SQS 佇列做為 Amazon SNS 訂閱的無效字母佇列。FIFO 主題訂閱使用 FIFO 佇列，而標準主題訂閱則使用標準佇列。
- 若要使用加密的 Amazon SQS 佇列做為無效字母佇列，您必須使用具有金鑰政策的自訂 KMS，以授予 Amazon SNS 服務主體對 AWS KMS API 動作的存取權。如需詳細資訊，請參閱本指南中的 [使用伺服器端加密保護 Amazon SNS 資料](#) 及 Amazon Simple Queue Service 開發人員指南中的 [保護 Amazon SQS 資料與使用伺服器端加密 \(SSE\) 和 AWS KMS](#)。

訊息傳遞為何失敗？

一般而言，當 Amazon SNS 因用戶端或伺服器端錯誤而無法存取訂閱的端點時，訊息傳遞就會失敗。當 Amazon SNS 收到用戶端錯誤，或繼續收到伺服器端錯誤 (因為訊息超出對應重試政策所指定的重試次數)，除非已將無效字母佇列附加至訂閱，否則 Amazon SNS 就會捨棄訊息。失敗傳遞不會改變您的訂閱狀態。如需詳細資訊，請參閱[Amazon SNS 訊息傳遞重試](#)。

用戶端錯誤

當 Amazon SNS 有過時的訂閱中繼資料時，可能就會發生用戶端錯誤。當擁有者刪除端點 (例如對 Amazon SNS 主題訂閱的 Lambda 函數) 或擁有者變更附加至訂閱端點的政策而阻止 Amazon SNS 將訊息傳遞至端點，則經常會發生這些錯誤。Amazon SNS 不會重試因用戶端錯誤而失敗的訊息傳遞。

伺服器端錯誤

當負責訂閱端點的系統無法使用或傳回例外狀況，表示無法處理 Amazon SNS 的有效請求，可能就會發生伺服器端錯誤。發生伺服器端錯誤時，Amazon SNS 會使用線性或指數退避函數來重試失敗的傳遞。對於由 Amazon SQS 或支援的 AWS 受管端點引起的伺服器端錯誤 AWS Lambda，Amazon SNS 會在 23 天內重試交付最多 100,015 次。

客戶管理的端點 (例如 HTTP、SMTP、SMS 或行動推播) 可能也會導致伺服器端錯誤。Amazon SNS 也會重試傳遞到這些類型的端點。雖然 HTTP 端點支援客戶定義的重試政策，但針對 SMTP、SMS 和行動推播端點，Amazon SNS 會將內部傳遞重設政策設為 6 小時內 50 次。

無效字母佇列的運作方式

無效字母佇列會附加至 Amazon SNS 訂閱 (而不是主題)，因為訊息傳遞是在訂閱層級發生。這可讓您更輕鬆地識別每個訊息的原始目標端點。

與 Amazon SNS 訂閱相關聯的無效字母佇列是一般的 Amazon SQS 佇列。如需訊息保留期間的詳細資訊，請參閱 Amazon Simple Queue Service 開發人員指南中的[訊息相關配額](#)。您可以使用 Amazon SQS [SetQueueAttributes](#) API 動作來變更訊息保留期間。若要讓應用程式更具彈性，建議您將無效字母佇列的最大保留期間設為 14 天。

訊息如何移至無效字母佇列？

訊息是透過再驅動政策移至無效字母佇列。再驅動政策是參照無效字母佇列之 ARN 的 JSON 物件。deadLetterTargetArn 屬性會指定 ARN。ARN 必須指向與您的 Amazon SNS 訂閱位於相同和區域中的 Amazon SQS 佇列。AWS 帳戶 Amazon SNS 如需詳細資訊，請參閱[設定訂閱的 Amazon SNS 無效字母佇列](#)。

以下 JSON 物件是附加至 SNS 訂閱的再驅動政策範例。

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

如何將訊息從無效字母佇列中移出？

您可以透過下列兩種方式將訊息從無效字母佇列中移出：

- 避免編寫 Amazon SQS 消費者邏輯 - 將無效字母佇列設為 Lambda 函數的事件來源，以耗盡無效字母佇列。
- 寫入 Amazon SQS 取用者邏輯 – 使用 Amazon SQS API、AWS SDK 或 AWS CLI 撰寫自訂取用者邏輯，以輪詢、處理和刪除無效字母佇列中的訊息。

如何監控和記錄無效字母佇列？

您可以使用 Amazon CloudWatch 指標來監控與您 Amazon SNS 訂閱相關聯的無效字母佇列。所有 Amazon SQS 佇列都會每隔一分鐘發出 CloudWatch 指標。如需詳細資訊，請參閱《[Amazon Simple Queue Service 開發人員指南](#)》中的 [Amazon SQS 的可用 CloudWatch 指標](#)。所有包含無效字母佇列的 Amazon SNS 訂閱也會發出 CloudWatch 指標 如需詳細資訊，請參閱[使用 Amazon CloudWatch 監控 Amazon SNS 主題](#)。

若要獲得無效字母佇列中的活動通知，您可以使用 CloudWatch 指標和警示。

為NumberOfMessagesSent指標設定警示並不適用，因為此指標不會擷取因處理嘗試失敗而傳送至 DLQ 的訊息。反之，請使用 ApproximateNumberOfMessagesVisible 指標來擷取 DLQ 中目前可用的所有訊息，包括因處理失敗而移動的訊息。

CloudWatch 警示設定範例

1. 建立指標的 [CloudWatch 警示](#)。 **ApproximateNumberOfMessagesVisible**
2. 將警示閾值設定為 1 (或根據您的期望和 DLQ 流量的其他適當值)。
3. 指定警示關閉時要通知的 Amazon SNS 主題。此 Amazon SNS 主題可將警示通知傳遞給任何端點類型 (例如電子郵件地址、電話號碼或行動傳呼應用程式)。

您可以使用 CloudWatch Logs 調查導致任何 Amazon SNS 傳遞失敗的例外狀況以及傳送至無效字母佇列的訊息。Amazon SNS 可以在 CloudWatch 中記錄成功和失敗的交付。如需詳細資訊，請參閱 [Amazon SNS 行動應用程式屬性](#)。

設定訂閱的 Amazon SNS 無效字母佇列

無效字母佇列是一種 Amazon SQS 佇列，Amazon SNS 將它作為無法成功傳遞給訂閱者訊息的目標。由於用戶端錯誤或伺服器錯誤而無法傳遞的訊息，會保留在無效字母佇列，以供進一步分析或重新處理。如需詳細資訊，請參閱 [Amazon SNS 無效字母佇列](#) 和 [Amazon SNS 訊息傳遞重試](#)。

此頁面說明如何使用 AWS Management Console、AWS 開發套件 AWS CLI、和 AWS CloudFormation 來設定 Amazon SNS 訂閱的無效字母佇列。

Note

對於 [FIFO 主題](#)，您可以使用 Amazon SQS 佇列做為 Amazon SNS 訂閱的無效字母佇列。FIFO 主題訂閱使用 FIFO 佇列，而標準主題訂閱則使用標準佇列。

先決條件

設定無效字母佇列之前，請完成以下先決條件：

1. [建立 Amazon SNS 主題](#)，命名為 MyTopic。
2. [建立 Amazon SQS 佇列](#)，命名為 MyEndpoint，用來做為 Amazon SNS 訂閱的端點。
3. (Skip for AWS CloudFormation) [將佇列訂閱至主題](#)。
4. [建立另一個 Amazon SQS 佇列](#)，命名為 MyDeadLetterQueue，用作 Amazon SNS 訂閱的無效字母佇列。
5. 若要授予 Amazon SNS 委託人對 Amazon SQS API 動作的存取權，請設定 MyDeadLetterQueue 的下列佇列政策。

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

使用 設定 Amazon SNS 訂閱的無效字母佇列 AWS Management Console

開始本教學課程之前，請先完成下列[先決條件](#)。

1. 請登入 [Amazon SQS 主控台](#)。
2. [建立 Amazon SQS 佇列](#)或使用現有佇列，並記下佇列 Details (詳細資訊) 索引標籤上顯示的佇列 ARN，例如：

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

3. 登入 [Amazon SNS 主控台](#)。
4. 在導覽面板上，選擇 Subscriptions (訂閱)。
5. 在 Subscriptions (訂閱) 頁面上，選取現有訂閱，然後選擇 Edit (編輯)。
6. 在 Edit **1234a567-bc89-012d-3e45-6fg7h890123i**(編輯 1234a567-bc89-012d-3e45-6fg7h890123i) 頁面上，展開 Redrive policy (dead-letter queue) (再驅動政策 (無效字母佇列)) 區段，然後執行下列動作：
 - a. 選擇 Enable (啟用)。
 - b. 指定 Amazon SQS 佇列的 ARN。

7. 選擇 Save changes (儲存變更)。

您的訂閱已設為使用無效字母佇列。

使用 AWS SDK 設定 Amazon SNS 訂閱的無效字母佇列

執行此範例之前，請確定您已完成[先決條件](#)。

若要使用 AWS 開發套件，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS SDK和工具參考指南中的[共享的配置和認證文件](#)。

下列程式碼範例示範如何使用 SetSubscriptionAttributesRedrivePolicy。

Java

適用於 Java 1.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#)中設定和執行。

```
// 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);
```

使用 設定 Amazon SNS 訂閱的無效字母佇列 AWS CLI

開始本教學課程之前，請先完成下列[先決條件](#)。

1. 安裝及設定 AWS CLI。如需詳細資訊，請參閱 [AWS Command Line Interface 使用者指南](#)。
2. 使用下列 命令。

```
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\"}"
```

使用 設定 Amazon SNS 訂閱的無效字母佇列 AWS CloudFormation

開始本教學課程之前，請先完成下列[先決條件](#)。

1. 將下列 JSON 程式碼複製到名為 MyDeadLetterQueue.json 的檔案。

```
{  
  "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. 登入 [AWS CloudFormation 主控台](#)。

3. 在 Select Template (選擇範本) 頁面中，選擇 Upload a template to Amazon S3 (上傳範本至 Amazon S3)、選擇您的 MyDeadLetterQueue.json 檔案，再選擇 Next (下一步)。
4. 在 Specify Details (指定詳細資訊) 頁面上，為 Stack Name (堆疊名稱) 輸入 MyDeadLetterQueue，然後選擇 Next (下一步)。
5. 在 Options (選項) 頁面上，選擇 Next (下一步)。
6. 在 Review (檢閱) 頁面上，選擇 Create (建立)。

AWS CloudFormation 開始建立MyDeadLetterQueue堆疊，並顯示 CREATE_IN_PROGRESS 狀態。程序完成時，AWS CloudFormation 會顯示 CREATE_COMPLETE 狀態。

Amazon SNS 訊息封存、重播和分析

Amazon SNS 標準主題支援透過 Amazon Data Firehose 進行訊息封存。您可以向 Firehose 交付串流發出通知，這可讓您將通知傳送至 Firehose 支援的儲存和分析目的地，包括 Amazon Simple Storage Service (Amazon S3)、Amazon Redshift 等。

Amazon SNS FIFO 主題支援就地、無程式碼的訊息封存，可讓主題擁有者將訊息儲存 (或封存) 發佈至主題的訊息長達 365 天。對於具有作用中 ArchivePolicy 的主題，訂閱用戶可以建立 ReplayPolicy 將封存的訊息擷取 (或重播) 回訂閱的端點。若要進一步了解此功能，請參閱 [FIFO 主題的 Amazon SNS 訊息封存和重播](#)。

功能	標準主題	FIFO 主題
訊息封存	散發到 Firehose 交付串流	FIFO 主題擁有者的 Amazon SNS 訊息封存
訊息重播	重播並不是標準主題的內建功能。許多客戶會根據自己的訊息封存來自行建立。	FIFO 主題訂閱者的 Amazon SNS 訊息重播

Amazon SNS 中的資源管理和最佳化

本主題提供如何透過確保最佳效能、降低不必要的成本和維護組織良好的資源，充分利用 Amazon SNS 完整潛力的指引。

主題

- [Amazon SNS 主題標記](#)

Amazon SNS 主題標記

Amazon SNS 支援 Amazon SNS 主題的標記功能。這可協助您追蹤和管理與主題相關的成本、在 AWS Identity and Access Management (IAM) 政策中提供增強的安全性，並可讓您輕鬆地搜尋或篩選數千個主題。標記可讓您使用 AWS 資源群組來管理 Amazon SNS 主題。如需 Resource Groups 的詳細資訊，請參閱《[AWS Resource Groups 使用者指南](#)》。

為分配成本加上標籤

若要整理並識別 Amazon SNS 主題以分配成本，您可新增標籤來識別主題的用途。擁有許多主題時特別實用。您可以使用成本分配標籤來組織 AWS 帳單，以反映您自己的成本結構。若要這樣做，請註冊以取得 AWS 您的帳戶帳單，以包含標籤索引鍵和值。如需詳細資訊，請參閱《[AWS 帳單與成本管理使用者指南](#)》中的「[設定每月成本分配報告](#)」。

例如，您可以新增代表成本中心和 Amazon SNS 主題之用途的標籤，如下所示：

資源	金鑰	值
主題 1	成本中心	43289
	應用程式	訂單處理
主題 2	成本中心	43289
	應用程式	付款處理
主題 3	成本中心	76585
	應用程式	封存

這種標記機制可讓您將同一個成本中心內執行相關任務的兩個主題歸為同一組，並且使用不同的成本分配標籤來標記不相關的活動。

為存取控制加上標籤

AWS Identity and Access Management 支援根據標籤控制對資源的存取。標記資源後，請在 IAM 政策的條件元素中提供有關資源標籤的資訊來管理標籤型存取。如需有關如何使用 [Amazon SNS 主控台](#) 或 [AWS SDK](#) 來標記資源的資訊，請參閱[設定標籤](#)。

您可以限制 IAM 身分的存取。例如，您可以限制 Publish 和 PublishBatch 對所有包含索引鍵 environment 和 值 production 的 Amazon SNS 主題的存取，同時卻也可以允許對所有其他 Amazon SNS 主題的存取。在下方範例中，政策會限制發布訊息到標記為 production 之主題的能力，同時卻也會允許將消息發布到標記為 development 之主題。如需詳細資訊，請參閱《IAM 使用者指南》中的「[使用標籤控制存取](#)」。

Note

設定 IAM 許可以供 Publish 設定 Publish 和 PublishBatch 這兩者的許可。

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

為資源搜尋和過濾加上標籤

AWS 帳戶可以有數萬個 Amazon SNS 主題（如需詳細資訊，請參閱 [Amazon SNS 配額](#)）。標記主題後，您就可以簡化搜尋或篩選主題的過程。

例如，您可能會有數以百計個與生產環境關聯的主題。您可以查詢具有指定標籤的所有主題，而不必手動搜尋這些主題：

```
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());  
    }  
}
```

```
}
```

設定 Amazon SNS 主題標籤

本主題說明如何使用 AWS Management Console、AWS SDK 或 設定 [Amazon SNS 主題](#) 的標籤 AWS CLI。

Important

請勿在標籤中加入個人身分識別資訊 (PII) 或其他機密或敏感資訊。其他 Amazon Web Services 可以存取標籤，包括帳單。標籤不適用於私人或敏感資料。

使用 列出、新增和移除 Amazon SNS 主題的標籤 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選擇主題，然後選擇 Delete (刪除)。
4. 展開 Tags (標籤) 區段。

隨即列出已新增到主題的標籤。

5. 修改主題標籤：
 - 若要新增標籤，請選擇 Add tag (新增標籤)，然後輸入 Key (索引鍵) 和 Value (值) (選用)。
 - 若要移除標籤，請選擇鍵/值對旁邊的 Remove tag (移除標籤)。
6. 選擇 Save changes (儲存變更)。

使用 AWS SDK 新增標籤到主題

若要使用 AWS 開發套件，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS SDK 和工具參考指南中的 [共享的配置和認證文件](#)。

下列程式碼範例示範如何使用 TagResource。

CLI

AWS CLI

將標籤新增至主題

下列 `tag-resource` 範例會將中繼資料標籤新增到指定的 Amazon SNS 主題。

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

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [TagResource](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);  
        }  
    }  
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [TagResource](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun addTopicTags(topicArn: String) {  
    val tag =  
        Tag {  
            key = "Team"  
            value = "Development"  
        }  
  
    val tag2 =  
        Tag {  
            key = "Environment"  
            value = "Gamma"  
        }  
  
    val tagList = mutableListOf<Tag>()  
    tagList.add(tag)  
    tagList.add(tag2)  
  
    val request =  
        TagResourceRequest {  
            resourceArn = topicArn  
            tags = tagList  
        }  
}
```

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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [TagResource](#)。

以 Amazon SNS API 動作管理標籤

若要使用 Amazon SNS API 管理標籤，請使用下列 API 動作：

- [ListTagsForResource](#)
- [TagResource](#)
- [UntagResource](#)

支援 ABAC 的 API 動作

以下是支援屬性型存取控制 (ABAC) 的 API 動作列表。如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的[什麼是 ABAC AWS？](#)。

- [AddPermission](#)
- [ConfirmSubscription](#)
- [DeleteTopic](#)
- [GetDataProtectionPolicy](#)
- [GetSubscriptionAttributes](#)
- [GetTopicAttributes](#)
- [ListSubscriptionsByTopic](#)
- [ListTagsForResource](#)
- [Publish](#)
- [PublishBatch](#)
- [PutDataProtectionPolicy](#)
- [RemovePermission](#)

- [SetSubscriptionAttributes](#)
- [SetTopicAttributes](#)
- [Subscribe](#)
- [TagResource](#)
- [Unsubscribe](#)
- [UntagResource](#)

Amazon SNS 事件來源和目的地

Amazon SNS 透過路由事件驅動的通知來連接 AWS 服務 和外部系統。Amazon SNS 接收來自各種事件 AWS 服務，例如資料管道更新、Amazon EC2 擴展動作或安全提醒，並將這些事件發佈至 Amazon SNS 主題。這些主題接著會傳送通知至指定的目的地。

Amazon SNS 支援兩種主要的目的地類型：[Application-to-Application\(A2A\)](#) [Application-to-Person\(A2P\)](#)。在 A2A 訊息中，Amazon SNS 可以將事件傳送至 Lambda 以觸發自訂商業邏輯、將訊息排入佇列的 Amazon SQS，以及將資料串流至儲存和分析服務的 Amazon Kinesis Data Firehose。對於 A2P 訊息，Amazon SNS 可以透過簡訊、電子郵件和推送通知將通知傳送到行動裝置，確保使用者或團隊及時收到提醒。

Amazon SNS 做為中央中樞，可將通知路由到正確的位置，協助您更有效地自動化和管理工作 AWS 基礎設施。此設定允許在 服務之間無縫整合，並與使用者和系統進行可靠的通訊。

Amazon SNS 事件來源

Amazon SNS 與 AWS 服務 各種類別的廣泛整合，允許這些服務將事件發佈至 Amazon SNS 主題。此整合提供關鍵事件的即時通知，例如基礎設施、應用程式效能和成本管理的變更。

Note

Amazon SNS 在 2020 年 10 月推出 [FIFO 主題](#)。目前，大多數 AWS 服務僅支援將事件傳送至標準主題。

分析服務

下表說明 Amazon SNS 如何與 Athena AWS Data Pipeline、和 Amazon Redshift 等 AWS 分析服務整合，以提供關鍵事件的即時通知，包括控制限制違規、管道狀態更新和資料倉儲活動。

您可以利用這些整合來自動化回應，並維持資料操作的有效監督。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Athena - 可讓您使用標準 SQL 來分析 Amazon S3 中的資料。	超出管制限制時接收通知。如需詳細資訊，請參閱 Amazon Athena 使用者指南中的 設定資料用量控制限制 。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Data Pipeline - 有助於自動化資料的移動和轉換。	接收有關管道元件狀態的通知。如需詳細資訊，請參閱 AWS Data Pipeline 開發人員指南 中的 SnsAlarm 。
Amazon Redshift - 管理設定、操作和擴展資料倉儲服務的所有工作。	接收 Amazon Redshift 事件的通知。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的 Amazon Redshift 事件通知 。

應用程式整合服務

下表說明 Amazon SNS 如何與 EventBridge 等應用程式整合服務整合 AWS Step Functions，並為業務關鍵型應用程式啟用即時資料路由和通知。

您可以利用這些整合來接收來自 EventBridge 事件的提醒，並使用 Step Functions 協調工作流程，增強應用程式的自動化和回應能力。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon EventBridge – 從您自己的應用程式、software-as-a-service(SaaS) 應用程式 AWS 和服務提供即時資料的串流，並將該資料路由到目標，包括 Amazon SNS。EventBridge 之前被稱為 CloudWatch Events。	接收事件 EventBridge 事件的通知。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南 中的 Amazon EventBridge 目標 。
AWS Step Functions – 可讓您結合 AWS Lambda 函數和其他 AWS 服務來建置業務關鍵型應用程式。	接收 Step Functions 事件的通知。如需詳細資訊，請參閱 AWS Step Functions 開發人員指南 中的 使用 Step Functions 叫用 Amazon SNS 。

帳單與成本管理服務

下表說明 如何與 Amazon SNS AWS 帳單與成本管理 整合，以提供預算、價格變更和成本異常的通知。

您可以利用此整合來設定 Amazon SNS 主題，以接收有關 AWS 支出的即時提醒，協助您監控成本並有效率地回應意外費用。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS 帳單與成本管理 - 提供可協助您監控成本並支付帳單的功能。	接收預算通知、價格變更通知和異常警示。如需詳細資訊，請參閱 AWS Billing 使用者指南 中的下列主題： • 建立預算通知的 Amazon SNS 主題 • 設定通知 • 使用 偵測不尋常的支出 AWS Cost Anomaly Detection

商業應用程式服務

下表說明 Amazon Chime 如何與 Amazon SNS 整合，以傳送重要會議事件的通知，讓您隨時掌握通訊和排程的相關資訊。

您可以利用此整合，利用 Amazon Chime SDK 事件通知來增強組織內外的協同合作工具。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Chime - 可讓您在您的組織內部和外部開會、聊天和撥打商務電話。	接收重要的會議事件通知。如需詳細資訊，請參閱 Amazon Chime 開發人員指南中的 Amazon Chime 開發套件事件通知 。

運算服務

下表說明 Amazon SNS 如何與各種 AWS 運算服務整合，讓您接收 Auto Scaling 動作、EC2 Image Builder 完成、Elastic Beanstalk 環境變更、Lambda Lightsail 函數輸出和指標閾值等重要事件的通知。

您可以利用這些整合，透過隨時掌握的重要更新和動作，有效率地管理您的應用程式和資源 AWS 服務。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon EC2 Auto Scaling - 協助您擁有正確數量的 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體可用來處理應用程式負載。	當 Auto Scaling 在 Auto Scaling 群組中啟動或終止 Amazon EC2 執行個體時收到通知。有關詳細資訊，請參閱 Amazon EC2 Auto Scaling 使用者指南中的 取得 Auto Scaling 群組擴展時的 Amazon SNS 通知 。
EC2 Image Builder – 協助自動化自訂、安全且最新伺服器映像的建立、管理和部署，這些映像會預先安裝並預先設定為軟體設定以符合特定 IT 標準。	組建完成時接收通知。如需詳細資訊，請參閱 AWS 運算部落格上的 追蹤 EC2 Image Builder 管道中的最新伺服器映像 。
AWS Elastic Beanstalk - 處理容量佈建、負載平衡、應用程式擴展，以及提供應用程式運作狀態監控等詳細資訊。	接收影響您的應用程式的重要事件通知。如需詳細資訊，請參閱 AWS Elastic Beanstalk 開發人員指南中的 使用 Amazon SNS 的 Elastic Beanstalk 環境通知 。
AWS Lambda - 可讓您直接執行程式碼，無需佈建或管理伺服器。	將 SNS 主題設定為 Lambda 無效字母佇列或 Lambda 目的地，以接收函數輸出資料。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的 非同步叫用 。
Amazon Lightsail – 協助開發人員開始使用 AWS 建置網站或 Web 應用程式。	在其中一個執行個體、資料庫或負載平衡器的指標超過指定的閾值時接收通知。如需詳細資訊，請參閱 Amazon Lightsail 開發人員指南中的 在 Amazon Lightsail 中新增通知聯絡人 。

容器服務

下表說明 Amazon SNS 如何與 Amazon EKS Distro 和 Amazon ECS 等 AWS 容器服務整合，可讓您追蹤 Amazon EKS 叢集的更新和安全修補程式，並接收新 ECS 最佳化 AMI 版本的通知。

您可以利用這些整合來保持容器部署的安全性和效率，方法是隨時掌握重要的更新和變更。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon EKS 發行版 - 可讓您在部署應用程式的任何位置建立可靠且安全的叢集。	追蹤使用 Amazon EKS 發行版建立的叢集的更新和安全修補程式。如需詳細資訊，請參閱 推出 Amazon EKS 發行版 - Amazon EKS 使用的開放原始碼 Kubernetes 發行版 。
Amazon Elastic Container Service (Amazon ECS) - 可讓您在叢集上執行、停止和管理容器。	當有新的 Amazon ECS 優化 AMI 可用時接收通知。如需詳細資訊，請參閱 Amazon Elastic Container Service 開發人員指南中的 訂閱 Amazon ECS 最佳化 AMI 更新通知 。

客戶參與度服務

下表說明 Amazon SNS 如何透過與 Amazon Connect AWS End User Messaging SMS 和 Amazon Simple Email Service (SES) 整合來增強客戶參與度服務，讓您能夠接收提醒和驗證、設定雙向簡訊，以及監控電子郵件通知是否有退信、投訴和交付。

這些整合可協助您跨多個管道管理客戶通訊。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Connect - 可讓您設定全通道雲端聯絡中心，與您的客戶互動。	接收警示和驗證。如需詳細資訊，請參閱 《Amazon Connect 管理員指南》中的 AWS 使用 Amazon Connect 的功能 。Amazon Connect
AWS End User Messaging SMS – 透過傳送電子郵件、簡訊和語音訊息以及推播通知，協助您與客戶互動。	設定雙向簡訊，這可讓您接收客戶的訊息。如需詳細資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的 雙向簡訊 。
Amazon Simple Email Service (Amazon SES) - 為您提供符合經濟效益的方式來使用自有電子郵件地址和網域傳送和接收電子郵件。	接收退信、投訴與遞送訊息的通知。如需詳細資訊，請參閱 Amazon Simple Notification Service 開發人員指南中的 設定 Amazon SES 的 Amazon SNS 通知 。

資料庫服務

下表說明 Amazon SNS 如何與 AWS 資料庫服務整合，例如 AWS Database Migration Service (DMS)、Amazon DynamoDB、Amazon ElastiCache、Amazon Neptune、Amazon Redshift 和 Amazon Relational Database Service (RDS)，以傳送有關資料遷移、維護活動、快取更新和資料庫變更等重要事件的通知。

這些整合可提供關鍵操作事件的及時提醒，協助您更有效地監控和管理資料庫環境。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Database Migration Service – 將資料從內部部署資料庫遷移至 AWS 雲端。	在 AWS DMS 事件發生時接收通知，例如建立或刪除複寫執行個體時。如需詳細資訊，請參閱 AWS Database Migration Service 使用者指南中的 在 AWS Database Migration Service 中使用事件及通知 。
Amazon DynamoDB - 是一項完全受管的 NoSQL 資料庫服務，可提供快速且可預期的效能及無縫的可擴展性。	在發生維護事件時接收通知。如需詳細資訊，請參閱 Amazon DynamoDB 開發人員指南中的 自訂 DAX 叢集設定 。
Amazon ElastiCache - 此項服務提供高效能、可調整大小和符合成本效益的記憶體內快取，同時消除了部署與管理分散式快取環境的相關複雜性。	在發生重大事件時接收通知。如需詳細資訊，請參閱《 Amazon ElastiCache (Memcached) 使用者指南 》中的 事件通知 和 Amazon SNS 。Amazon ElastiCache
Amazon Neptune - 可讓您建立與執行使用高度連線資料集的應用程式。	在發生 Neptune 事件時接收通知。如需詳細資訊，請參閱 Neptune 使用者指南中的 使用 Neptune 事件通知 。
Amazon Redshift - 管理設定、操作和擴展資料倉儲服務的所有工作。	接收 Amazon Redshift 事件的通知。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的 Amazon Redshift 事件通知 。
Amazon Relational Database Service – 可讓您更輕鬆地在 AWS 雲端中設定、操作和擴展關聯式資料庫。	接收 Amazon RDS 事件的通知。如需詳細資訊，請參閱 Amazon RDS 使用者指南中的 使用 Amazon RDS 事件通知 。

開發人員工具服務

下表說明 Amazon SNS 如何與 AWS 開發人員工具服務整合 AWS CodeBuild，例如 AWS CodeCommit AWS CodeDeploy Amazon CodeGuru，並為建置狀態變更、儲存庫更新、部署進度、效能異常和管道動作等重要事件 AWS CodePipeline 提供通知。

這些整合可透過接收有關重要事件的及時提醒，協助您有效率地監控和管理軟體開發工作流程。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS CodeBuild - 可編譯來源碼、執行單位測試，並產生可立即部署的成品。	當組建成功、失敗或從一個組建階段移至另一個組建階段時，接收通知。如需詳細資訊，請參閱 AWS CodeBuild 使用者指南中的 建置 CodeBuild 通知範例 。
AWS CodeCommit - 提供版本控制，以便在雲端中私有儲存和管理資產。	接收有關 CodeCommit 存放庫事件的通知。如需詳細資訊，請參閱 AWS CodeCommit 《使用者指南》中的 範例：建立 Amazon SNS 主題的 AWS CodeCommit 觸發條件 。
AWS CodeDeploy - 自動將應用程式部署至 Amazon EC2 執行個體、內部部署執行個體、無伺服器 Lambda 函數或 Amazon ECS 服務。	接收 CodeDeploy 或執行個體事件的通知。如需詳細資訊，請參閱 AWS CodeDeploy 使用者指南中的 建立 CodeDeploy 事件的觸發 。
Amazon CodeGuru - 從即時應用程式收集執行時間效能資料，並提供可協助您微調應用程式效能的建議。	發生異常時接收通知。如需詳細資訊，請參閱 Amazon CodeGuru 使用者指南中的 使用異常和建議報告 。
AWS CodePipeline - 自動執行持續發行軟體變更所需的步驟。	接收有關核准動作的通知。如需詳細資訊，請參閱 AWS CodePipeline 使用者指南中的 管理 CodePipeline 中的核准動作 。

前端網頁與行動服務

下表說明 Amazon SNS 如何與整合 AWS End User Messaging SMS，透過傳送電子郵件、簡訊、語音訊息和推送通知來增強客戶參與度，包括設定雙向簡訊以接收客戶訊息的能力。

此整合可讓您更有效地跨各種通訊管道與客戶互動。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS End User Messaging SMS – 透過傳送電子郵件、簡訊和語音訊息以及推播通知，協助您與客戶互動。	設定雙向簡訊，這可讓您接收客戶的訊息。如需詳細資訊，請參閱AWS End User Messaging SMS 《使用者指南》中的 雙向簡訊 。

遊戲開發服務

下表說明 Amazon SNS 如何與 Amazon GameLift 伺服器整合，以在工作階段型多玩家遊戲伺服器中提供配對和佇列事件的通知。

此整合可協助遊戲開發人員自動化和監控其遊戲伺服器的部署、操作和擴展，確保順暢的遊戲體驗。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon GameLift Servers – 提供在雲端託管以工作階段為基礎的多玩家遊戲伺服器的解決方案，包括用於部署、操作和擴展遊戲伺服器的全受管服務。	接收配對和佇列事件通知。如需詳細資訊，請參閱下列頁面： - 如需配對通知，請參閱《Amazon GameLift Servers FlexMatch 開發人員指南》中的設定 FlexMatch 事件通知。 GameLift FlexMatch - 如需佇列通知，請參閱《Amazon GameLift Servers 開發人員指南》中的設定遊戲工作階段放置的事件通知。

物聯網服務

下表說明 Amazon SNS 如何與 AWS IoT Core AWS IoT Device Defender、AWS IoT Events和 等 AWS IoT 服務整合 AWS IoT Greengrass，以提供 IoT 事件和警示的通知。

這些整合可讓您有效監控裝置行為、接收異常活動的提醒，以及透過即時更新和動作管理 IoT 裝置。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS IoT Core – 提供雲端服務，將您的 IoT 裝置連線至其他裝置 AWS 雲端 和服務。	接收 AWS IoT Core 事件通知。如需詳細資訊，請參閱 AWS IoT 開發人員指南中的 建立 Amazon SNS 規則 。
AWS IoT Device Defender - 許可您稽核裝置組態、監控連網裝置以偵測異常行為，並且防範安全風險。	當裝置違反行為時，接收警報。如需詳細資訊，請參閱《AWS IoT 開發人員指南》中的 如何使用 AWS IoT Device Defender 偵測 。
AWS IoT Events - 讓您監控設備和裝置機群的營運是否故障或變更，以及在這類事件發生時觸發動作。	接收 AWS IoT Events 事件通知。如需詳細資訊，請參閱 AWS IoT Events 開發人員指南中的 Amazon Simple Notification Service 。
AWS IoT Greengrass – 延伸至實體裝置 AWS，以便它們可以在本機對產生的資料採取行動，同時仍然使用雲端進行管理、分析和耐用儲存。	接收 AWS IoT Greengrass 事件通知。如需詳細資訊，請參閱 AWS IoT Greengrass Version 1 開發人員指南中的 SNS 連線 。

機器學習服務

下表說明 Amazon SNS 如何與 AWS 機器學習服務整合，例如 Amazon CodeGuru、Amazon DevOpsGuru、Amazon Lookout for Metrics、Amazon Rekognition 和 Amazon SageMaker AI，以提供異常、操作洞察和資料標記活動的通知。

這些整合可讓您監控應用程式效能、接收資料異常警示，以及透過即時更新簡化機器學習模型的部署。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon CodeGuru - 從即時應用程式收集執行時間效能資料，並提供可協助您微調應用程式效能的建議。	發生異常時接收通知。如需詳細資訊，請參閱 Amazon CodeGuru 使用者指南中的 使用異常和建議報告 。
Amazon DevOps Guru - 使用機器學習產生營運深入解析，協助您改善營運應用程式的效能。	轉傳深入解析和確認。如需詳細資訊，請參閱 AWS 管理與控管部落格中的 使用 PagerDuty 搭配 Amazon DevOpsGuru ，將採用 ML 技術的營運洞見傳遞給您的待命團隊。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Lookout for Metrics - 尋找資料中的異常、判斷其根本原因，並可讓您快速採取動作。	接收異常通知。如需詳細資訊，請參閱 Amazon Lookout for Metrics 開發人員指南中的 使用 Amazon SNS 搭配 Lookout for Metrics 。
Amazon Rekognition - 可讓您將影像與視訊分析新增至您的應用程式	接收請求結果的通知。如需詳細資訊，請參閱 Amazon Rekognition 開發人員指南中的 參考：影片分析結果通知 。
Amazon SageMaker AI - 可讓資料科學家和開發人員建置和訓練機器學習模型，然後將其直接部署到生產就緒的託管環境中。	標示資料物件時接收通知。如需詳細資訊，請參閱《Amazon SageMaker AI 開發人員指南》中的 建立串流標籤工作 。

管理與治理服務

下表說明 Amazon SNS 如何在聊天應用程式、CloudTrail AWS CloudFormation、CloudWatch、AWS Config AWS Control Tower、AWS Service Catalog、AWS License Manager 和中與 Amazon Q Developer 等 AWS 管理和控管服務整合 AWS Systems Manager，為基礎設施變更、合規提醒和營運洞察等重要事件提供通知。

這些整合透過為相關團隊和系統提供及時的提醒和更新，協助您有效監控和管理 AWS 環境。

AWS 服務	搭配使用 Amazon SNS 的優點
聊天應用程式中的 Amazon Q Developer - 可讓 DevOps 和軟體開發團隊使用 Amazon Chime 和 Slack 聊天室來監控和回應 中的操作事件 AWS 雲端。	將通知傳送至聊天室。如需詳細資訊，請參閱 《聊天應用程式管理員指南》中的在聊天應用程式中設定 Amazon Q 開發人員 。
AWS CloudFormation - 可讓您以可預測且重複的方式建立和佈建 AWS 基礎設施部署。	建立和更新堆疊時接收通知。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的 設定 AWS CloudFormation 堆疊選項 。
AWS CloudTrail - 提供您 AWS 帳戶 活動的事件記錄。	當 CloudTrail 發佈新的日誌檔案到您的 Amazon S3 儲存貯體時收到通知。如需詳細資訊，請

AWS 服務	搭配使用 Amazon SNS 的優點 參閱 AWS CloudTrail 使用者指南中的設定 CloudTrail 的 Amazon SNS 通知。
Amazon CloudWatch – AWS 即時監控您的 AWS 資源和您在 上執行的應用程式。	在警示變更狀態時收到通知。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南中的使用 Amazon CloudWatch 警示。
AWS Config – 提供 中資源組態 AWS 的詳細檢視 AWS 帳戶。	在資源更新或 AWS Config 會根據您的資源評估自訂或受管規則時接收通知。如需詳細資訊，請參閱《AWS Config 開發人員指南》中的AWS Config 傳送至 SNS 主題的通知和範例組態項目變更通知。
AWS Control Tower – 可讓您設定和管理安全、合規的多帳戶 AWS 環境。	使用警示可協助您防止在登陸區域內漂移，並接收法規遵循通知。如需詳細資訊，請參閱 AWS Control Tower 使用者指南中的透過 Amazon Simple Notification Service 追蹤警示。
AWS License Manager – 協助您集中管理來自軟體供應商的軟體授權，包括跨 AWS 和內部部署環境。	接收 License Manager 通知和警示。如需詳細資訊，請參閱《License Manager 使用者指南》中的 License Manager 中的設定，以及 AWS《管理與控管部落格》上的建立 AWS License Manager 通知的 ServiceNow 事件。
AWS Service Catalog - 可讓 IT 管理員建立、管理以及將已核准的產品組合分發給最終使用者，然後，最終使用者便可以在個人化入口網站存取他們所需的產品。	接收堆疊事件的通知。如需詳細資訊，請參閱《Service Catalog 管理員指南》中的 AWS Service Catalog 通知限制。
AWS Systems Manager – 可讓您檢視和控制您的基礎設施 AWS。	接收有關指令狀態的通知。如需詳細資訊，請參閱 AWS Systems Manager 使用者指南中的使用 Amazon SNS 通知監控 Systems Manager 狀態變更。

媒體服務

下表說明 Amazon SNS 如何與 Amazon Elastic Transcoder 整合，以在媒體轉碼任務變更狀態時傳送通知，讓您有效率地監控和管理存放在 Amazon S3 中的媒體檔案轉換為適合消費者播放裝置的格式。

此整合可提供任務狀態的即時提醒，協助您簡化媒體處理工作流程。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Elastic Transcoder - 可讓您將存放在 Amazon S3 的媒體檔案轉換成客戶播放裝置所要求格式的媒體檔案。	當工作狀態變更時接收通知。如需詳細資訊，請參閱Amazon Elastic Transcoder 開發人員指南中的 任務狀態通知 。

移轉與傳輸服務

下表說明 Amazon SNS 如何與 AWS 遷移和傳輸服務整合 AWS Application Discovery Service，例如 AWS Database Migration Service (DMS) AWS Snowball Edge，以及為伺服器資料收集、資料庫遷移活動和資料傳輸任務等事件提供通知。

這些整合透過提供關鍵遷移任務的即時提醒和更新，協助您有效管理和監控雲端遷移程序。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Application Discovery Service – AWS 雲端透過收集內部部署伺服器的用量和組態資料，協助您規劃遷移至。	透過接收事件通知 AWS CloudTrail。如需詳細資訊，請參閱 Application Discovery Service 使用者指南中的 使用 AWS CloudTrail記錄 Application Discovery Service API 呼叫 。
AWS Database Migration Service – 將資料從內部部署資料庫遷移至 AWS 雲端。	在 AWS DMS 事件發生時接收通知，例如建立或刪除複寫執行個體時。如需詳細資訊，請參閱 AWS Database Migration Service 使用者指南中的 在 AWS Database Migration Service中使用事件及通知 。
AWS Snowball Edge - 使用實體儲存裝置，在 Amazon S3 和您的現場資料儲存位置之間，透過優於網際網路的速度來傳輸大量資料。	接收 Snowball Edge 任務的通知。如需詳細資訊，請參閱AWS Snowball Edge 《使用者指南》中的 Snow Family 裝置的通知 。

聯網與內容交付服務

下表說明 Amazon SNS 如何與 AWS 聯網和內容交付服務整合，例如 Amazon API Gateway、Amazon CloudFront AWS Direct Connect、Elastic Load Balancing、Amazon Route 53 和 Amazon VPC，以傳送 API 訊息、CloudFront 指標警示、連線狀態變更、負載平衡器事件、運作狀態檢查狀態和 VPC 端點活動的通知。

這些整合透過提供及時的提醒和更新，協助您監控和管理網路和內容交付操作。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon API Gateway - 可讓您建立和部署任何規模的 REST 和 WebSocket API。	接收張貼至 API Gateway 端點的訊息。如需詳細資訊，請參閱《 API Gateway 開發人員指南 》中的教學課程： 建置具有 AWS 整合的 API Gateway REST API 。
Amazon CloudFront - 可更快速發佈靜態與動態 web 內容，例如 .html、.css、.php、映像與媒體檔案。	根據指定的 CloudFront 指標發生警示時接收通知。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的 設定警示以接收通知 。
AWS Direct Connect – 透過標準乙太網路光纖纜線，將內部網路連結至 AWS Direct Connect 位置。	當監控 AWS Direct Connect 連線的警示變更狀態時接收通知。如需詳細資訊，請參閱 AWS Direct Connect 《使用者指南》中的 建立 CloudWatch 警示以監控 AWS Direct Connect 連線 。
Elastic Load Balancing - 自動將傳入流量分配到多個可用區域的多個目標，例如 Amazon EC2 執行個體、容器和 IP 地址。	接收您為負載平衡器事件建立的警示通知。如需詳細資訊，請參閱 Classic Load Balancer 使用者指南中的 建立負載平衡器的 CloudWatch 警示 。
Amazon Route 53 - 提供網域註冊、DNS 路由和運作狀態檢查。	在運作狀態檢查狀態為不良時接收通知。如需詳細資訊，請參閱 Amazon Route 53 開發人員指南中的 若要在運作狀態檢查狀態為不良時收到 Amazon SNS 通知 (主控台) 。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Virtual Private Cloud (Amazon VPC) – 可讓您在已定義的虛擬網路中啟動 AWS 資源。	於界面端點發生特定事件時接收通知。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 建立和管理端點服務的通知 。

安全、身分與合規服務

下表說明 Amazon SNS 如何與 AWS 安全性、身分和合規服務整合，例如 AWS Directory Service Amazon GuardDuty、Amazon Inspector AWS Security Hub，以及提供目錄狀態變更、安全性問題清單、Inspector 事件和安全中樞公告的通知。

這些整合透過提供安全與合規事件的及時提醒和更新，協助您維護健全的安全實務。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Directory Service – 提供多種將 Microsoft Active Directory (AD) 與其他 搭配使用的方式 AWS 服務。	在目錄狀態變更時，收到電子郵件或文字 (SMS) 簡訊。如需詳細資訊，請參閱 AWS Directory Service 管理指南中的 設定目錄狀態通知 。
Amazon GuardDuty – 提供持續的安全監控，以協助識別 AWS 環境中未預期和可能未經授權或惡意的活動。	接收通知，了解最新發佈的問題清單類型、現有問題清單類型的更新以及其他功能改變。如需詳細資訊，請參閱 Amazon GuardDuty 使用者指南中的 訂閱 GuardDuty 公告 SNS 主題 。
Amazon Inspector - 測試 Amazon EC2 執行個體的聯網存取能力，以及這些執行個體上執行的應用程式安全性狀態。	接收 Amazon Inspector 事件的通知。如需詳細資訊，請參閱 Amazon Inspector 使用者指南中的 為 Amazon Elastic Inspector 通知設定 SNS 主題 。
AWS Security Hub - 自動執行 AWS 安全檢查並集中處理安全警示。	接收有關 AWS Security Hub 公告的通知，包括有關已新增、編輯或淘汰的 AWS Security Hub 控制項或標準的通知。如需詳細資訊，請參閱 使用 Amazon SNS 訂閱 AWS Security Hub 公告 。

無伺服器服務

下表說明 Amazon SNS 如何與 Amazon DynamoDB、Amazon EventBridge 和 Lambda 等服務整合，以傳送維護更新、即時資料串流和 Lambda 函數輸出等重要事件的通知。

這些整合可協助您接收關鍵操作的及時提醒，並將對這些事件的回應自動化，以有效率地監控和管理應用程式。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon DynamoDB - 是一項完全受管的 NoSQL 資料庫服務，可提供快速且可預期的效能及無縫的可擴展性。	在發生維護事件時接收通知。如需詳細資訊，請參閱 Amazon DynamoDB 開發人員指南中的 自訂 DAX 叢集設定 。
Amazon EventBridge – 從您自己的應用程式、software-as-a-service(SaaS) 應用程式提供即時資料串流，AWS 服務並將該資料路由至目標，包括 Amazon SNS。EventBridge 之前被稱為 CloudWatch Events。	接收事件 EventBridge 事件的通知。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的 Amazon EventBridge 目標 。
AWS Lambda - 可讓您直接執行程式碼，無需佈建或管理伺服器。	將 SNS 主題設定為 Lambda 無效字母佇列或 Lambda 目的地，以接收函數輸出資料。如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的 非同步叫用 。

儲存服務

下表說明 Amazon SNS 如何與 AWS 儲存服務整合 AWS Backup，例如 Amazon Elastic File System (EFS)、Amazon S3 Glacier、Amazon S3，以及為備份活動、檔案系統警示、資料擷取任務、儲存貯體變更和資料傳輸操作等各種事件 AWS Snowball Edge 提供通知。

這些整合可透過接收關鍵儲存事件的及時提醒，協助您有效率地監控和管理儲存解決方案。

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Backup – 協助您集中和自動化雲端和內部部署 AWS 服務 中跨 的資料備份	接收 AWS Backup 事件通知。如需詳細資訊，請參閱《AWS Backup 開發人員指南》中的 使用 Amazon SNS 追蹤 AWS Backup 事件 。

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Elastic File System - 為您的 Amazon EC2 執行個體提供檔案儲存。	接收您針對 Amazon EFS 事件建立的警示通知。如需詳細資訊，請參閱 Amazon Elastic File System 使用者指南中的 自動化監控工具 。
Amazon S3 Glacier - 為不常使用的資料提供儲存空間。	可以在文件庫中設定通知組態，以便在任務完成後將訊息傳送到 SNS 主題。如需詳細資訊，請參閱 Amazon S3 Glacier 開發人員指南中的 在 Amazon S3 Glacier 中設定文件庫通知 。
Amazon Simple Storage Service (Amazon S3) - 提供物件儲存。	當 Amazon S3 儲存貯體發生變更時，或在極少數的情況下，當物件未複製至其目的地區域時收到通知。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的 演練：設定通知的儲存貯體 (SNS 主題或 SQS 佇列) 和 監控複製指標和 Amazon S3 事件通知進度 。
AWS Snowball Edge - 使用實體儲存裝置，在 Amazon S3 和您的現場資料儲存位置之間，透過優於網際網路的速度來傳輸大量資料。	接收 Snowball Edge 任務的通知。如需詳細資訊，請參閱 AWS Snowball Edge 《使用者指南》中的 Snow Family 裝置的通知 。

其他事件來源

下表說明如何使用 Amazon SNS 接收有關 AWS 每日功能更新和 AWS IP 地址範圍變更的及時通知，確保您收到最新的 AWS 服務版本、執行個體類型、VPC 端點和公有 IP 地址變更的通知。

這些整合可協助您隨時掌握 AWS 基礎設施變更 up-to-date，並有效地管理您的資源。

來源	搭配使用 Amazon SNS 的優點
AWS 每日功能更新	透過 Amazon SNS 主題，即時接收有關 AWS 發布和更新的詳細資訊。這些版本包括 AWS 區域、AWS 服務、Amazon VPC 端點、AWS 服務與 AWS Service Quotas 整合、Amazon EC2 執行個體類型、Amazon SageMaker AI 執行個體類型、Amazon Nimble Studio 執行個體類

來源	搭配使用 Amazon SNS 的優點
	型、Amazon RDS 資料庫引擎版本和 Amazon MSK Apache Kafka 版本。如需詳細資訊，請參閱AWS 新聞部落格中的 透過 Amazon SNS 訂閱 AWS 每日功能更新 。
AWS IP 地址範圍	透過 Amazon SNS 主題接收 AWS IP 範圍變更的通知。如需詳細資訊，請參閱 中的 AWS IP 地址範圍通知 Amazon Web Services 一般參考，以及AWS 新聞部落格中的 透過 Amazon SNS 訂閱 AWS 公有 IP 地址變更 。

如需事件導向運算的詳細資訊，請參閱下列來源：

- [什麼是事件驅動的架構？](#)
- 運算部落格中的[使用 Amazon SNS 和 AWS 運算、儲存、資料庫和網路服務進行事件驅動](#)運算 AWS
- AWS 運算部落格上的[使用事件分叉管道豐富 AWS 事件驅動型架構](#)

Amazon SNS 事件目的地

本主題列出所有事件目的地，依[application-to-application\(A2A\) 訊息傳送](#)和[application-to-person\(A2P\) 通知](#)進行組織。

Note

Amazon SNS 在 2020 年 10 月推出 [FIFO 主題](#)。目前，大多數 AWS 服務僅支援從 SNS 標準主題接收事件。Amazon SQS 支援從 SNS 標準和 FIFO 主題接收事件。

A2A 目的地

下表說明 Amazon SNS 如何將事件交付至各種application-to-application(A2A) 目的地，例如 Amazon Data Firehose、Lambda、Amazon SQS、AWS 事件分叉管道和 HTTP/S 端點。

這些整合可讓您封存和分析資料、觸發自訂商業邏輯、促進應用程式整合，以及將事件路由至外部 Webhook，進而提升事件驅動型架構的效率和彈性。

事件目的地	搭配使用 Amazon SNS 的優點
Amazon Data Firehose	將事件傳遞至交付串流，以供封存和分析之用。透過交付串流，您可以將事件傳遞至 AWS 目的地，例如 Amazon Simple Storage Service (Amazon S3)、Amazon Redshift 和 Amazon OpenSearch Service (OpenSearch Service) 或第三方目的地，如 Datadog、New Relic、MongoDB 和 Splunk。如需詳細資訊，請參閱 散發到 Firehose 交付串流 。
AWS Lambda	將事件傳遞至函數，以觸發自訂商業邏輯的執行。如需詳細資訊，請參閱 將 Amazon SNS 通知散發至 Lambda 函數以進行自動化處理 。
Amazon SQS	將事件傳遞至佇列以供應用程式整合之用。如需詳細資訊，請參閱 將 Amazon SNS 通知散發至 Amazon SQS 佇列以進行非同步處理 。
AWS 事件分叉管道	將事件傳遞至事件備份與儲存、事件搜尋與分析，或事件重新執行管道。如需詳細資訊，請參閱 將 Amazon SNS 事件散發至 AWS 事件分叉管道 。
HTTP/S	將事件傳遞給外部網路掛鉤。如需詳細資訊，請參閱 將 Amazon SNS 通知散發至 HTTPS 端點 。

A2P 目的地

下表說明 Amazon SNS 如何透過 SMS 和原生推送通知、電子郵件收件匣、Amazon Chime 聊天室、Slack 管道，以及透過 PagerDuty 的營運洞察，將 application-to-person (A2P) 通知交付至各種目的地。

這些整合透過啟用跨多個平台和通訊管道的即時提醒和更新，來增強通訊和營運效率。

事件目的地	搭配使用 Amazon SNS 的優點
簡訊	以簡訊形式將事件傳遞至行動電話。如需詳細資訊，請參閱 使用 Amazon SNS 傳送行動簡訊 。
電子郵件	以電子郵件的形式傳遞事件至收件匣。如需詳細資訊，請參閱 Amazon SNS 電子郵件訂閱設定和管理 。
平台端點	將事件做為原生推送通知傳送至行動電話。如需詳細資訊，請參閱 使用 Amazon SNS 傳送行動推播通知 。
聊天應用程式中的 Amazon Q Developer	將活動傳遞至 Amazon Chime 聲聊天室或 Slack 頻道。如需詳細資訊，請參閱聊天應用程式管理員指南中的 Amazon Q 開發人員中的下列頁面： • 使用 Amazon Chime 在聊天應用程式中設定 Amazon Q Developer • 使用 Slack 在聊天應用程式中設定 Amazon Q Developer • 與其他 AWS 服務在聊天應用程式中使用 Amazon Q Developer
PagerDuty	將營運洞察提供給待命團隊。如需詳細資訊，請參閱 AWS 管理與治理部落格上的 透過使用 Amazon DevOps Guru 的 PagerDuty 將以 ML 為基礎的營運深入解析提供給您的待命團隊 。

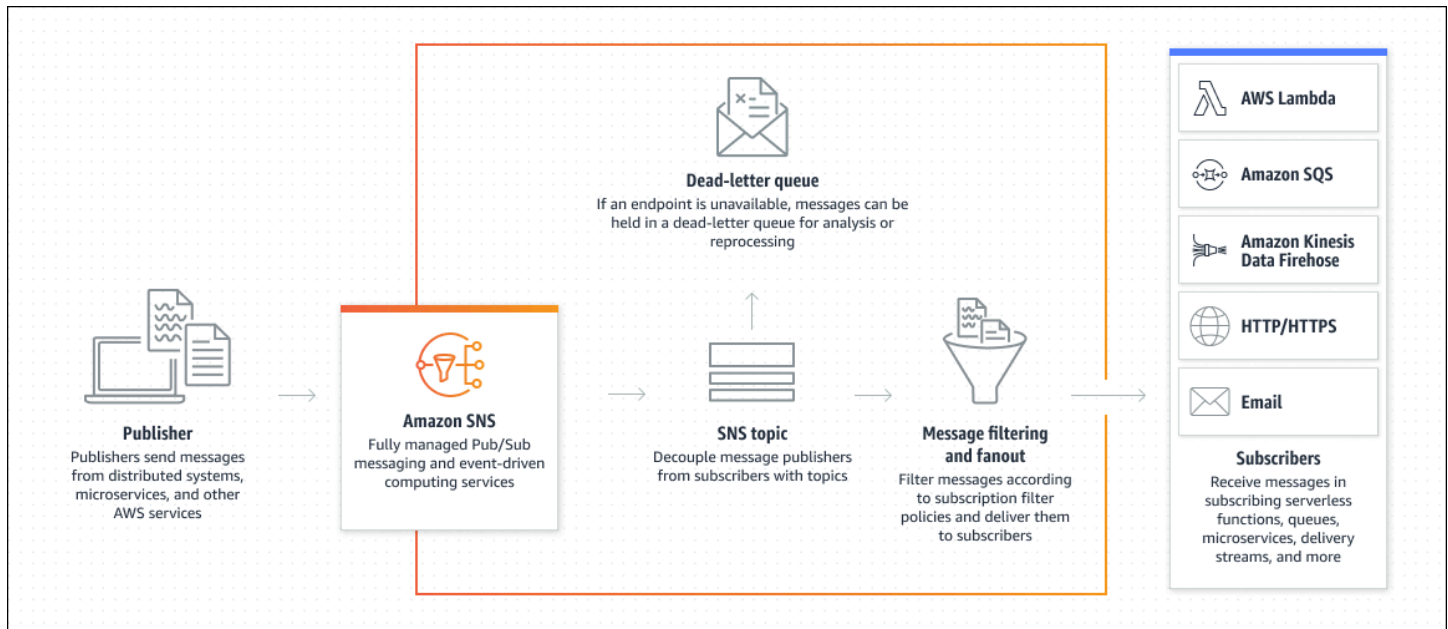
Note

您可以將原生 AWS 事件和自訂事件交付至聊天應用程式：

- 原生 AWS 事件 – 您可以在聊天應用程式中使用 Amazon Q Developer，透過 Amazon SNS 主題將原生 AWS 事件傳送至 Amazon Chime 和 Slack。支援的一組原生 AWS 事件包括來自 AWS 帳單與成本管理、AWS Health AWS CloudFormation、Amazon CloudWatch 等的事件。如需詳細資訊，請參閱 [《聊天應用程式管理員指南》中的在聊天應用程式中使用 Amazon Q 開發人員與其他服務](#)。
- 自訂事件 - 您也可以透過 Amazon SNS 主題將自訂事件傳送至 Amazon Chime、Slack 和 Microsoft 團隊。若要執行這項操作，您可以將自訂事件發佈至 SNS 主題，該主題將事件傳遞至已訂閱的 Lambda 函數。然後 Lambda 函數會使用聊天應用程式的 webhook 將事件傳遞給收件者。如需詳細資訊，請參閱[如何使用網路掛鉤將 Amazon SNS 訊息發佈到 Amazon Chime、Slack 或 Microsoft 團隊？](#)

將 Amazon SNS application-to-application 傳訊

Amazon SNS 透過將發佈者與支援微服務、分散式系統和無伺服器應用程式的訂閱者分開，簡化 application-to-application (A2A) 的訊息傳送。訊息會傳送到 Amazon SNS 主題，您可以在其中篩選並交付給訂閱者，例如 Lambda、Amazon SQS 或 HTTP 端點。如果傳遞失敗，訊息會存放在無效字母佇列中，以供進一步分析或重新處理。



散發到 Firehose 交付串流

您可以訂閱 [Amazon Data Firehose 交付串流](#) 到 Amazon SNS 主題，讓您將通知傳送到其他儲存和分析端點。發佈至 Amazon SNS 主題的訊息會傳送至訂閱的 Firehose 交付串流，並傳送至 Firehose 中設定的目的地。訂閱擁有者最多可將五個 Firehose 交付串流訂閱至 Amazon SNS 主題。每個 Firehose 交付串流都有請求和每秒輸送量的[預設配額](#)。這項限制可能會導致發佈的訊息數 (輸入流量) 多於傳遞的訊息數 (輸出流量)。當輸入流量超過輸出流量時，您的訂閱可能會累積大量的訊息待處理項目，便會導致郵件傳遞延遲過高。您可以根據發布率請求[增加配額](#)，以避免對工作負載造成不利影響。

透過 Firehose 交付串流，您可以將 Amazon SNS 通知分發給 Amazon Simple Storage Service (Amazon S3)、Amazon Redshift、Amazon OpenSearch Service (OpenSearch Service)，以及第三方服務供應商，例如 Datadog、New Relic、MongoDB 和 Splunk。

例如，您可以使用此功能將傳送至某個主題的訊息永久存放在 Amazon S3 儲存貯體中，以進行合規、封存或其他用途。若要這樣做，請使用 Amazon S3 儲存貯體目的地建立 Firehose 交付串流，並將該交付串流訂閱至 Amazon SNS 主題。另一個範例，若要對傳送至 Amazon SNS 主題的訊息執行分

析，請使用 OpenSearch Service 索引目標建立交付串流。然後，您可以將 Firehose 交付串流訂閱 Amazon SNS 主題。

Amazon SNS 也支援傳送至 Firehose 端點之通知的訊息傳遞狀態記錄。如需詳細資訊，請參閱[Amazon SNS 訊息傳遞狀態](#)。

訂閱 Firehose 交付串流至 Amazon SNS 主題的先決條件

若要訂閱 Amazon Data Firehose 交付串流至 SNS 主題，您的 AWS 帳戶 必須具有：

- 標準 SNS 主題。如需詳細資訊，請參閱[建立 Amazon SNS 主題](#)。
- Firehose 交付串流。如需詳細資訊，請參閱《[Amazon Data Firehose 開發人員指南](#)》中的[建立 Amazon Data Firehose 交付串流](#)和[授予應用程式存取 Firehose 資源的權限](#)。
- 信任 Amazon SNS 服務主體並具有寫入交付串流許可的 AWS Identity and Access Management (IAM) 角色。您將在建立訂閱時輸入此角色的 Amazon 資源名稱 (ARN) 做為 SubscriptionRoleARN。Amazon SNS 擔任此角色，允許 Amazon SNS 將記錄放入 Firehose 交付串流。

以下範例政策顯示建議的許可：

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

若要提供使用 Firehose 的完整許可，您也可以使用 AWS 受管政策 `AmazonKinesisFirehoseFullAccess`。或者，若要提供更嚴格的 Firehose 使用許可，您可以建立自己的政策。至少，政策必須提供在特定的交付串流上執行 `PutRecord` 操作的許可。

在所有情況下，您也必須編輯信任關係以包含 Amazon SNS 服務主體。例如：

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

如需建立角色的詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以將許可委派給 AWS 服務](#)。

完成這些需求後，您可以[將交付串流訂閱至 SNS 主題](#)。

訂閱 Firehose 交付串流至 Amazon SNS 主題

若要將 Amazon SNS 通知交付至 [Amazon Data Firehose 交付串流](#)，請先確定您已滿足所有[先決條件](#)。如需支援的端點清單，請參閱《》中的 [Amazon Data Firehose 端點和配額](#) Amazon Web Services 一般參考。

訂閱 Firehose 交付串流至主題

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選擇訂閱。
3. 在訂閱頁面，選擇建立訂閱。
4. 在建立訂閱頁面上，於詳細資訊區段中，執行以下作業：
 - a. 針對 Topic ARN (ARN 主題)，選擇標準主題的 Amazon 資源名稱 (ARN)。
 - b. 針對通訊協定，選擇 Firehose。

- c. 針對端點，選擇可從 Amazon SNS 接收通知的 Firehose 交付串流 ARN。
 - d. 針對訂閱角色 ARN，指定您為寫入 Firehose 交付串流所建立之 AWS Identity and Access Management (IAM) 角色的 ARN。如需詳細資訊，請參閱[訂閱 Firehose 交付串流至 Amazon SNS 主題的先決條件](#)。
 - e. (選擇性) 若要從已發佈訊息中移除任何 Amazon SNS 中繼資料，請選擇啟用原始訊息交付。如需詳細資訊，請參閱[Amazon SNS 原始訊息交付](#)。
5. (選用) 若要設定篩選政策，請展開 Subscription filter policy (訂閱篩選政策) 區段。如需詳細資訊，請參閱[Amazon SNS 訂閱篩選政策](#)。
 6. (選用) 若要設定訂閱的無效字母佇列，請展開 Redrive policy (dead-letter queue) (重新磁碟機政策 (無效字母佇列)) 區段。如需詳細資訊，請參閱[Amazon SNS 無效字母佇列](#)。
 7. 選擇 Create subscription (建立訂閱)。

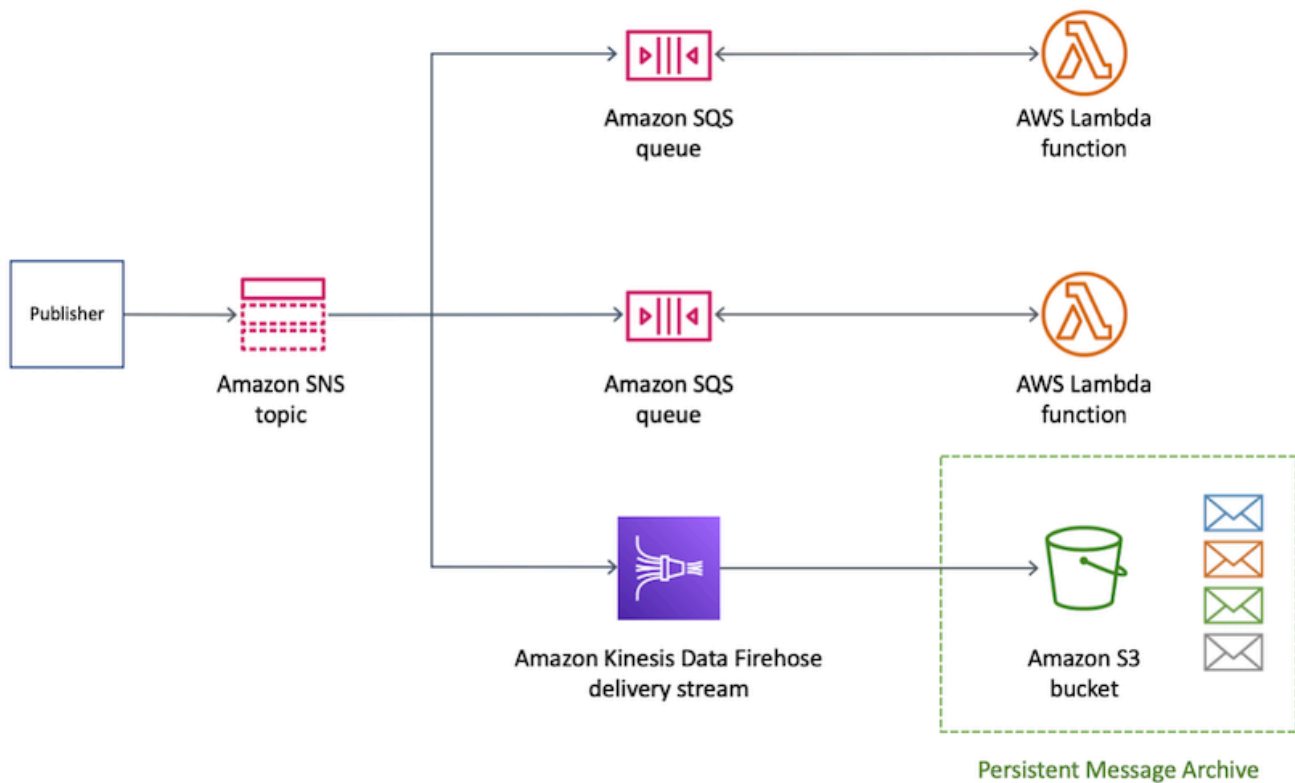
主控台會建立訂閱並開啟訂閱的 Details (詳細資訊) 頁面

跨多個交付串流目的地管理 Amazon SNS 訊息

[Amazon Data Firehose 交付串流](#)可讓您跨多個目的地管理 Amazon SNS 訊息，進而與 Amazon S3、Amazon OpenSearch Service、Amazon Redshift 和 HTTP 端點整合，以進行儲存、索引和分析。透過正確設定訊息格式和交付，您可以將 Amazon SNS 通知存放在 Amazon S3 中以供日後處理、使用 Amazon Athena 分析結構化訊息資料、在 OpenSearch 中編製索引訊息以進行即時搜尋和視覺化，以及在 Amazon Redshift 中封存結構以進行進階查詢。

在 Amazon S3 目的地中存放和分析 Amazon SNS 訊息 Amazon S3

本主題說明 Amazon Data Firehose 交付串流如何將資料發佈至 Amazon Simple Storage Service (Amazon S3)。



主題

- [格式化 Amazon SNS Amazon S3 SNS 通知](#)
- [使用 Athena 分析存放在 Amazon S3 中的 Amazon SNS 訊息 Amazon S3](#)

格式化 Amazon SNS Amazon S3 SNS 通知

下列範例顯示傳送至 Amazon Simple Storage Service (Amazon S3) 儲存貯體的 Amazon SNS 通知，其中包含可讀性縮排。Amazon S3

Note

在此範例中，已針對已發佈的訊息停用原始訊息交付。停用原始訊息交付時，Amazon SNS 會將 JSON 中繼資料新增至訊息，包括下列屬性：

- Type
- MessageId
- TopicArn
- Subject

- Timestamp
- UnsubscribeURL
- MessageAttributes

如需原始交付的詳細資訊，請參閱 [Amazon SNS 原始訊息交付](#)。

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

下列範例顯示透過 Amazon Data Firehose 交付串流傳送至相同 Amazon S3 儲存貯體的三則 SNS 訊息。會套用緩衝，且各行會分隔每個訊息。

```
{"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:sns:us-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"}
```

使用 Athena 分析存放在 Amazon S3 中的 Amazon SNS 訊息 Amazon S3

此頁面說明如何分析透過 Amazon Data Firehose 交付串流傳送至 Amazon Simple Storage Service (Amazon S3) 目的地的 Amazon SNS 訊息。Amazon S3

分析透過 Firehose 交付串流傳送至 Amazon S3 目的地的 SNS 訊息

1. 設定您的 Amazon S3 資源。如需指示，請參閱《Amazon Simple Storage Service 使用者指南》中的[建立儲存貯體](#)和《Amazon Simple Storage Service 使用者指南》中的[使用 Amazon S3 儲存貯體](#)。
2. 設定交付串流。如需說明，請參閱《[Amazon Data Firehose 開發人員指南](#)》中的為您的目的地選擇 [Amazon S3](#)。
3. 使用 [Amazon Athena](#) 查詢使用標準 SQL 的 Amazon S3 物件。如需詳細資訊，請參閱 Amazon Athena 使用者指南中的[入門](#)。

查詢範例

在本範例查詢中，假設下列情況：

- 訊息會儲存在 default 結構描述的 notifications 表格。
- 所以此 notifications 表格包含具有 string 類型的 timestamp 欄位。

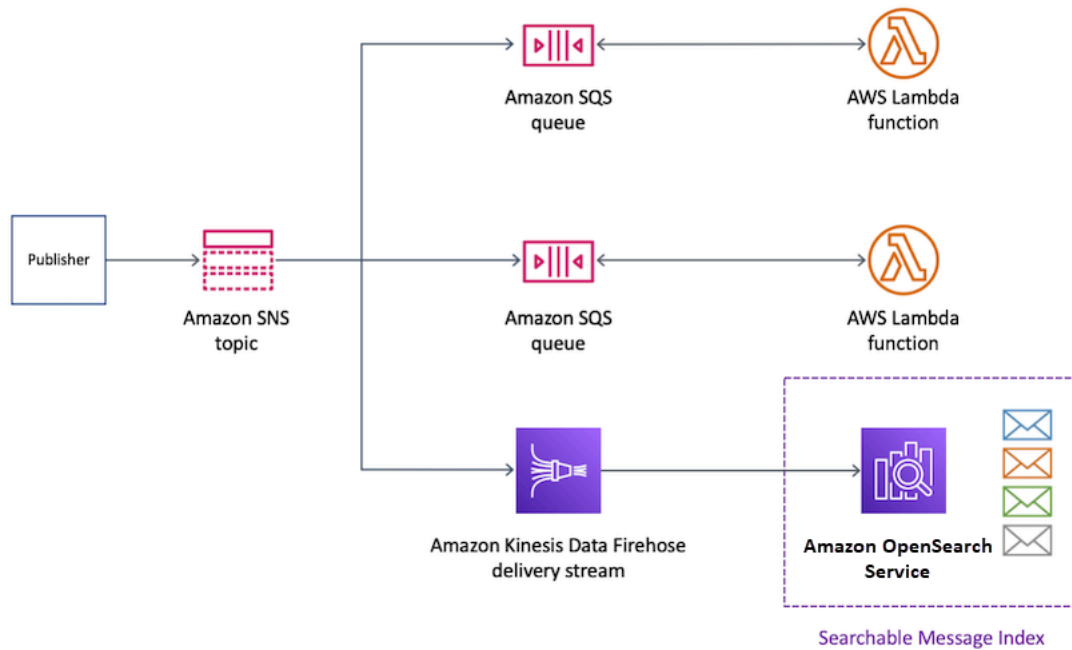
下列查詢會傳回在指定日期範圍內收到的所有 SNS 訊息：

```
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';
```

將 Amazon SNS 訊息與 Amazon OpenSearch Service 目的地整合

本節說明 Amazon Data Firehose 交付串流如何將資料發佈至 Amazon OpenSearch Service (OpenSearch Service)。



主題

- [在 OpenSearch Service 索引中存放和格式化 Amazon SNS 通知](#)
- [分析 OpenSearch Service 目的地的 Amazon SNS 訊息](#)

在 OpenSearch Service 索引中存放和格式化 Amazon SNS 通知

下列範例示範傳送至名為 Amazon OpenSearch Service (OpenSearch Service) 索引的 Amazon SNS 通知my-index。OpenSearch OpenSearch 此索引在 Timestamp 欄位。SNS 通知會放在 _source 屬性的酬載。

Note

在此範例中，已針對已發佈的訊息停用原始訊息交付。停用原始訊息交付時，Amazon SNS 會將 JSON 中繼資料新增至訊息，包括下列屬性：

- Type

- MessageId
- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

如需原始交付的詳細資訊，請參閱 [Amazon SNS 原始訊息交付](#)。

```
{
  "_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
  ]
}
```

```
]
}
```

分析 OpenSearch Service 目的地的 Amazon SNS 訊息

本主題說明如何分析透過 Amazon Data Firehose 交付串流傳送至 Amazon OpenSearch Service (OpenSearch Service) 目的地的 Amazon SNS 訊息。OpenSearch OpenSearch

分析透過 Firehose 交付串流傳送至 OpenSearch Service 目的地的 SNS 訊息

1. 設定您的 OpenSearch Service 資源。如需指示，請參閱 Amazon OpenSearch Service 開發人員指南中的 [Amazon OpenSearch Service 入門](#)。
2. 設定交付串流。如需說明，請參閱《Amazon Data Firehose 開發人員指南》中的 [為您的目的地選擇 OpenSearch Service](#)。
3. 執行使用 OpenSearch Service 查詢和 Kibana 的查詢。如需詳細資訊，請參閱 Amazon OpenSearch Service 開發人員指南中的 [步驟 3：在 OpenSearch Service 網域中搜尋文件和 Kibana](#)。

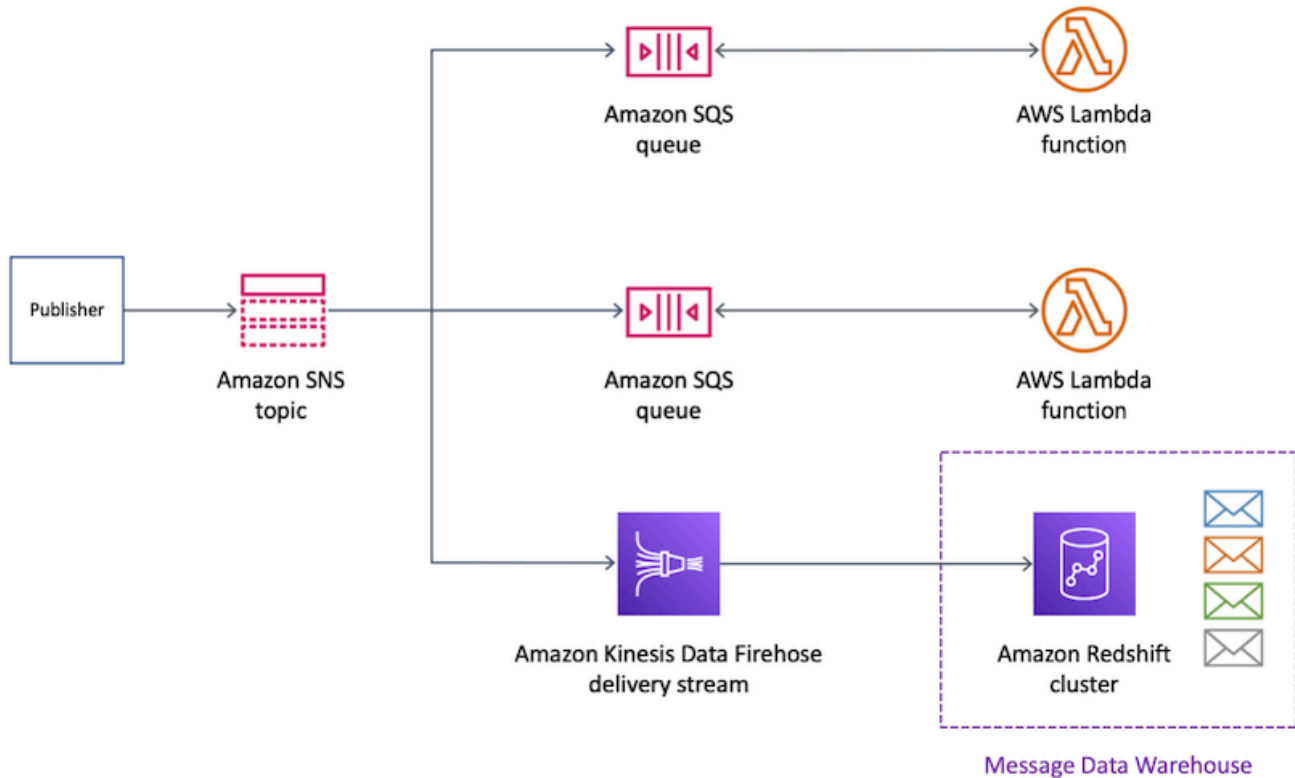
查詢範例

以下範例會查詢指定日期範圍內接收之所有 SNS 訊息的 my-index 索引：

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

在 Amazon Redshift 目的地中設定 Amazon SNS 訊息交付和分析

本主題說明如何將 Amazon SNS 通知散發至 Amazon Data Firehose 交付串流，然後發佈資料至 Amazon Redshift。透過此設定，您可以連線至 Amazon Redshift 資料庫，並使用 SQL 查詢工具擷取符合特定條件的 Amazon SNS 訊息。



主題

- [在 Amazon Redshift 資料表中結構化 Amazon SNS 訊息封存](#)
- [分析存放在 Amazon Redshift 目的地中的 Amazon SNS 訊息](#)

在 Amazon Redshift 資料表中結構化 Amazon SNS 訊息封存

對於 Amazon Redshift 端點，Amazon SNS 訊息會以資料表中的資料列封存。以下是如何存放資料的範例：

Note

在此範例中，已針對已發佈的訊息停用原始訊息交付。停用原始訊息交付時，Amazon SNS 會將 JSON 中繼資料新增至訊息，包括下列屬性：

- Type
- MessageId
- TopicArn
- Subject
- Message
- Timestamp
- UnsubscribeURL
- MessageAttributes

如需原始交付的詳細資訊，請參閱 [Amazon SNS 原始訊息交付](#)。

雖然 Amazon SNS 會使用此清單中顯示的大小寫將屬性新增至訊息，但 Amazon Redshift 表格中的欄名稱會以全部小寫字元顯示。若要轉換 Amazon Redshift 端點的 JSON 中繼資料，您可以使用 SQL COPY 命令。如需詳細資訊，請參閱 Amazon Redshift 資料庫開發人員指南中的 [從 JSON 範例複製](#) 和 [使用「auto 忽略」選項從 JSON 資料載入](#)。

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
通知	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	範例主旨	訊息範例	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:m	{"my_attribute\":\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						y-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	
通知	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	範例主旨2	範例訊息2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{"my_attribute2":{"Type":"String","Value":"my_value"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
通知	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	範例主旨3	範例訊息3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{"my_attribute3":{"Type":"String","Value":"my_value"}}

如需將通知展開到 Amazon Redshift 端點的詳細資訊，請參閱 [在 Amazon Redshift 目的地中設定 Amazon SNS 訊息交付和分析](#)。

分析存放在 Amazon Redshift 目的地中的 Amazon SNS 訊息

本主題說明如何分析透過 Amazon Data Firehose 交付串流傳送至 Amazon Redshift 目的地的 Amazon SNS 訊息。

分析透過 Firehose 交付串流傳送至 Amazon Redshift 目的地的 SNS 訊息

1. 設定您的 Amazon Redshift 資源。如需說明，請參閱 Amazon Redshift 入門指南中的 [Amazon Redshift 入門](#)。
2. 設定交付串流。如需說明，請參閱 [《Amazon Data Firehose 開發人員指南》](#) 中的為您的目的地選擇 [Amazon Redshift](#)。
3. 執行查詢。如需詳細資訊，請參閱 [《Amazon Redshift 管理指南》](#) 中的 [使用查詢編輯器來查詢資料庫](#)。

查詢範例

在本範例查詢中，假設下列情況：

- 訊息會儲存在預設 public 結構描述的 notifications 表格。
- 來自 SNS 訊息的 Timestamp 屬性會儲存在表格欄位資料類型為 timestampz 的 timestamp 欄。

Note

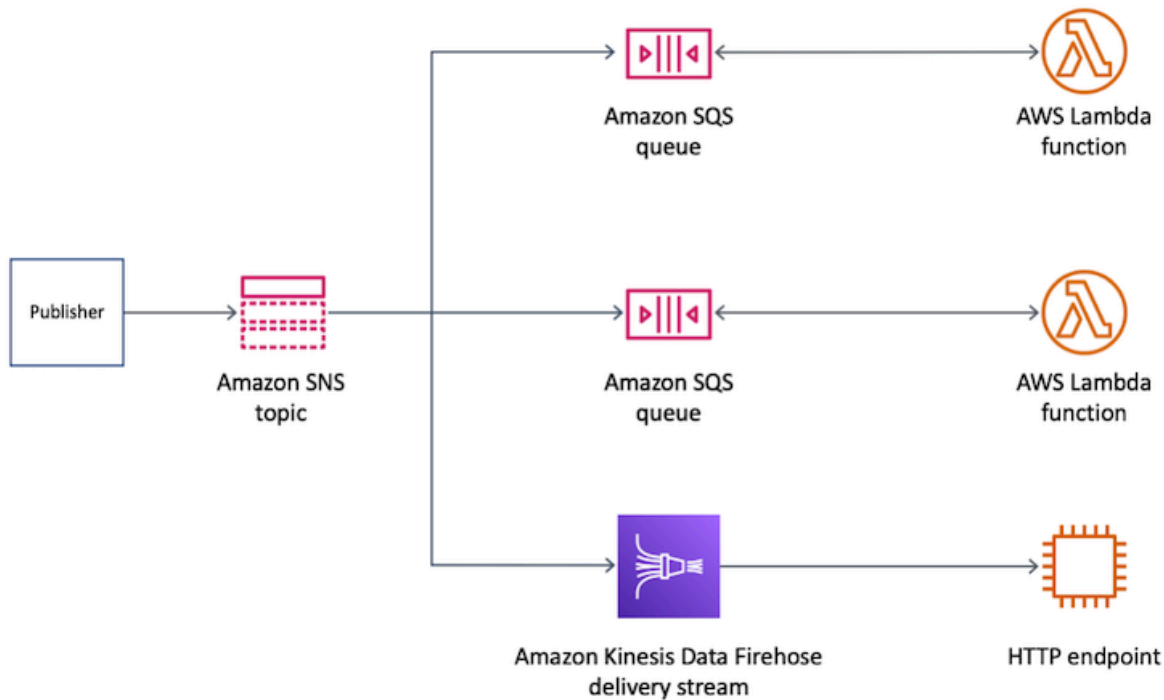
若要轉換 Amazon Redshift 端點的 JSON 中繼資料，您可以使用 SQL COPY 命令。如需詳細資訊，請參閱 Amazon Redshift 資料庫開發人員指南中的 [從 JSON 範例複製](#) 和 [使用「auto 忽略」選項從 JSON 資料載入](#)。

下列查詢會傳回在指定日期範圍內收到的所有 SNS 訊息：

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
'2020-12-02T09:00:00.000Z';
```

使用 Amazon Data Firehose 設定 Amazon SNS 訊息傳遞至 HTTP 目的地

本主題說明 Amazon Data Firehose 交付串流如何將資料發佈至 HTTP 端點。



主題

- [用於傳遞至 HTTP 目的地的 Amazon SNS 通知格式](#)

用於傳遞至 HTTP 目的地的 Amazon SNS 通知格式

以下是透過 Amazon Data Firehose 交付串流傳送至 HTTP 端點的 Amazon SNS HTTP POST 請求內文範例。Amazon SNS 通知在記錄屬性中編碼為 base64 承載。

Note

在此範例中，已針對已發佈的訊息停用原始訊息交付。如需原始交付的詳細資訊，請參閱 [Amazon SNS 原始訊息交付](#)。

```

"body": {
  "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
  "timestamp": 1606255960435,
  "records": [
    {

```

```
    "data":  
      "eyJUeXB1IjoiTm90aWZpY2F0aW9uIiwiaWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFnYzg0M"  
    }  
  ]  
}
```

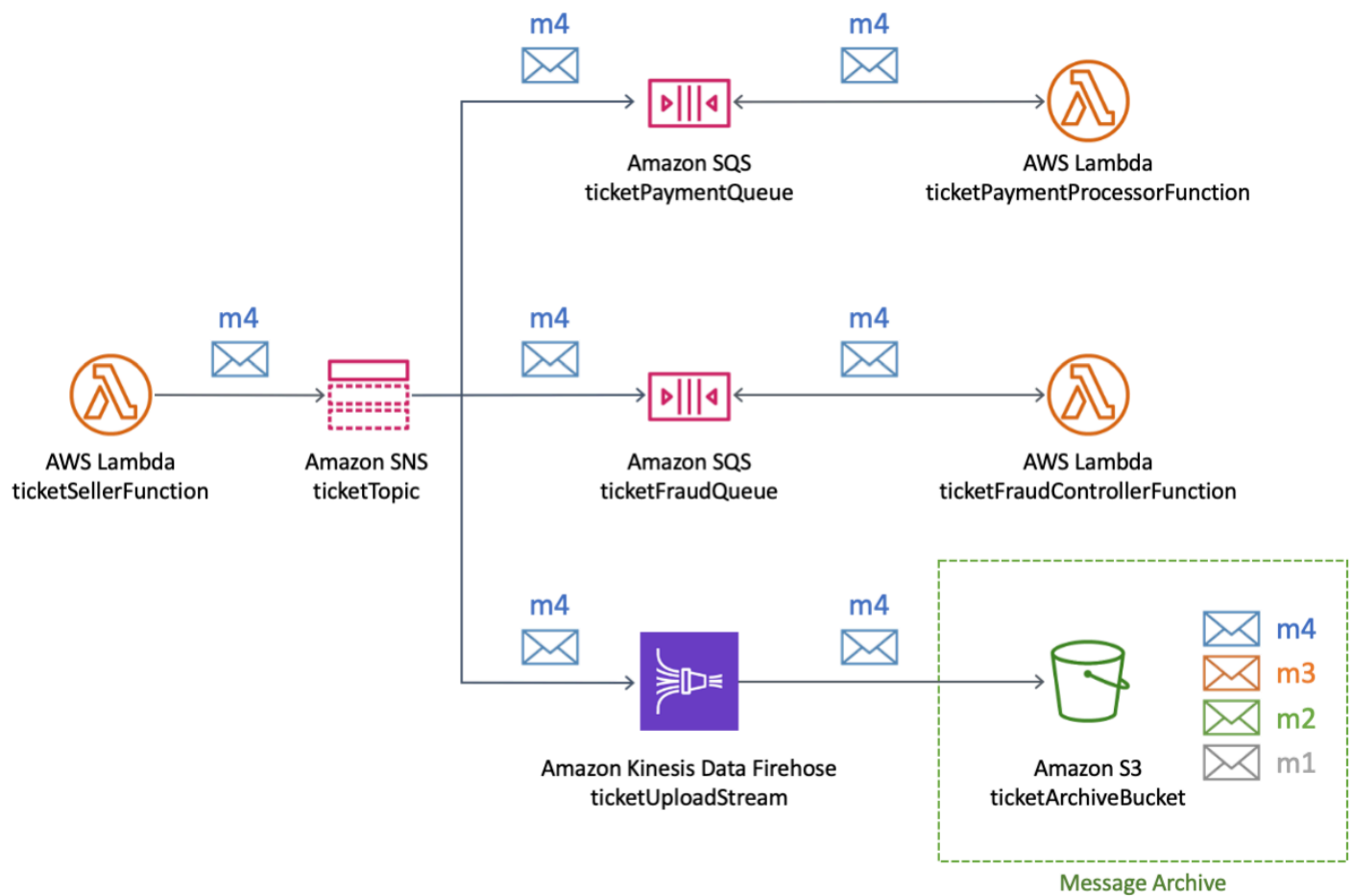
Amazon SNS 訊息封存和分析：航空公司售票平台的範例使用案例

本主題提供封存和分析 Amazon SNS 訊息之常見使用案例的教學課程。

此使用案例的設定是在受規範的環境中運作的航空公司票務平台。

1. 該平台受到合規架構的規範，該架構要求公司將所有門票銷售封存至少五年。
2. 為了符合資料保留的合規目標，公司會將 Amazon Data Firehose 交付串流訂閱至現有的 Amazon SNS 主題。
3. 交付串流的目標是 Amazon Simple Storage Service (Amazon S3) 儲存貯體。透過此組態，發佈至 SNS 主題的所有事件都會封存在 Amazon S3 儲存貯體中。

以下圖表說明此組態的架構：



為了執行分析並獲得門票銷售的見解，該公司使用 Amazon Athena 進行 SQL 查詢。例如，公司可以查詢以了解最受歡迎的目的地和最常旅客。

若要建立此使用案例 AWS 的資源，您可以使用 AWS Management Console 或 AWS CloudFormation 範本。

主題

- [設定 Amazon SNS 訊息封存和分析的初始 AWS 資源](#)
- [設定 Firehose 交付串流以進行 Amazon SNS 訊息封存](#)
- [訂閱 Firehose 交付串流至 Amazon SNS 主題](#)
- [測試和查詢 Amazon SNS 組態，以進行有效的資料管理](#)
- [使用 AWS CloudFormation 範本自動化 Amazon SNS 訊息封存](#)

設定 Amazon SNS 訊息封存和分析的初始 AWS 資源

本主題說明如何建立[訊息封存和分析範例使用案例](#)所需的資源：

- 一個 Amazon Simple Storage Service (Amazon S3) 儲存貯體
- 兩個 Amazon Simple Queue Service (Amazon SQS) 佇列
- Amazon SNS 主題
- 兩個 Amazon SQS 訂閱的 Amazon SNS 主題

建立初始資源

1. 建立 Amazon S3 儲存貯體：

- a. 開啟 [Amazon S3 主控台](#)。
- b. 選擇 建立儲存貯體。
- c. 對於 Bucket name (儲存貯體名稱)，輸入一個唯一名稱。將其他欄位保留為預設值。
- d. 選擇建立儲存貯體。

如需 Amazon S3 儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[建立儲存貯體](#)和《Amazon Simple Storage Service 使用者指南》中的[使用 Amazon S3 儲存貯體](#)。

2. 建立兩個 Amazon SQS 佇列：

- a. 開啟 [Amazon SQS 主控台](#)。
- b. 選擇建立佇列。
- c. 針對類型，選擇標準。
- d. 對於名稱，輸入 **ticketPaymentQueue**。
- e. 在 Access policy (存取政策) 下方的 Choose method (選擇方法) 中，選擇 Advanced (進階)。
- f. 在 JSON 政策方塊中，貼上下列政策：

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
    "Service": "sns.amazonaws.com"
  },
  "Action": "sqs:SendMessage",
  "Resource": "*",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:ticketTopic"
    }
  }
}
]
```

在此存取政策中，將 AWS 帳戶 號碼 (*123456789012*) 取代為您自己的號碼，並相應地變更 AWS 區域 (*us-east-1*)。

- g. 選擇建立佇列。
- h. 重複這些步驟，建立第二個名為 **ticketFraudQueue** 的 SQS 佇列。

如需建立 SQS 佇列的詳細資訊，請參閱 Amazon Simple Queue Data 開發人員指南中的[建立 Amazon SQS 佇列 \(主控台\)](#)。

3. 建立 SNS 主題：

- a. 在 Amazon SNS 主控台開啟 [Topics](#) (主題) 頁面。
- b. 請選擇建立主題。
- c. 在 Details (詳細資訊) 下方的 Type (類型) 中，選擇 Standard (標準)。
- d. 對於名稱，輸入 **ticketTopic**。
- e. 請選擇建立主題。

如需建立 SNS 主題的詳細資訊，請參閱 [建立 Amazon SNS 主題](#)。

4. 訂閱 SQS 佇列至 SNS 主題：

- a. 在 [Amazon SNS 主控台](#) 中的 ticketTopic 主題的詳細資訊頁面，選擇 Create subscription (建立訂閱)。
- b. 在 Details (詳細資訊) 的 Protocol (通訊協定) 中，選擇 Amazon SQS。
- c. 針對 Endpoint (端點)，選擇 ticketPaymentQueue 佇列的 Amazon 資源名稱 (ARN)。
- d. 選擇建立訂閱。

- e. 重複這些步驟來建立使用 `ticketFraudQueue` 佇列的 ARN 的第二個訂閱。

如需訂閱 SNS 主題的詳細資訊，請參閱 [建立 Amazon SNS 主題的訂閱](#)。您也可以從 Amazon SQS 主控台訂閱 SQS 佇列至 SNS 主題。如需詳細資訊，請參閱 Amazon Simple Queue Service 開發人員指南中的 [訂閱 Amazon SNS 主題 \(主控台\) 的 Amazon SQS 佇列](#)。

您已為此範例使用案例建立初始資源。若要繼續，請參閱 [設定 Firehose 交付串流以進行 Amazon SNS 訊息封存](#)。

設定 Firehose 交付串流以進行 Amazon SNS 訊息封存

本主題說明如何為[訊息封存和分析範例使用案例](#)建立 Amazon Data Firehose 交付串流。

建立 Firehose 交付串流

1. 開啟 [Amazon Kinesis 服務主控台](#)。
2. 選擇 Firehose，然後選擇建立交付串流。
3. 在 New delivery stream (新增交付串流) 頁面上的 Delivery stream name (交付串流名稱)，輸入 **ticketUploadStream**，然後選擇 Next (下一步)。
4. 在 Process records (處理記錄) 頁面上，選擇 Next (下一步)。
5. 在 Choose a Destination (選擇目的地) 頁面上，執行下列動作：
 - a. 針對 Destination (目的地)，選擇 Amazon S3。
 - b. 在 S3 destination (S3 目的地) 下方的 S3 bucket (S3 儲存貯體)，選擇您[最初建立](#)的 S3 儲存貯體。
 - c. 選擇 Next (下一步)。
6. 在 Configure settings (組態設定) 頁面的 S3 buffer conditions (S3 緩衝區條件)，請執行下列動作：
 - 對於 Buffer size (緩衝大小)，輸入 **1**。
 - 對於 Buffer interval (緩衝區間隔)，輸入 **60**。

將這些值用於 Amazon S3 緩衝區可讓您快速測試組態。第一個滿足的緩衝區條件會觸發 S3 儲存貯體的交付資料。

7. 在設定設定頁面上，針對許可，選擇使用自動指派的必要許可建立 AWS Identity and Access Management (IAM) 角色。然後選擇下一步。

8. 在 Review (檢閱) 頁面上，選擇 Create delivery stream (建立交付串流)。
9. 從 Kinesis Data Firehose 交付串流頁面選擇您剛建立的交付串流 (ticketUploadStream)。在 Details (詳細資訊) 索引標籤上，記下串流的 Amazon 資源名稱 (ARN) 以供日後使用。

如需建立交付串流的詳細資訊，請參閱 [《Amazon Data Firehose 開發人員指南》](#) 中的 [建立 Amazon Data Firehose 交付串流](#)。如需建立 IAM 角色的詳細資訊，請參閱 [《IAM 使用者指南》](#) 中的 [建立角色以將許可委派給 AWS 服務](#)。

您已使用所需的許可建立 Firehose 交付串流。若要繼續，請參閱 [訂閱 Firehose 交付串流至 Amazon SNS 主題](#)。

訂閱 Firehose 交付串流至 Amazon SNS 主題

本主題說明如何為[訊息封存和分析範例使用案例](#)建立下列資源：

- 允許 Amazon SNS 訂閱將記錄放置在 Amazon Data Firehose 交付串流上的 AWS Identity and Access Management (IAM) 角色。
- Amazon SNS 主題的 Firehose 交付串流訂閱。

為 Amazon SNS 訂閱建立 IAM 角色

1. 開啟 IAM 主控台中的 [Roles page](#) (角色頁面)。
2. 選擇 建立角色。
3. 對於 Select type of trusted entity (選取信任的實體類型)，選擇 AWS service (AWS 服務)。
4. 對於 Choose a use case (選擇使用案例)，選擇 SNS。然後選擇下一步：許可。
5. 選擇 Next: Tags (下一步：標籤)。
6. 選擇 Next: Review (下一步：檢閱)。
7. 在 Review (檢閱) 頁面，針對 Role name (角色名稱) 輸入 **ticketUploadStreamSubscriptionRole**。然後選擇 Create role (建立角色)。
8. 建立角色時，請選擇其名稱 (ticketUploadStreamSubscriptionRole)。
9. 在角色的 Summary (摘要) 頁面上，選擇 Add inline policy (新增內嵌政策)。
10. 在 Create policy (建立政策) 頁面上，選擇 JSON 索引標籤，然後張貼以下政策至文字方塊中：

```
{  
  "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"
  }
]
```

在此政策中，將 AWS 帳戶 數字 (*123456789012*) 取代為您自己的數字，並相應地變更 AWS 區域 (*us-east-1*)。

11. 選擇 Review policy (檢閱政策)。
12. 在 Review policy (檢閱政策) 頁面的 Name (名稱) 中，輸入 **FirehoseSnsPolicy**。然後選擇 Create policy (建立政策)。
13. 在角色的 Summary (摘要) 頁面上，注意角色 ARN 供日後使用。

如需建立 IAM 角色的詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可給 AWS 服務](#)。

將 Firehose 交付串流訂閱 SNS 主題

1. 在 Amazon SNS 主控台開啟 [Topics](#) (主題) 頁面。
2. 在 Subscriptions (訂閱) 索引標籤上，選擇 Create subscription (建立訂閱)。
3. 在詳細資訊下，針對通訊協定選擇 Amazon Data Firehose。
4. 對於 Endpoint (端點)，輸入 ticketUploadStream 的 Amazon 資源名稱 (ARN)，您先前建立的交付串流。例如，輸入 **arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream**。
5. 對於 Subscription role ARN (訂閱角色 ARN)，輸入 ticketUploadStreamSubscriptionRole 的 ARN，您先前建立的 IAM 角色。例如，輸入 **arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole**。

6. 選取 Enable raw message delivery (啟用原始訊息交付) 核取方塊。
7. 選擇 Create subscription (建立訂閱)。

您已建立 IAM 角色和 SNS 主題訂閱。若要繼續，請參閱 [測試和查詢 Amazon SNS 組態，以進行有效的資料管理](#)。

測試和查詢 Amazon SNS 組態，以進行有效的資料管理

本主題說明如何透過將[訊息發佈至 Amazon SNS 主題來測試訊息封存和分析範例使用案例](#)。Amazon SNS 這些指示包括一個範例查詢，您可以執行並適應自己的需求。

若要測試組態

1. 在 Amazon SNS 主控台開啟 [Topics](#) (主題) 頁面。
2. 選擇 **ticketTopic** 主題。
3. 選擇 Publish message (發佈訊息)。
4. 在 Publish message to topic (將訊息發佈至主題) 頁面上，輸入訊息內文的下列資訊。在訊息的結尾新增換行字元。

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15  
04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
```

所有其他選項保持為預設值。

5. 選擇 Publish message (發佈訊息)。

如需發佈郵件的詳細資訊，請參閱 [發佈 Amazon SNS 訊息](#)。

6. 在 60 秒的交付串流間隔之後，開啟 [Amazon Simple Storage Service \(Amazon S3\) 主控台](#) 並選擇您[最初建立](#)的 Amazon S3 儲存貯體。

發佈的訊息會出現在儲存貯體中。

查詢資料

1. 開啟 [Amazon Athena 主控台](#)。
2. 執行查詢。

例如，假設 default 架構中的 notifications 表格包含下列資料：

```
{"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"}
```

若要尋找第一目的地，執行下列查詢：

```
SELECT destination  
FROM default.notifications  
GROUP BY destination  
ORDER BY count(*) desc  
LIMIT 1;
```

若要查詢特定日期和時間範圍內銷售的票證，請執行類似下列的查詢：

```
SELECT *  
FROM default.notifications  
WHERE bookingtime  
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'  
  AND TIMESTAMP '2020-12-15 12:00:00';
```

您可以根據自己的需求調整這兩個範例查詢。如需使用 Athena 執行查詢的詳細資訊，請參閱 Amazon Athena 使用者指南中的[入門](#)。

清除

若要避免在完成測試後產生使用費用，請刪除您在教學課程期間建立的下列資源：

- Amazon SNS 訂閱
- Amazon SNS 主題
- Amazon Simple Queue Service (Amazon SQS) 佇列
- Amazon S3 儲存貯體
- Amazon Data Firehose 交付串流
- AWS Identity and Access Management (IAM) 角色和政策

使用 AWS CloudFormation 範本自動化 Amazon SNS 訊息封存

若要自動部署 Amazon SNS [訊息封存和分析範例使用案例](#)，您可以使用以下 YAML 範本：

```
---
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

```

將 Amazon SNS 通知散發至 Lambda 函數以進行自動化處理

Amazon SNS 與整合 AWS Lambda，可讓您觸發 Lambda 函數以回應 Amazon SNS 通知。當訊息發佈到 Lambda 函數訂閱的 SNS 主題時，將使用已發佈訊息的酬載來叫用 Lambda 函數。Lambda 函數會接收訊息承載做為輸入參數，並可操作訊息中的資訊、將訊息發佈至其他 SNS 主題，或將訊息傳送至其他 AWS 服務。

此外，Amazon SNS 也針對傳送到 Lambda 端點的訊息通知，支援其訊息傳遞狀態屬性。如需詳細資訊，請參閱[Amazon SNS 訊息傳遞狀態](#)。

主題

- [跨區域整合 Amazon SNS 與 Lambda 函數的先決條件](#)
- [訂閱 Lambda 函數至 Amazon SNS 主題](#)

跨區域整合 Amazon SNS 與 Lambda 函數的先決條件

若要使用 Amazon SNS 通知叫用 Lambda 函數，您需要下列項目：

- Lambda 函數
- Amazon SNS 主題

如需建立 Lambda 函數以搭配使用的相 Amazon SNS 資訊，請參閱[搭配使用 Lambda 與 Amazon SNS](#)。如需關於建立 Amazon SNS 主題的詳細資訊，請參閱[建立 SNS 主題](#)。

當您使用 Amazon SNS 將訊息從選擇加入區域傳遞至預設啟用的區域時，必須將委託人 `sns.amazonaws.com` 取代為 `sns.<opt-in-region>.amazonaws.com`，以變更在 AWS Lambda 函數中建立的政策。

例如，如果您想要讓美國東部 (維吉尼亞北部) 的 Lambda 函數訂閱亞太區域 (香港) 的 SNS 主題，請將 AWS Lambda 函數政策中的委託人變更為 `sns.ap-east-1.amazonaws.com`。選擇加入區域包括 2019 年 3 月 20 日後推出的任何區域，其中包括亞太區域 (香港)、中東 (巴林)、歐洲 (米蘭) 和非洲 (開普敦)。2019 年 3 月 20 日之前推出的區域為預設啟用。

Note

AWS 不支援從預設為選擇加入區域啟用的區域跨區域交付至 Lambda。此外，也不支援從選擇加入區域到其他選擇加入區域的跨區域轉送 SNS 訊息。

訂閱 Lambda 函數至 Amazon SNS 主題

本主題說明如何將 Lambda 函數訂閱至 Amazon SNS 主題，讓該函數可由已發佈的訊息觸發。

1. 登入 [Amazon SNS 主控台](#)。

2. 在導覽面板上，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選擇主題。
4. 在 Subscription (訂閱) 區段中，選擇 Create subscription (建立訂閱)。
5. 在建立訂閱頁面上，於詳細資訊區段中，執行以下作業：
 - a. 驗證所選擇的 Topic ARN (主題 ARN)。
 - b. 針對通訊協定選擇 AWS Lambda。
 - c. 針對 Endpoint (端點)，輸入函數的 ARN。
 - d. 選擇 Create subscription (建立訂閱)。

當訊息發佈到 Lambda 函數訂閱的 SNS 主題時，將使用已發佈訊息的酬載來叫用 Lambda 函數。如需如何 AWS Lambda 搭配 Amazon SNS 使用的資訊，包括教學課程，請參閱[AWS Lambda 搭配 Amazon SNS 使用](#)。

將 Amazon SNS 通知散發至 Amazon SQS 佇列以進行非同步處理

[Amazon SNS](#) 與 Amazon Simple Queue Service (Amazon SQS) 緊密合作。這些服務提供開發人員不同優勢。Amazon SNS 可透過「推播」機制讓應用程式傳送分秒必爭的訊息給多個訂閱者，免除定期檢查或「輪詢」更新的需要。Amazon SQS 是一項訊息佇列服務，由所發佈的應用程式用來透過輪詢模式來交換訊息，並且可用來斷開傳送元件和接收元件的連結，而無需每個元件都同時可供使用。透過一起使用 Amazon SNS 和 Amazon SQS，訊息即可交付到需要立即事件通知的應用程式，並且也會保存在 Amazon SQS 佇列中以便稍後供其他應用程式處理使用。

當您訂閱 Amazon SQS 佇列到 Amazon SNS 主題時，您可以發佈訊息到主題，而 Amazon SNS 會傳送 Amazon SQS 訊息到訂閱的佇列。Amazon SQS 訊息包含已發佈至主題的主旨和訊息，同時包含有關 JSON 文件中訊息的中繼資料。Amazon SQS 訊息將看起來類似以下 JSON 文件。

```
{
  "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"
}
```

為 Amazon SQS 佇列訂閱 Amazon SNS 主題

若要讓 Amazon SNS 主題傳送訊息至 Amazon SQS 佇列，請選擇下列其中一項：

- 使用 [Amazon SQS 主控台](#)，這簡化了這個過程。如需詳細資訊，請參閱 Amazon Simple Queue Service 開發人員指南中的[訂閱 Amazon SNS 主題的 Amazon SQS 佇列](#)。
- 使用下列步驟：
 1. [取得您要傳送訊息的目標佇列的 Amazon Resource Name \(ARN\)](#)，以及您要訂閱佇列的主題。
 2. [將 sqs:SendMessage 許可提供給 Amazon SNS 主題](#)，以便其可以將訊息傳送至佇列。
 3. [訂閱佇列至 Amazon SNS 主題](#)
 4. [提供 IAM 使用者或 AWS 帳戶 適當的許可，以發佈至 Amazon SNS 主題，並讀取來自 Amazon SQS 佇列的訊息。](#)
 5. [將訊息發佈至主題並讀取來自佇列的訊息以進行測試。](#)

若要了解如何設定主題以傳送訊息至不同 AWS 帳戶中的佇列，請參閱 [傳送 Amazon SNS 訊息至其他帳戶中的 Amazon SQS 佇列](#)。

若要查看建立將訊息傳送至兩個佇列之主題的 AWS CloudFormation 範本，請參閱 [使用將 Amazon SNS 自動化至 Amazon SQS 訊息 AWS CloudFormation](#)。

步驟 1：取得佇列和主題的 ARN

訂閱佇列到您的主題時，您需要一份佇列的 ARN 副本。同樣地，提供許可給主題以傳送訊息至佇列時，您需要一份主題的 ARN 副本。

若要取得佇列 ARN，您可以使用 Amazon SQS 主控台或 [GetQueueAttributes](#) API 動作。

從 Amazon SQS 主控台取得佇列 ARN

1. 登入 AWS Management Console 並開啟位於 <https://console.aws.amazon.com/sqs/> Amazon SQS 主控台。
2. 選取您要取得其 ARN 之佇列的方塊。

3. 從 Details (詳細資訊) 區段，複製 ARN 值，以便您可以用它訂閱到 Amazon SNS 主題。

若要取得主題 ARN，您可以使用 Amazon SNS 主控台、[sns-get-topic-attributes](#) 命令或 [GetQueueAttributes](#) API 動作。

從 Amazon SNS 主控台取得主題 ARN

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇您要取得 ARN 的主題。
3. 從 Details (詳細資訊) 區段，複製 ARN 值，以便您可以用它將許可提供給 Amazon SNS 主題以將訊息傳送至佇列。

步驟 2：將許可提供給 Amazon SNS 主題，以將訊息傳送至 Amazon SQS 佇列。

為了讓 Amazon SNS 主題能夠傳送訊息至佇列，您必須對佇列設定政策，允許 Amazon SNS 主題執行 `sqs:SendMessage` 動作。

在您訂閱佇列到主題之前，您需要主題和佇列。如果您尚未建立主題或佇列，請現在建立。如需詳細資訊，請參閱 [建立主題](#)，並參閱 Amazon Queue Service 開發人員指南中的 [建立佇列](#)。

若要取得有關佇列的政策，您可以使用 Amazon SQS 主控台或 [SetQueueAttributes](#) API 動作。開始之前，請確定您擁有您想要允許傳送訊息至佇列之主題的 ARN。如果您要訂閱佇列至多個主題，則政策必須為每個主題包含一個 Statement 元素。

使用 Amazon SQS 主控台設定佇列的 SendMessage 政策

1. 登入 AWS Management Console 並開啟位於 <https://console.aws.amazon.com/sqs/> Amazon SQS 主控台。
2. 選取您要設定其政策之佇列的方塊，選擇 Access policy (存取政策) 索引標籤，然後選擇 Edit (編輯)。
3. 在存取政策區段中，定義誰可以存取您的佇列。
 - 新增條件以允許用於主題的動作。
 - 將 Principal 設定為 Amazon SNS 服務，如下列範例所示。
 - 使用 [aws:SourceArn](#) 或者 [aws:SourceAccount](#) 全域條件金鑰，以防止[混淆代理人](#)案例。如要使用這些條件金鑰，請將值設定為主題的 ARN。若您的佇列訂閱了多個主題，則可改用 `aws:SourceAccount`。

例如，下列政策允許 MyTopic 傳送訊息至 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"
        }
      }
    }
  ]
}
```

步驟 3：將佇列訂閱至 Amazon SNS 主題

若要透過主題傳送訊息至佇列，您必須訂閱佇列至 Amazon SNS 主題。您按照其 ARN 指定佇列。若要訂閱主題，您可以使用 Amazon SNS 主控台、[sns-subscribe](#) CLI 命令或 [Subscribe](#) API 動作。開始之前，請確定您擁有想要訂閱佇列的 ARN。

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選擇主題。
4. 在 **MyTopic** (我的主題) 頁面上，於 Subscriptions (訂閱) 頁面中，選擇 Create subscription (建立訂閱)。
5. 在建立訂閱頁面上，於詳細資訊區段中，執行以下作業：
 - a. 驗證 Topic ARN (主題 ARN)。
 - b. 在 Protocol (通訊協定) 中，選擇 Amazon SQS。
 - c. 針對 Endpoint (端點)，輸入 Amazon SQS 佇列的 ARN。
 - d. 選擇 Create Subscription (建立訂閱)。

確認訂閱之後，您新訂閱的 Subscription ID (訂閱 ID) 就會顯示其訂閱 ID。如果佇列的擁有者建議訂閱，訂閱會自動確認並且訂閱應該立即啟用。

通常，您會訂閱您自己的佇列到您帳戶中您自己的主題。不過，您也可以從不同帳戶訂閱佇列到您的主題。如果建立訂閱的使用者不是佇列的擁有者 (例如，帳戶 A 的使用者訂閱帳戶 B 的佇列到帳戶 A 中的主題)，訂閱必須經過確認。如需有關訂閱不同帳戶的佇列和確認訂閱的詳細資訊，請參閱 [傳送 Amazon SNS 訊息至其他帳戶中的 Amazon SQS 佇列](#)。

步驟 4：提供使用者適當主題和佇列動作的許可

您應該使用 AWS Identity and Access Management (IAM) 只允許適當的使用者發佈至 Amazon SNS 主題，並從 Amazon SQS 佇列讀取/刪除訊息。如需控制 IAM 使用者之主題和佇列動作的詳細資訊，請參閱 [使用以身分為基礎的政策搭配 Amazon SNS](#)，以及位於 Amazon Simple Queue Service 開發人員指南中 [Amazon SQS 中的 Identity and Access Management](#)。

控制對主題或佇列的存取有兩種方式：

- [新增政策到 IAM 使用者或群組](#)。提供使用者主題或佇列的許可的最簡單方式，是建立群組和新增適當的政策到群組，然後新增使用者到該群組。從群組新增和移除使用者比起追蹤對個別使用者設定了哪些政策要來得簡單多。
- [新增政策到主題或佇列](#)。如果您想要將主題或佇列的許可授予另一個 AWS 帳戶，唯一的方法是新增政策做為 AWS 帳戶 您想要授予許可的委託人。

對於大多數案例，您應使用第一個方式 (透過新增或移除適當使用者到群組，來套用政策到群組和管理許可)。如果您需要提供許可給其他帳戶中的使用者，您應使用第二個方式。

新增政策到 IAM 使用者或群組

如果已新增下列政策到 IAM 使用者或群組，您要提供該使用者或該群組的成員許可，以對主題取得 MyTopic 執行 sns:Publish 動作。

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

```
    }
  ]
}
```

如果您已新增下列政策到 IAM 使用者或群組，您要提供該使用者或該群組的成員許可，以對佇列 MyQueue1 和 MyQueue2 執行 `sqs:ReceiveMessage` 和 `sqs:DeleteMessage` 動作。

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

新增政策到主題或佇列

下列範例政策顯示如何提供其他帳戶主題和佇列的許可。

Note

當您授予帳戶中資源的另一個 AWS 帳戶 存取權時，您也會授予具有該資源管理員層級存取權（萬用字元存取）的 IAM 使用者。其他帳戶中的所有其他 IAM 使用者都會自動拒絕存取您的資源。如果您想要提供該 AWS 帳戶 中特定 IAM 使用者對您資源的存取權，具有管理員層級權限的帳戶或 使用者必須將資源的許可委派給那些 IAM 使用者。如需跨帳戶委派的詳細資訊，請參閱[使用 IAM 指南](#)中的啟用跨帳戶存取權。

如果您已新增下列政策到帳戶 123456789012 中的主題「我的主題」，您要提供帳戶 111122223333 許可，以對該主題執行 `sns:Publish` 動作。

```
{
```

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

如果您已新增下列政策到帳戶 123456789012 中的佇列 MyQueue，您要提供帳戶 111122223333 許可，以對該佇列執行 `sqs:ReceiveMessage` 和 `sqs:DeleteMessage` 動作。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

步驟 5：測試主題的佇列訂閱

您可以透過發佈至主題並檢視主題傳送至佇列的訊息，來測試主題的佇列。

使用 Amazon SNS 主控台發佈至主題

1. 使用 AWS 帳戶 或 IAM 使用者的登入資料，並具有發佈至 主題的許可，請登入 <https://console.aws.amazon.com/sns/>，AWS Management Console 然後開啟位於 <https://console.aws.amazon.com/sns/> : //www.Amazon SNS 主控台。

2. 在導覽面板上，選擇主題並選擇 Publish to Topic (發佈至主題)。
3. 在 Subject (主旨) 方塊中，輸入主旨 (例如，**Testing publish to queue**)，在 Message (訊息) 方塊中，輸入一些文字 (例如，**Hello world!**)，並選擇 Publish Message (發佈訊息)。下列訊息顯示：您的訊息已成功發佈。

使用 Amazon SQS 主控台檢視來自主題的訊息

1. 使用 AWS 帳戶 或 IAM 使用者的登入資料，並具有檢視佇列中訊息的許可，請登入，AWS Management Console 然後開啟位於 <https://console.aws.amazon.com/sqs/> : //www.Amazon SQS 主控台。
2. 選擇已訂閱至主題的 queue (佇列)。
3. 選擇 Send and receive messages (傳送和接收訊息)，然後選擇 Poll for messages (訊息輪詢)。類型為 Notification (通知) 的訊息隨即出現。
4. 在 Body (內文) 欄位中，選擇 More Details (更多詳細資訊)。Message Details (更多詳細資訊) 方塊包含 JSON 文件，該文件包含您發佈至主題的主旨和訊息。訊息看起來類似以下 JSON 文件。

```
{
  "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. 選擇 Close (關閉)。您已成功發佈至主題，其會傳送通知訊息至佇列。

使用 將 Amazon SNS 自動化至 Amazon SQS 訊息 AWS CloudFormation

AWS CloudFormation 可讓您使用範本檔案，將資源集合 AWS 一起建立和設定為單一單位。本節含有可輕鬆部署發佈至佇列之主題的範例範本。範本為您執行所有設定步驟，建立兩個佇列，建立訂閱佇列

的主題，新增政策至佇列，以便主題可以傳送訊息至佇列，以及建立 IAM 使用者和群組，以控制對那些資源的存取。

如需使用 AWS CloudFormation 範本部署 AWS 資源的詳細資訊，請參閱AWS CloudFormation 《使用者指南》[中的入門](#)。

使用 AWS CloudFormation 範本在 中設定主題和佇列 AWS 帳戶

範例範本使用適當的許可，為一個 IAM 群組的成員，建立可將訊息傳送至兩個 Amazon SQS 佇列的 Amazon SNS 主題，以發佈至主題而另一個讀取來自佇列的訊息。範本也建立新增至各個群組的 IAM 使用者。

您可以將範本內容複製到檔案中。您也可以從 [AWS CloudFormation 範本頁面](#) 下載範本。在範本頁面上，選擇依 AWS 服務瀏覽範例範本，然後選擇 Amazon Simple Queue Service。

MySNSTopic 是設定為發佈兩個已訂閱的端點，其為兩個 Amazon SQS 佇列 (MyQueue1 和 MyQueue2)。MyPublishTopicGroup 是 IAM 群組，其成員具有使用 [Publish](#) (發佈) API 動作或 [sns-publish](#) 命令發佈至 MySNSTopic 的許可。範本建立 IAM 使用者 MyPublishUser 和 MyQueueUser，並提供他們登入設定檔和存取金鑰。使用此範本建立堆疊的使用者，指定登入設定檔的密碼做為輸入參數。範本建立兩個 IAM 使用者的存取金鑰，分別為 MyPublishUserKey 和 MyQueueUserKey。AddUserToMyPublishTopicGroup 將 MyPublishUser 新增至 MyPublishTopicGroup，以便使用者將擁有指派至群組的許可。

MyRDMessageQueueGroup 是 IAM 群組，其成員具有使用 [ReceiveMessage](#) 和 [DeleteMessage](#) API 動作讀取和刪除來自兩個 Amazon SQS 佇列之訊息的許可。AddUserToMyQueueGroup 將 MyQueueUser 新增至 MyRDMessageQueueGroup，以便使用者將擁有指派至群組的許可。MyQueuePolicy 指派 MySNSTopic 的許可，以發佈其通知至兩個佇列。

下列清單顯示 AWS CloudFormation 範本內容。

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

```

```
}  
}  
}
```

將 Amazon SNS 通知散發至 HTTPS 端點

您可使用 Amazon SNS 將通知訊息傳送至一個或多個 HTTP 或 HTTPS 端點。當您訂閱端點到主題時，您可以發佈通知到主題，而 Amazon SNS 會傳送 HTTP POST 請求以傳遞通知內容到訂閱的端點。當您訂閱端點時，您選擇 Amazon SNS 是否使用 HTTP 或 HTTPS 傳送 POST 請求到端點。如果您使用 HTTPS，則您可以善用下列項目在 Amazon SNS 中的支援：

- 伺服器名稱指示 (SNI) - 這可讓 Amazon SNS 支援需要 SNI 的 HTTPS 端點，例如需要多個憑證以代管多個網域的伺服器。如需 SNI 的詳細資訊，請參閱[伺服器名稱指示](#)。
- 基本和摘要存取身分驗證 - 這可讓您在 HTTP POST 請求的 HTTPS URL 中指定使用者名稱和密碼，例如 `https://user:password@domain.com` 或 `https://user@domain.com`。系統會透過使用 HTTPS 時建立的 SSL 連線對使用者名稱和密碼進行加密。只有網域名稱是以純文字傳送。如需有關基本和摘要存取身分驗證的詳細資訊，請參閱[RFC-2617](#)。

Important

Amazon SNS 目前不支援私有 HTTP (S) 端點。

HTTPS URL 只能從 Amazon SNS `GetSubscriptionAttributes` API 動作中檢索，適用於您已授予 API 存取權的委託人。

Note

用戶端服務必須能夠支援 HTTP/1.1 401 Unauthorized 標頭回應

請求包含已發佈至主題的主旨和訊息，同時包含有關 JSON 文件中通知的中繼資料。請求將看起來類似以下 HTTP POST 請求。如需 HTTP 標頭和 JSON 格式的請求內文的詳細資訊，請參閱[HTTP/HTTPS 標頭](#) 和 [HTTP/HTTPS 通知 JSON 格式](#)。

Note

Amazon SNS 會將所有 5XX 錯誤和 429（傳送的請求太多）錯誤視為可重試。這些錯誤受交付政策約束。所有其他錯誤都視為永久失敗，不會嘗試重試。

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" :
    "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfMpn2nk5HVS7Xn0n/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"
}
```

訂閱 HTTPS 端點至 Amazon SNS 主題

本主題說明如何將 HTTP/S 端點訂閱至 Amazon SNS 主題。

主題

- [步驟 1：確保您的端點已就緒可處理 Amazon SNS 訊息](#)

- [步驟 2：訂閱 HTTP/HTTPS 端點至 Amazon SNS 主題](#)
- [步驟 3：確認您的 Amazon SNS 訂閱](#)
- [步驟 4：選用 - 設定 Amazon SNS 訂閱的交付政策](#)
- [步驟 5：選用 - 授予使用者發佈至 Amazon SNS 主題的許可](#)
- [步驟 6：將 Amazon SNS 訊息傳送至 HTTP/HTTPS 端點](#)

步驟 1：確保您的端點已就緒可處理 Amazon SNS 訊息

在您訂閱您的 HTTP 或 HTTPS 端點到主題之前，您必須確保 HTTP 或 HTTPS 端點擁有處理 HTTP POST 請求 (Amazon SNS 用來傳送訂閱確認和通知訊息) 的能力。通常，這表示建立和部署處理來自 Amazon SNS 之 HTTP 請求的 Web 應用程式 (例如，如果您的端點主機執行 Linux 搭配 Apache 和 Tomcat)，則為 Java servlet。在您訂閱 HTTP 端點時，Amazon SNS 會傳送確認請求至該端點。您的端點必須在您建立訂閱時準備好接收和處理此請求，因為 Amazon SNS 會在那時候傳送此請求。Amazon SNS 不會傳送通知至端點，除非您已確認訂閱。一旦您確認訂閱，Amazon SNS 將在訂閱的主題上執行發佈動作時，傳送通知到端點。

設定您的端點以處理訂閱確認和通知訊息

1. 您的程式碼應會讀取 Amazon SNS 傳送至您端點的 HTTP POST 請求的 HTTP 標頭。您的程式碼應會尋找標頭欄位 `x-amz-sns-message-type`，此欄位告訴您 Amazon SNS 所傳送給您的訊息類型。透過查看標頭，您可以判定訊息類型，而無須剖析 HTTP 訊息內文。有兩種類型您需要處理：SubscriptionConfirmation 和 Notification。UnsubscribeConfirmation 訊息唯有從主題刪除訂閱時才使用。

如需有關 HTTP 標頭的詳細資訊，請參閱 [HTTP/HTTPS 標頭](#)。以下 HTTP POST 請求是訂閱確認訊息的範例。

```
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.\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+...",
"SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}

```

2. 您的程式碼應該剖析 HTTP POST 請求和內文類型文字/純文字，和內文中的 JSON 文件，以讀取組成 Amazon SNS 訊息的名稱值組。使用 JSON 剖析器，處理將逸出表示法的控制字元轉換回其 ASCII 字元值 (例如，將 \n 轉換為新行字元)。您可以使用現有的 JSON 剖析器，例如 [Jackson JSON Processor](#) 或者您自己撰寫。為傳送主旨和訊息欄位中的文字做為有效的 JSON，Amazon SNS 必須轉換一些控制字元為可包含在 JSON 文件中的逸出表示法。當您接收傳送至您端點的 POST 請求內文中的 JSON 文件時，如果您想要確切呈現發佈至主題的原始主旨和訊息，您必須將逸出的字元轉換回其原始字元值。如果您想要驗證通知的簽章，這很重要，因為簽章使用其原始形式的訊息和主旨做為要簽署之字串的一部分。
3. 您的程式碼應該驗證 Amazon SNS 所傳送之通知、訂閱確認或取消訂閱確認訊息的真偽。使用包含在 Amazon SNS 訊息中的資訊，您的端點可以重新建立簽章，以便您能夠透過比對簽章與 Amazon SNS 隨訊息傳送的簽章，來驗證訊息的內容。如需有關驗證訊息之簽章的詳細資訊，請參閱 [驗證 Amazon SNS 訊息的簽章](#)。
4. 根據標頭欄位 x-amz-sns-message-type 所指定的類型，您的程式碼應該讀取 HTTP 請求內文中所包含的 JSON 文件並處理訊息。以下是處理兩個主要類型的訊息的準則：

SubscriptionConfirmation

讀取 SubscribeURL 的值並造訪該 URL。若要確認訂閱並開始在端點接收通知，您必須造訪 SubscribeURL (例如，透過傳送 HTTP GET 請求至 URL)。請查看上一個步驟中的範例 HTTP 請求，以得知 SubscribeURL 看起來像什麼。如需有關 SubscriptionConfirmation 訊息之格式的詳細資訊，請參閱 [HTTP/HTTPS 訂閱確認 JSON 格式](#)。當您造訪 URL，您將得到類似下列 XML 文件的回應。文件會傳回 ConfirmSubscriptionResult 元素內端點的訂閱 ARN。

```
<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>
```

做為造訪 `SubscribeURL` 的替代方案，您可以使用 [ConfirmSubscription](#) 動作且 Token 設定為其 `SubscriptionConfirmation` 訊息中的對應值，來確認訂閱。如果您想要僅允許主題擁有者和訂閱擁有者可以取消訂閱端點，您使用 AWS 簽章呼叫 `ConfirmSubscription` 動作。

Notification

讀取 `Subject` 和 `Message` 的值來取得已發佈至主題的通知資訊。

如需有關 Notification 訊息之格式的詳細資訊，請參閱 [HTTP/HTTPS 標頭](#)。以下 HTTP POST 請求是傳送至端點 `example.com` 之通知訊息的範例。

```
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. 確保您的端點從具適當狀態碼的 Amazon SNS 對應至 HTTP POST 訊息。連線會在大約 15 秒內逾時。如果您的端點未在連線逾時之前回應，或您的端點傳回 200–4xx 範圍以外的狀態碼，Amazon SNS 會將訊息交付視為失敗嘗試。
6. 確保您的程式碼可以處理來自 Amazon SNS 的訊息傳遞重試。如果 Amazon SNS 未從您的端點收到成功回應，它會再次嘗試傳遞訊息。這會套用到所有訊息，包括訂閱確認訊息。根據預設，如果初始的訊息傳遞失敗，Amazon SNS 最多嘗試重試三次，各次失敗的重試之間的延遲設定在 20 秒。

Note

訊息請求會在大約 15 秒後逾時。這表示如果訊息傳遞失敗是因為逾時所導致，Amazon SNS 會在之前的傳遞重試後大約 35 秒進行重試。您可以為端點設定不同的傳遞政策。

Amazon SNS 使用 `x-amz-sns-message-id` 標頭欄位對發布到 Amazon SNS 主題的每則訊息進行唯一識別。透過比較您已處理傳入之訊息的 ID，您可以判斷訊息是否為重試的嘗試。

7. 如果您將訂閱 HTTPS 端點，請確保您的端點具有來自信任的憑證授權機構 (CA) 的伺服器憑證。Amazon SNS 將僅傳送訊息至具有 Amazon SNS 所信任之 CA 簽署的伺服器憑證的 HTTPS 端點。
8. 部署您所建立以接收 Amazon SNS 訊息的程式碼。當您訂閱端點時，端點必須就緒以至少接收訂閱確認訊息。

步驟 2：訂閱 HTTP/HTTPS 端點至 Amazon SNS 主題

若要透過主題傳送訊息至 HTTP 或 HTTPS 端點，您必須訂閱端點至 Amazon SNS 主題。您使用其 URL 來指定端點。若要訂閱主題，您可以使用 Amazon SNS 主控台、[sns-subscribe](#) 命令，或 [Subscribe](#) (訂閱) API 動作。在您開始前，請確保您擁有像要訂閱之端點的 URL，並且您的端點已如步驟 1 所述準備好接收確認和通知訊息。

使用 Amazon SNS 主控台訂閱 HTTP 或 HTTPS 端點至主題

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Subscriptions (訂閱)。
3. 選擇 Create subscription (建立訂閱)。
4. 在 Protocol (協定) 下拉式清單中，選擇 HTTP 或 HTTPS。
5. 在 Endpoint (端點) 方塊中，貼上您想要主題傳送訊息之目的地端點的 URL，然後選擇 Create subscription (建立訂閱)。
6. 會顯示確認訊息。選擇關閉。

您的新訂閱的 Subscription ID (訂閱 ID) 就會顯示 PendingConfirmation。在您確認訂閱時，Subscription ID (訂閱 ID) 將會顯示訂閱 ID。

步驟 3：確認您的 Amazon SNS 訂閱

若要確認您的 Amazon SNS 訂閱，請依照下列步驟確保您的端點可以成功接收訊息。此程序涉及設定您的端點來處理傳入的確認訊息、擷取確認 URL，以及確認訂閱。您可以自動或手動確認訂閱，視您的設定而定。

1. 訂閱 Amazon SNS 主題後，Amazon SNS 會傳送確認訊息到您的端點。此訊息包含SubscribeURL您必須用來確認訂閱的。
2. 必須設定您的端點，以接聽來自 Amazon SNS 的傳入訊息。確認訊息送達時，**SubscribeURL**請從訊息中擷取。
3. 擁有 後SubscribeURL，您可以透過下列兩種方式之一確認訂閱：

- 自動確認 – 您的端點可以透過傳送 HTTP GET 請求到 來自自動確認訂閱SubscribeURL。

此方法不需要手動介入。

- 手動確認 – 如果未設定自動確認，**SubscribeURL**請從確認訊息複製，並將其貼到瀏覽器的地址列。

這將手動確認訂閱。

4. 您也可以使用 Amazon SNS 主控台驗證訂閱狀態：

- a. 登入 [Amazon SNS 主控台](#)。
- b. 在導覽窗格中，選擇訂閱。

- c. 在清單中尋找您的訂閱。
 - 如果確認，SubscriptionArn則會顯示。
 - 如果仍未確認，則會顯示為 PendingConfirmation。

步驟 4：選用 - 設定 Amazon SNS 訂閱的交付政策

根據預設，如果初始的訊息傳遞失敗，Amazon SNS 最多嘗試重試三次，各次失敗的重試之間的延遲設定在 20 秒。如[步驟 1](#)中所討論，您的端點應具有可處理重試訊息的程式碼。透過設定主題或訂閱的傳遞政策，您可以控制 Amazon SNS 將重試失敗的訊息的頻率和間隔。您也可以可以在 DeliveryPolicy 中指定 HTTP/S 通知的內容類型。如需詳細資訊，請參閱[建立 HTTP/S 傳遞政策](#)。

步驟 5：選用 - 授予使用者發佈至 Amazon SNS 主題的許可

根據預設，主題擁有者擁有發佈至主題的許可。若要讓其他使用者或應用程式發佈至主題，您應該使用 AWS Identity and Access Management (IAM) 授予主題的發佈許可。如需將 Amazon SNS 動作的許可授予 IAM 使用者的詳細資訊，請參閱[使用以身分為基礎的政策搭配 Amazon SNS](#)。

控制對主題的存取有兩種方式：

- 新增政策到 IAM 使用者或群組。提供使用者主題的許可的最簡單方式，是建立群組和新增適當的政策到群組，然後新增使用者到該群組。從群組新增和移除使用者比起追蹤對個別使用者設定了哪些政策要來得簡單多。
- 新增政策到主題。如果您想要提供主題的許可給其他 AWS 帳戶，您可以做的唯一方式是透過加入已經有的政策，做為您想要提供許可的 AWS 帳戶 帳戶的委託人。

對於大多數案例，您應使用第一個方式 (透過新增或移除適當使用者到群組，來套用政策到群組和管理許可)。如果您需要提供許可給其他帳戶中的使用者，請使用第二個方式。

如果已新增下列政策到 IAM 使用者或群組，您要提供該使用者或該群組的成員許可，以對主題取得 MyTopic 執行 sns:Publish 動作。

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



```
    ]]  
  }
```

下列範例政策顯示如何提供其他帳戶主題的許可。

Note

當您授予帳戶中資源的另一個 AWS 帳戶 存取權時，您也會授予對該資源具有管理員層級存取（萬用字元存取）許可的 IAM 使用者。其他帳戶中的所有其他 IAM 使用者都會自動拒絕存取您的資源。如果您想要讓 中的特定 IAM 使用者 AWS 帳戶 存取您的資源，則具有管理員層級存取權的帳戶或 IAM 使用者必須將資源的許可委派給這些 IAM 使用者。如需跨帳戶委派的詳細資訊，請參閱[使用 IAM 指南](#)中的啟用跨帳戶存取權。

如果您已新增下列政策到帳戶 123456789012 中的主題「我的主題」，您要提供帳戶 111122223333 許可，以對該主題執行 `sns:Publish` 動作。

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

步驟 6：將 Amazon SNS 訊息傳送至 HTTP/HTTPS 端點

您可以透過發佈至主題，來傳送訊息到主題的訂閱。若要發佈主題，您可以使用 Amazon SNS 主控台、[sns-publish](#) CLI 命令或 [Publish](#) API。

如果您已遵照[步驟 1](#)，您在端點已部署的程式碼應處理通知。

使用 Amazon SNS 主控台發佈至主題

1. 使用具有發佈至主題許可的 AWS 帳戶或 IAM 使用者的登入資料，登入 AWS Management Console 並開啟位於 <https://console.aws.amazon.com/sns/> 的 Amazon SNS 主控台。

2. 在導覽面板上，選擇 Topics (主題)，然後選取主題。
3. 選擇 Publish message (發佈訊息) 按鈕。
4. 在 Subject (主題) 方塊中，輸入主題 (例如，**Testing publish to my endpoint**)。
5. 在 Message (訊息) 方塊中，輸入一些文字 (例如，**Hello world!**)，並選擇 Publish message (發佈訊息)。

下列訊息顯示：您的訊息已成功發佈。

驗證 Amazon SNS 訊息的簽章

Amazon SNS 使用訊息簽章來確認傳送至 HTTP 端點的訊息真實性。為了確保訊息完整性並防止詐騙，您必須先驗證簽章，才能處理任何 Amazon SNS 訊息。

何時應驗證 Amazon SNS 簽章？

在下列案例中，您應該驗證 Amazon SNS 訊息簽章：

- 當 Amazon SNS 傳送通知訊息到您的 HTTP(S) 端點時。
- 當 Amazon SNS 在 [Subscribe](#) 或 [Unsubscribe](#) API 呼叫後傳送確認訊息到您的端點時。

Amazon SNS 支援兩個簽章版本：

- SignatureVersion1 – 使用訊息的SHA1雜湊。
- SignatureVersion2 – 使用訊息的SHA256雜湊。這可提供更強大的安全性，並且是建議的選項。

若要正確驗證 SNS 訊息簽章，請遵循下列最佳實務：

- 一律使用 HTTPS 擷取簽署憑證，以防止未經授權的攔截攻擊。
- 檢查憑證是否由 Amazon SNS 發行。
- 確認憑證的信任鏈有效。
- 憑證應該來自 SNS 簽署的 URL。
- 未經驗證，請勿信任訊息中提供的任何憑證。
- 拒絕任何具有非預期的訊息TopicArn，以防止詐騙。
- Amazon SNS AWS SDKs 提供內建驗證邏輯，可降低實作錯誤的風險。

在 Amazon SNS 主題上設定訊息簽章版本

在 Amazon SNS 主題上設定訊息簽章版本，可讓您增強訊息驗證程序的安全性和相容性。

在 `SignatureVersion1` (SHA1) 和 `SignatureVersion2` (SHA256) 之間選取，以控制用於簽署訊息的雜湊演算法。Amazon SNS 主題預設為 `SignatureVersion1`。您可以使用 [SetTopicAttributes](#) API 動作來設定此設定。

使用下列範例，`SignatureVersion` 使用 設定主題屬性 AWS CLI：

```
aws sns set-topic-attributes \
  --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
  --attribute-name SignatureVersion \
  --attribute-value 2
```

使用 HTTP 查詢型請求時，驗證 Amazon SNS 訊息的簽章

在使用 HTTP 查詢型請求時，驗證 Amazon SNS 訊息的簽章可確保訊息的真偽和完整性。此程序會確認訊息來自 Amazon SNS，且未在傳輸期間遭到竄改。透過剖析訊息、建構正確的簽署字串，以及針對信任的公有金鑰驗證簽章，您可以保護系統免於詐騙和未經授權的訊息變更。

1. 在 Amazon SNS 傳送的 HTTP POST 請求內文中，從 JSON 文件擷取金鑰/值對。這些欄位是建構要簽署的字串的必要欄位。

- Message
- Subject (如果有)
- MessageId
- Timestamp
- TopicArn
- Type

例如：

```
MESSAGE_FILE="message.json"
FIELDS=("Message" "MessageId" "Subject" "Timestamp" "TopicArn" "Type")
```

 Note

如果任何欄位包含逸出字元（例如 \n），請將它們轉換為原始格式，以確保完全相符。

2. 在 Amazon SNS 訊息中尋找 SigningCertURL 欄位。此憑證包含驗證訊息簽章所需的公有金鑰。例如：

```
SIGNING_CERT_URL=$(jq -r '.SigningCertURL' "$MESSAGE_FILE")
```

3. 確定 SigningCertURL 來自信任的 AWS 網域（例如 <https://sns.us-east-1.amazonaws.com>）。基於安全理由，拒絕 AWS 網域外的任何 URLs。
4. 從提供的 URL 下載 X.509 憑證。例如：

```
curl -s "$SIGNING_CERT_URL" -o signing_cert.pem
```

5. 從下載的 X.509 憑證中擷取公有金鑰。公有金鑰可讓您解密訊息的簽章並驗證其完整性。例如：

```
openssl x509 -pubkey -noout -in signing_cert.pem > public_key.pem
```

6. 不同的訊息類型需要字串中不同的鍵值對才能簽署。識別訊息類型 (Type Amazon SNS 訊息中的欄位)，以決定要包含哪些鍵值對：
 - 通知訊息 – 包括 Message、MessageId、Subject (如果有) TopicArn、Timestamp 和 Type。
 - SubscriptionConfirmation 或 UnsubscribeConfirmation 訊息 – 包括 Message、MessageId、SubscribeURL、TimestampToken、TopicArn 和 Type。
7. Amazon SNS 需要字串簽署，以遵循嚴格的固定欄位順序進行驗證。只必須包含明確的必要欄位，無法新增額外的欄位。只有在訊息中存在時 Subject，才會包含選用欄位，例如，且必須出現在必要欄位順序所定義的確切位置。例如：

```
KeyNameOne\nValueOne\nKeyNameTwo\nValueTwo
```

 Important

請勿在字串結尾新增換行字元。

8. 依位元組排序順序排列鍵/值對（按鍵名稱的字母）。

9. 使用下列格式範例建構字串以簽署：

```
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
```

通知訊息範例：

```
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 範例：

```
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. 訊息中的 Signature 欄位以 Base64-encoded。您需要將其解碼，才能將其原始二進位格式與衍生的雜湊進行比較。例如：

```
SIGNATURE=$(jq -r '.Signature' "$MESSAGE_FILE")
echo "$SIGNATURE" | base64 -d > signature.bin
```

11. 使用 SignatureVersion 欄位來選取雜湊演算法：

- 對於 SignatureVersion1，請使用 SHA1（例如，-sha1）。
- 對於 SignatureVersion2，請使用 SHA256（例如 -sha256）。

12. 若要確認 Amazon SNS 訊息的真實性，請產生建構字串的雜湊，並使用公有金鑰驗證簽章。

```
openssl dgst -sha256 -verify public_key.pem -signature signature.bin <<<
"$STRING_TO_SIGN"
```

如果簽章有效，則輸出為 Verified OK。否則，輸出為 Verification Failure。

具有錯誤處理的範例指令碼

下列範例指令碼會自動執行驗證程序：

```
#!/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)
```

剖析 Amazon SNS 訊息格式

當 Amazon SNS 傳送訊息至 HTTP/HTTPS 端點時，它們同時包含 HTTP 標頭和 JSON 訊息內文。這些訊息遵循結構化格式，其中包含中繼資料，例如訊息類型、主題 ARN、時間戳記和數位簽章。透過正確剖析 Amazon SNS 訊息，您可以判斷訊息是訂閱確認、通知還是取消訂閱確認、擷取相關資料，以及使用簽章驗證真實性。

HTTP/HTTPS 標頭

當 Amazon SNS 傳送訂閱確認、通知或取消訂閱 HTTP/HTTPS 端點的確認訊息時，會傳送含有數個 Amazon SNS 特定標頭值的 POST 訊息。您可以將這些標頭值使用於識別訊息類型等任務，而無需剖析 JSON 訊息內文以讀取 Type 值。根據預設，Amazon SNS 會將所有通知傳送至 Content-Type 設定為 text/plain; charset=UTF-8 的 HTTP/S 端點。若要選擇文字/純文字 (預設值) 以外的 Content-Type，請參閱 [建立 HTTP/S 傳遞政策](#) 中的 headerContentType。

x-amz-sns-message-type

訊息的類型。可能的值為 SubscriptionConfirmation、Notification 和 UnsubscribeConfirmation。

x-amz-sns-message-id

全域唯一識別符 (UUID)，對於每個發布的訊息均為唯一。若是 Amazon SNS 在重試期間重送的通知，會使用原始訊息的訊息 ID。

x-amz-sns-topic-arn

發佈此訊息之主題的 Amazon 資源名稱 (ARN)。

x-amz-sns-subscription-arn

訂閱此端點的 ARN。

以下 HTTP POST 標頭是發至 HTTP 端點的 Notification 訊息的範例。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

HTTP/HTTPS 訂閱確認 JSON 格式

在您訂閱 HTTP/HTTPS 端點之後，Amazon SNS 會傳送訂閱確認訊息至 HTTP/HTTPS 端點。此訊息包含 `SubscribeURL` 值，您必須造訪才能確認訂閱 (或者，您可以使用具有 [ConfirmSubscription](#) 的 Token 值)。

Note

除非已確認訂閱，否則 Amazon SNS 不會傳送通知至此端點

訂閱確認訊息是 POST 訊息，訊息內文包含 JSON 文件及以下名稱值組。

Type

訊息的類型。若是訂閱確認，類型為 `SubscriptionConfirmation`。

MessageId

全域唯一識別符 (UUID)，對於每個發布的訊息均為唯一。若是 Amazon SNS 在重試期間重送的訊息，會使用原始訊息的訊息 ID。

Token

您可以與 [ConfirmSubscription](#) 動作搭配使用以確認訂閱。或者，您也可直接造訪 SubscribeURL。

TopicArn

此端點訂閱之主題的 Amazon Resource Name (ARN)。

Message

說明訊息的字串。若是訂閱確認，此字串看起來如下：

```
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

為了確認訂閱您必須造訪的 URL。或者，您可以改為使用 Token 和 [ConfirmSubscription](#) 動作以確認訂閱。

Timestamp

送出確認訂閱的時間 (GMT)。

SignatureVersion

Amazon SNS 簽章所使用的版本。

- 如果 SignatureVersion 為 1，則 Signature 為 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base-64 編碼 SHA1withRSA 簽章。
- 如果 SignatureVersion 為 2，則 Signature 為 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base-64 編碼 SHA256withRSA 簽章。

Signature

以 Base64 編碼 SHA1withRSA 或 Message、MessageId、Type、Timestamp 和 TopicArn 值的 SHA256withRSA 簽章。

SigningCertURL

用於簽署訊息的憑證的 URL。

以下 HTTP POST 訊息是發至 HTTP 端點的 SubscriptionConfirmation 訊息的範例。

```
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"
}
```

HTTP/HTTPS 通知 JSON 格式

當 Amazon SNS 傳送通知至已訂閱之 HTTP 或 HTTPS 端點時，已傳送至端點的 POST 訊息具有包含 JSON 文件及以下名稱值組的訊息內文。

Type

訊息的類型。若是通知，類型為 Notification。

MessageId

全域唯一識別符 (UUID)，對於每個發布的訊息均為唯一。若是 Amazon SNS 在重試期間重送的通知，會使用原始訊息的訊息 ID。

TopicArn

發佈此訊息之主題的 Amazon 資源名稱 (ARN)。

Subject

發布通知至主題時指定的 Subject 參數。

Note

這是選擇性的參數。如果未指定 Subject，則此名稱值組不會顯示在 JSON 文件中。

Message

發布通知至主題時指定的 Message 值。

Timestamp

發佈通知的時間 (GMT)。

SignatureVersion

Amazon SNS 簽章所使用的版本。

- 如果 SignatureVersion 為 1，則 Signature 為 Message、MessageId、Subject (如果存在)、Type、Timestamp 和 TopicArn 值的 Base-64 編碼 SHA1withRSA 簽章。
- 如果 SignatureVersion 為 2，則 Signature 為 Message、MessageId、Subject (如果存在)、Type、Timestamp 和 TopicArn 值的 Base-64 編碼 SHA256withRSA 簽章。

Signature

以 Base64 編碼 SHA1withRSA 或 Message、MessageId、Subject (如果存在)、Type、Timestamp 和 TopicArn 值的 SHA256withRSA 簽章。

SigningCertURL

用於簽署訊息的憑證的 URL。

UnsubscribeURL

您可以用來從此主題取消訂閱端點的 URL。如果您造訪此 URL，Amazon SNS 會取消訂閱端點，並停止傳送通知至此端點。

以下 HTTP POST 訊息是發至 HTTP 端點的 Notification 訊息的範例。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

HTTP/HTTPS 取消訂閱確認 JSON 格式

在從主題取消訂閱 HTTP/HTTPS 端點之後，Amazon SNS 會傳送取消訂閱確認訊息至端點。

取消訂閱確認訊息是 POST 訊息，訊息內文包含 JSON 文件及以下名稱值組。

Type

訊息的類型。若是取消訂閱確認，類型為 `UnsubscribeConfirmation`。

MessageId

全域唯一識別符 (UUID)，對於每個發布的訊息均為唯一。若是 Amazon SNS 在重試期間重送的訊息，會使用原始訊息的訊息 ID。

Token

您可以與 [ConfirmSubscription](#) 動作搭配使用，以重新確認訂閱。或者，您也可直接造訪 SubscribeURL。

TopicArn

此端點已取消訂閱之主題的 Amazon 資源名稱 (ARN)。

Message

說明訊息的字串。若是取消訂閱確認，此字串看起來如下：

```
You have chosen to deactivate subscription arn:aws:sns:us-east-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

為了重新確認訂閱您必須造訪的 URL。或者，您可以改為使用 Token 和 [ConfirmSubscription](#) 動作來重新確認訂閱。

Timestamp

送出確認取消訂閱的時間 (GMT)。

SignatureVersion

Amazon SNS 簽章所使用的版本。

- 如果 SignatureVersion 為 1，則 Signature 為 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base-64 編碼 SHA1withRSA 簽章。
- 如果 SignatureVersion 為 2，則 Signature 為 Message、MessageId、Type、Timestamp 和 TopicArn 值的 Base-64 編碼 SHA256withRSA 簽章。

Signature

以 Base64 編碼 SHA1withRSA 或 Message、MessageId、Type、Timestamp 和 TopicArn 值的 SHA256withRSA 簽章。

SigningCertURL

用於簽署訊息的憑證的 URL。

以下 HTTP POST 訊息是發至 HTTP 端點的 UnsubscribeConfirmation 訊息的範例。

```
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 交付政策 JSON 格式

如果您傳送請求至 SetSubscriptionAttributes 動作，並將 AttributeName 參數設定為 DeliveryPolicy 的值，AttributeValue 參數的值必須為有效的 JSON 物件。例如，下面的範例設定交付政策為總計 5 次重試。

```
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}}
...
```

AttributeValue 參數的值使用以下 JSON 格式。

```
{
  "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"
  }
}
```

如需有關 SetSubscriptionAttribute 動作的詳細資訊，請前往 Amazon Simple Notification Service API 參考 中的 [SetSubscriptionAttributes](#)。如需有關支援之 HTTP 內容類型標頭的詳細資訊，請參閱 [建立 HTTP/S 傳遞政策](#)。

SetTopicAttributes 交付政策 JSON 格式

如果您傳送請求至 SetTopicAttributes 動作，並將 AttributeName 參數設定為 DeliveryPolicy 的值，AttributeValue 參數的值必須為有效的 JSON 物件。例如，下面的範例設定交付政策為總計 5 次重試。

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

AttributeValue 參數的值使用以下 JSON 格式。

```
{
```

```
"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"
  }
}
```

如需有關 `SetTopicAttribute` 動作的詳細資訊，請前往 Amazon Simple Notification Service API 參考 中的 [SetTopicAttributes](#)。如需有關支援之 HTTP 內容類型標頭的詳細資訊，請參閱 [建立 HTTP/S 傳遞政策](#)。

將 Amazon SNS 事件散發至 AWS 事件分叉管道

對於事件封存和分析，Amazon SNS 現在建議使用其與 Amazon Data Firehose 的原生整合。您可以訂閱 Firehose 交付串流至 SNS 主題，這可讓您將通知傳送至封存和分析端點，例如 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon Redshift 資料表、Amazon OpenSearch Service (OpenSearch Service) 等。搭配使用 Amazon SNS 與 Firehose 交付串流是全受管且無程式碼的解決方案，不需要您使用 AWS Lambda 函數。如需詳細資訊，請參閱 [散發到 Firehose 交付串流](#)。

您可以使用 Amazon SNS 建置事件驅動型應用程式，這些應用程式會使用訂閱者服務自動執行工作，以回應發佈者服務所觸發的事件。這個架構模式可讓服務更具重複使用性、互通性和可擴展性。不過，可能需要投入更多勞力將事件分支到管道中處理，以滿足常見的事件處理需求，例如事件儲存、備份、搜尋、分析和重播。

為了加速事件驅動型應用程式的開發，您可以將事件處理管道訂閱 Amazon SNS 主題，由 AWS 事件分叉管道提供支援。AWS 事件分叉管道是一組開放原始碼 [巢狀應用程式](#)，以 [AWS 無伺服器應用程式](#)

[模型](#) (AWS SAM) 為基礎，您可以直接從[AWS 事件分叉管道套件](#)（選擇顯示建立自訂 IAM 角色或資源政策的應用程式）部署到 AWS 您的帳戶。

如需 AWS 事件分叉管道使用案例，請參閱 [部署和測試 Amazon SNS 事件分支管道範例應用程式](#)。

主題

- [AWS 事件分叉管道的運作方式](#)
- [部署 AWS 事件分叉管道](#)
- [部署和測試 Amazon SNS 事件分支管道範例應用程式](#)
- [訂閱 Amazon SNS 主題 AWS 的事件分叉管道](#)

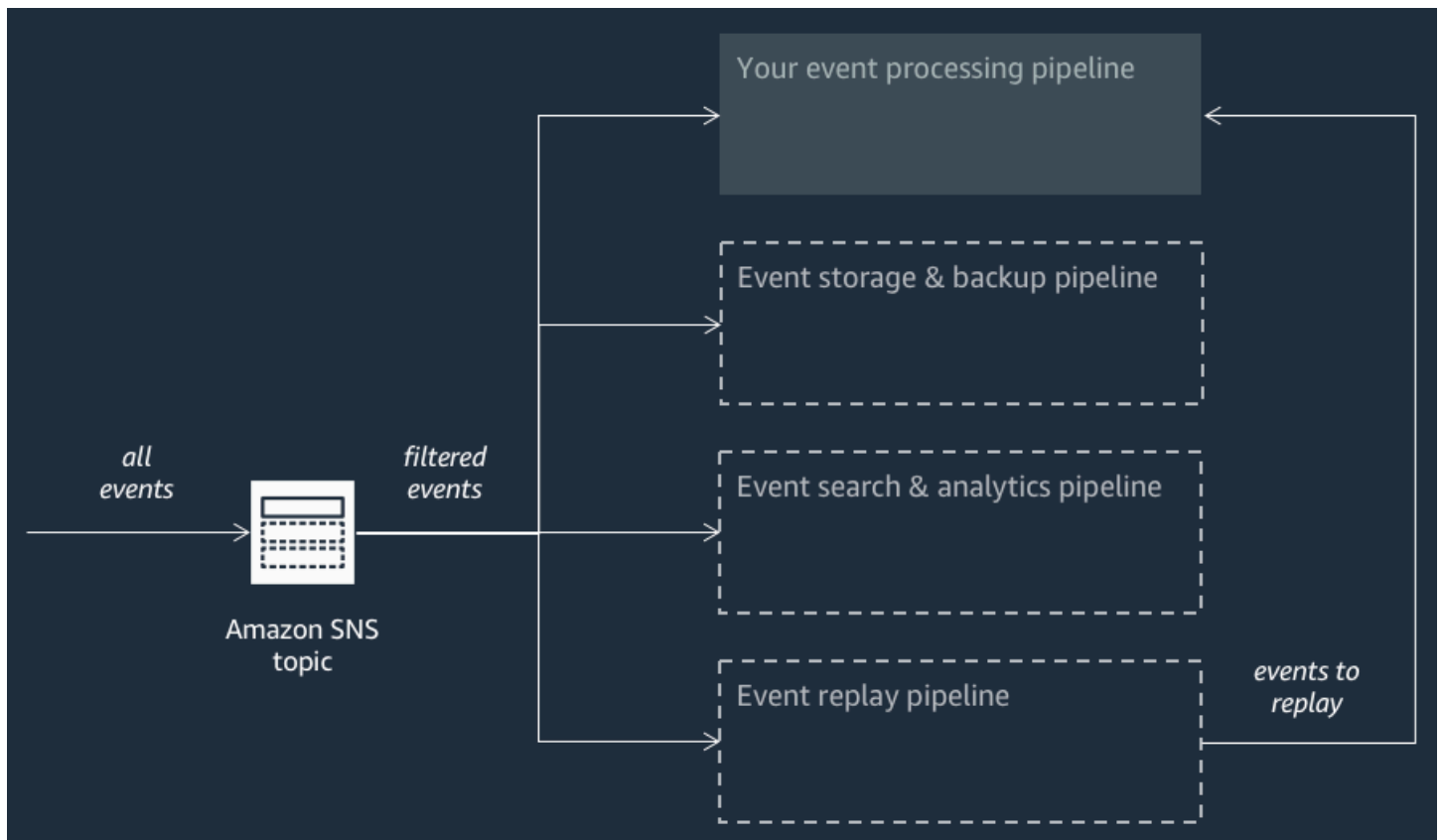
AWS 事件分叉管道的運作方式

AWS 事件分叉管道是一種無伺服器設計模式。不過，它也是一套以 SAM AWS 為基礎的巢狀無伺服器應用程式（您可以直接從 AWS Serverless Application Repository (AWS SAR) 部署到，AWS 帳戶以豐富您的事件驅動平台）。因應您的架構要求，您可以單獨部署這些巢狀應用程式。

主題

- [事件儲存和備份管道](#)
- [事件搜尋和分析管道](#)
- [事件重播管道](#)

下圖顯示由三個巢狀應用程式補充 AWS 的事件分叉管道應用程式。您可以根據您的架構需求，從 SAR 上的 AWS AWS 事件分叉管道套件獨立部署任何管道。



每個管道都訂閱同一個 Amazon SNS 主題，讓它本身可在事件發佈到主題時平行處理這些事件。每個管道各自獨立，而且可以設定自己的[訂閱篩選政策](#)。這可讓管道只處理它有感興趣的事件子集 (而不是發佈到該主題的所有事件)。

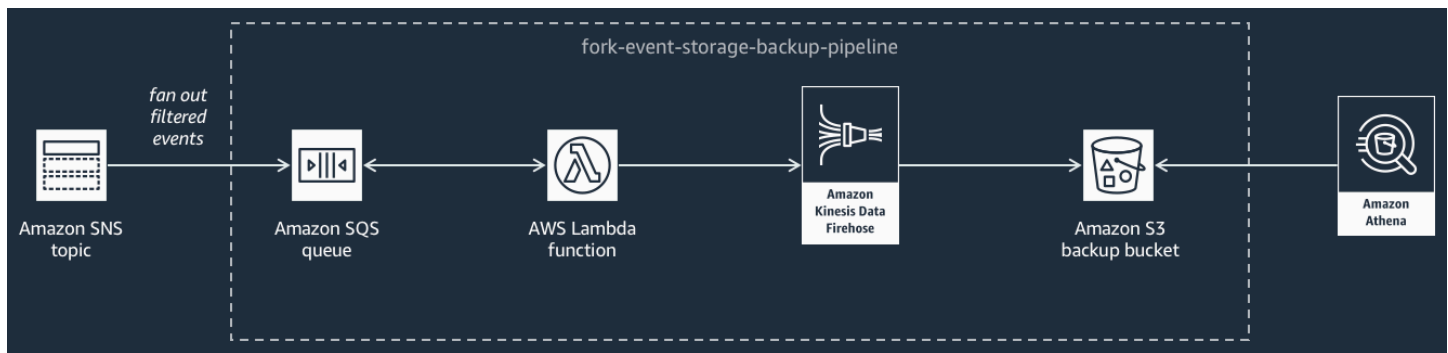
Note

由於您將三個 AWS Event Fork Pipelines 與一般事件處理管道一起放置 (可能已訂閱 Amazon SNS 主題)，因此您不需要變更目前訊息發佈者的任何部分，即可在現有工作負載中利用 AWS Event Fork Pipelines。

事件儲存和備份管道

下圖顯示[事件儲存和備份管道](#)。您可以讓此管道訂閱您的 Amazon SNS 主題，以自動備份流經您系統的事件。

此管道包含緩衝 Amazon SNS 主題所交付事件的 Amazon SQS 佇列、自動輪詢佇列中這些事件並將其推送至 Amazon Data Firehose 串流的 AWS Lambda 函數，以及長期備份串流載入事件的 Amazon S3 儲存貯體。Amazon SNS

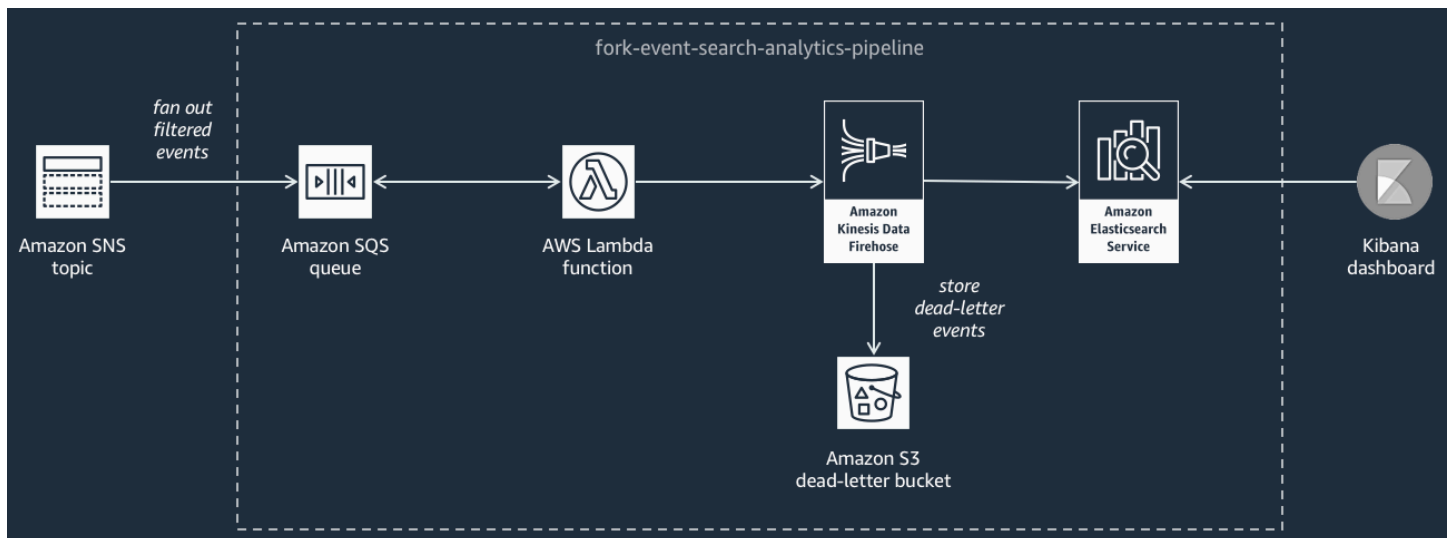


若要微調 Firehose 串流的行為，您可以將它設定為將事件載入到儲存貯體之前，先緩衝、轉換和壓縮事件。隨著事件載入，您可以透過 Amazon Athena 使用標準 SQL 查詢來查詢儲存貯體。您也可以將管道設定為重複使用現有的 Amazon S3 儲存貯體，或建立新的儲存貯體。

事件搜尋和分析管道

下圖顯示[事件搜尋和分析管道](#)。您可以讓此管道訂閱您的 Amazon SNS 主題，對搜尋網域中流經您系統的事件編製索引，然後對事件進行分析。

此管道包含緩衝 Amazon SNS 主題所交付事件的 Amazon SQS 佇列、輪詢佇列事件並將其推送至 Amazon Data Firehose 串流的 AWS Lambda 函數、編製 Firehose 串流載入事件索引的 Amazon OpenSearch Service 網域，以及存放無法在搜尋網域中編製索引之無效字母事件的 Amazon S3 儲存貯體。Amazon SNS



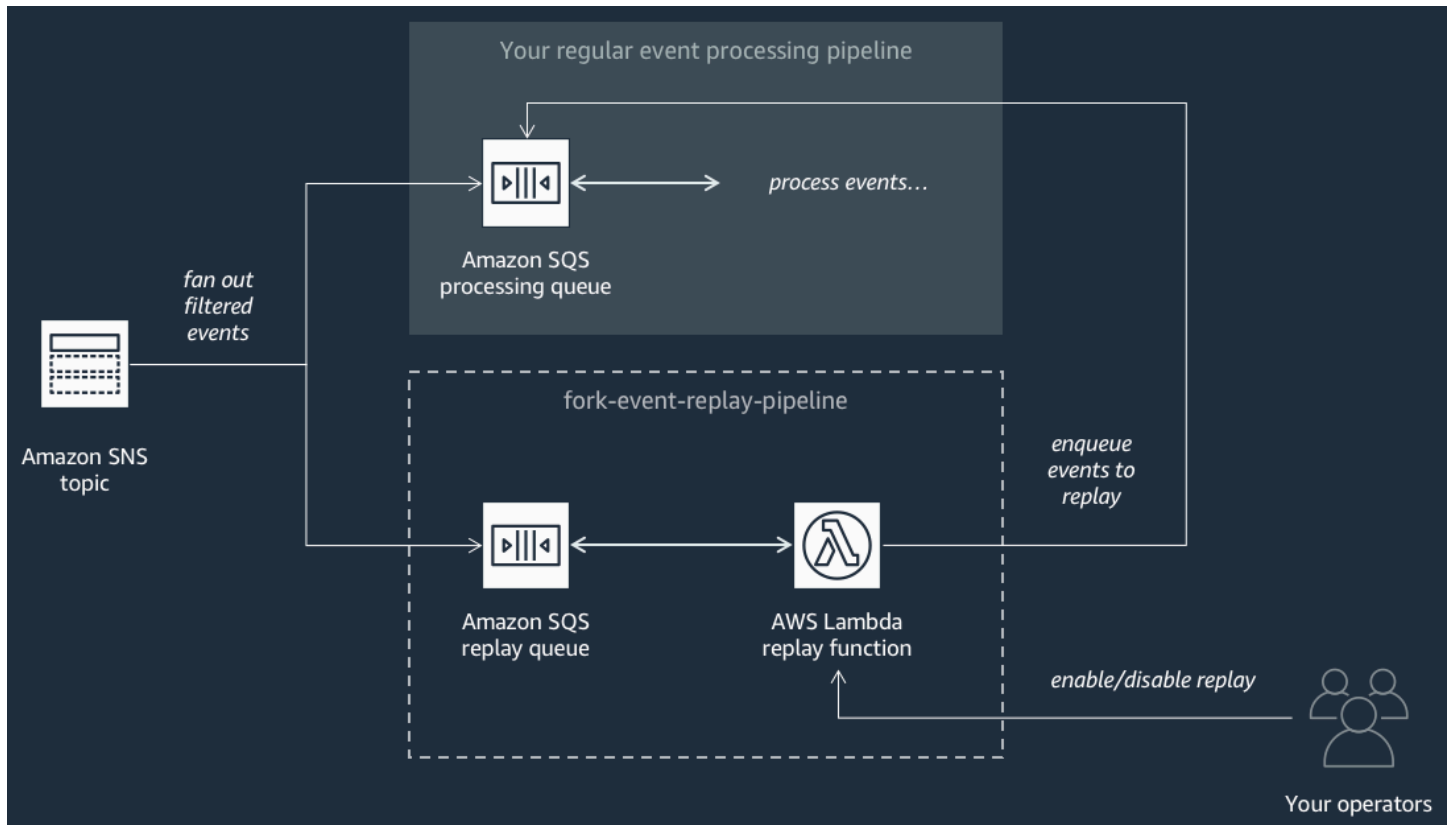
若要調整 Firehose 串流的事件緩衝、轉換和壓縮，您可以設定此管道。

您也可以設定管道是否應該重複使用中的現有 OpenSearch 網域，AWS 帳戶或為您建立新的網域。隨著搜尋網域中編製事件的索引，您可以使用 Kibana 對您的事件進行分析，並即時更新視覺化儀表板。

事件重播管道

下圖顯示[事件重播管道](#)。若要記錄您的系統在過去 14 天已處理的事件 (例如當您的平台需要從故障中復原時)，您可以讓此管道訂閱 Amazon SNS 主題，然後重新處理事件。

此管道包含緩衝 Amazon SNS 主題所交付事件的 Amazon SQS 佇列，以及輪詢佇列中的事件並將其重新驅動至您一般事件處理管道的 AWS Lambda 函數，該管道也會訂閱您的主題。Amazon SNS



Note

預設情況下，重播函數會停用，不會轉送您的事件。如果您需要重新處理事件，則必須啟用 Amazon SQS 重播佇列做為 AWS Lambda 重播函數的事件來源。

部署 AWS 事件分叉管道

[AWS 事件分叉管道套件](#) (選擇顯示建立自訂 IAM 角色或資源政策的應用程式) 可在 [中](#) 做為公有應用程式群組使用 AWS Serverless Application Repository，您可以在其中使用 [AWS Lambda 主控台](#) 手動部署和測試這些應用程式。如需使用 AWS Lambda 主控台部署管道的資訊，請參閱 [訂閱 Amazon SNS 主題 AWS 的事件分叉管道](#)。

在生產案例中，我們建議在整體應用程式的 AWS SAM 範本中嵌入 AWS 事件分叉管道。巢狀應用程式功能可讓您透過將資源新增至 [AWS::Serverless::Application](#) AWS SAM 範本、參考 AWS SAR ApplicationId 和巢狀應用程式的 SemanticVersion 來執行此操作。

例如，您可以將下列 YAML 程式碼片段新增至 AWS SAM 範本的 Resources 區段，以使用事件儲存和備份管道做為巢狀應用程式。

```
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
```

當您指定參數值時，您可以使用 AWS CloudFormation 內部函數來參考範本中的其他資源。例如，在上述 YAML 程式碼片段中，TopicArn 參數會參考 AWS SAM 範本中其他位置 MySNSTopic 定義的 [AWS::SNS::Topic](#) 資源。如需詳細資訊，請參閱 AWS CloudFormation 《使用者指南》中的 [內部函數參考](#)。

Note

SAR AWS 應用程式的 AWS Lambda 主控台頁面包含複製為 SAM 資源按鈕，可將巢狀 AWS SAR 應用程式所需的 YAML 複製到剪貼簿。

部署和測試 Amazon SNS 事件分支管道範例應用程式

為了加速事件驅動型應用程式的開發，您可以將事件處理管道訂閱 Amazon SNS 主題，由 AWS 事件分叉管道提供支援。AWS 事件分叉管道是一組開放原始碼 [巢狀應用程式](#)，以 [AWS 無伺服器應用程式模型](#) (AWS SAM) 為基礎，您可以直接從 [AWS 事件分叉管道套件](#)（選擇顯示建立自訂 IAM 角色或資源政策的應用程式）部署到 AWS 您的帳戶。如需詳細資訊，請參閱 [AWS 事件分叉管道的運作方式](#)。

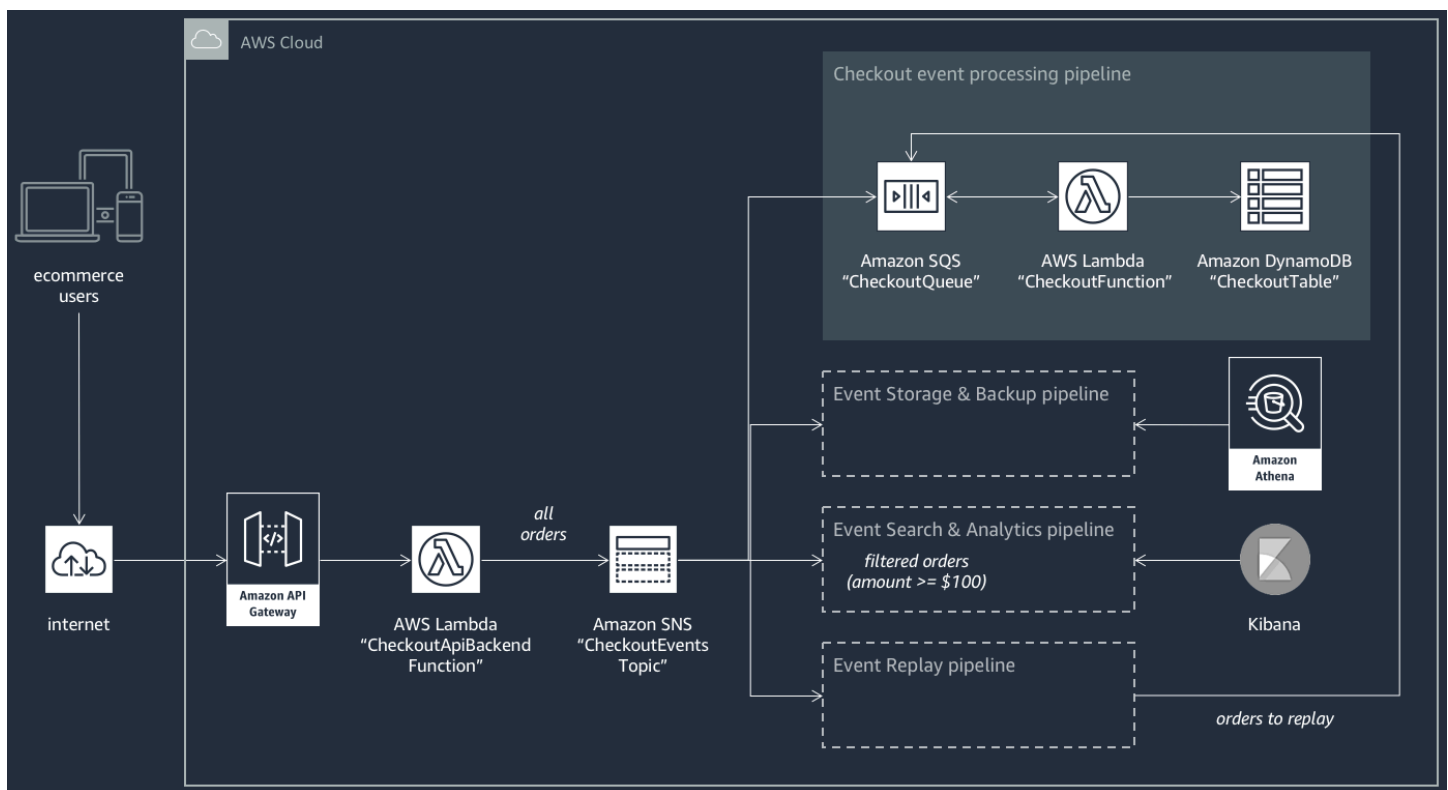
此頁面說明如何使用 AWS Management Console 部署和測試 AWS 事件分叉管道範例應用程式。

⚠ Important

若要避免在完成部署 AWS 事件分叉管道範例應用程式之後產生不必要的成本，請刪除其 AWS CloudFormation 堆疊。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的[刪除 AWS CloudFormation 主控台的堆疊](#)。

AWS 事件分叉管道使用案例範例

下列案例說明使用 AWS Event Fork Pipelines 的事件驅動型無伺服器電子商務應用程式。您可以在 [中](#) 使用此[範例電子商務應用程式](#)，AWS Serverless Application Repository 然後使用 AWS Lambda 主控台將其部署在 AWS 帳戶 中，您可以在其中進行測試，並在 GitHub 中檢查其原始程式碼。



此電子商務應用程式會透過 API Gateway 託管並由 AWS Lambda 函數支援的 RESTful API，接收買方的訂單CheckoutApiBackendFunction。此函數將所有收到的訂單發佈到名為 CheckoutEventsTopic 的 Amazon SNS 主題，此主題又進而將訂單散發到四個不同的管道。

第一個管道是一般結帳處理管道，由電子商務應用程式的擁有者所設計和實作。此管道具有CheckoutQueue緩衝所有已接收訂單的 Amazon SQS 佇列、名為的 AWS Lambda 函數CheckoutFunction，該函數會輪詢佇列以處理這些訂單，以及安全儲存所有已下訂訂單CheckoutTable的 DynamoDB 資料表。

套用 AWS 事件分叉管道

電子商務應用程式的元件處理核心商業邏輯。不過，電子商務應用程式擁有者還需要解決下列問題：

- 合規 靜態加密安全、壓縮的備份，並淨化敏感資訊
- 彈性 在履行政序中斷時重播最近的訂單
- 可搜尋性 對已提交的訂單進行分析並產生指標

應用程式擁有者可以訂閱 CheckoutEventsTopicAmazon SNS 主題 AWS 的事件分叉管道，而不是實作此事件處理邏輯

- [事件儲存和備份管道](#) 設定為轉換資料以移除信用卡詳細資訊、緩衝資料 60 秒、使用 GZIP 來壓縮資料，以及使用 Amazon S3 的預設客戶受管金鑰來加密資料。此金鑰由 管理 AWS，並由 AWS Key Management Service () 提供支援AWS KMS。

如需詳細資訊，請參閱 [《Amazon Data Firehose 開發人員指南》](#) 中的 [為您的目的地選擇 Amazon S3、Amazon Data Firehose 資料轉換和設定設定](#)。

- [事件搜尋和分析管道](#) 設定為索引重試持續時間 30 秒、使用儲存貯體來存放無法在搜尋網域中編製索引的訂單，以及使用篩選政策來限制可編製索引的訂單集。

如需詳細資訊，請參閱 [《Amazon Data Firehose 開發人員指南》](#) 中的 [為您的目的地選擇 OpenSearch Service](#)。

- 設定 [事件重播管道](#) 為使用電子商務應用程式擁有者所設計和實作的一般訂單處理管道的 Amazon SQS 佇列部分。

如需詳細資訊，請參閱 Amazon Simple Queue Service 開發人員指南中的 [佇列名稱與 URL](#)。

事件搜尋和分析管道的組態中設定下列 JSON 篩選政策。它只符合總金額為 100 USD 或更多金額的傳入訂單。如需詳細資訊，請參閱[Amazon SNS 訊息篩選](#)。

```
{
  "amount": [ { "numeric": [ ">=", 100 ] } ]
}
```

使用 AWS 事件分叉管道模式，電子商務應用程式擁有者可以避免通常遵循編碼事件處理之不區分邏輯的開發開銷。相反地，她可以直接從 將 AWS 事件分叉管道部署 AWS Serverless Application Repository 到她的 AWS 帳戶。

步驟 1：部署範例 Amazon SNS 應用程式

1. 登入 [AWS Lambda 主控台](#)。
2. 在導覽面板上，選擇 Functions (函數)，然後選擇 Create function (建立函數)。
3. 在 Create function (建立函數) 頁面上，執行下列動作：
 - a. 選擇瀏覽無伺服器應用程式存放庫、公有應用程式、顯示建立自訂 IAM 角色或資源政策的應用程式。
 - b. 搜尋 fork-example-ecommerce-checkout-api，然後選擇應用程式。
4. 在 fork-example-ecommerce-checkout-api 頁面上，執行下列動作：
 - a. 在 Application settings (應用程式設定) 區段中，輸入 Application name (應用程式名稱) (例如，fork-example-ecommerce-my-app)。

Note

- 為了在稍後輕鬆找到您的資源，請保留字首 fork-example-ecommerce。
- 對於每個部署，應用程式名稱必須是唯一的。如果您重複使用應用程式名稱，部署只會更新先前部署的堆疊（而不是建立新的 AWS CloudFormation 堆疊）。

- b. (選用) 輸入下列其中之一 LogLevel 設置來執行應用程序的 Lambda 函數：
 - DEBUG
 - ERROR
 - INFO (default)
 - WARNING
5. 選擇 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (我了解此應用程式會建立自訂 IAM 角色、資源政策及部署巢狀應用程式。)，然後在頁面底部選擇 Deploy (部署)。

在分叉範例電子商務的部署狀態 **my-app** 頁面上，Lambda 會顯示您的應用程式正在部署狀態。

在資源區段中，AWS CloudFormation 開始建立堆疊並顯示每個資源的 CREATE_IN_PROGRESS 狀態。程序完成時，AWS CloudFormation 會顯示 CREATE_COMPLETE 狀態。

Note

可能需要 20 到 30 分鐘才能部署完所有資源。

當部署完成時，Lambda 會顯示 Your application has been deployed (您的應用程式已經完成部署) 狀態。

步驟 2：執行 SNS 連結範例應用程式

1. 在 AWS Lambda 主控台的導覽面板上，選擇應用程式。
2. 在 Applications (應用程式) 頁面的搜尋欄位中，搜尋 `serverlessrepo-fork-example-ecommerce-my-app`，然後搜尋應用程式。
3. 在 Resources (資源) 區段中，執行下列動作：
 - a. 若要尋找類型為 ApiGateway RestApi 的資源，請依 Type (類型) 排序資源 (例如 ServerlessRestApi)，然後展開資源。
 - b. 將會顯示兩個巢狀資源，類型為 ApiGateway Deployment 和 ApiGateway Stage。
 - c. 複製 Prod API endpoint 連結，並附加 `/checkout`，例如：

```
https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

4. 將下列 JSON 複製到名為 `test_event.json` 的檔案。

```
{
  "id": 15311,
  "date": "2019-03-25T23:41:11-08:00",
  "status": "confirmed",
  "customer": {
    "id": 65144,
    "quantity": 2,
    "price": 25.00,
    "subtotal": 50.00
  }
}
```

5. 若要將 HTTPS 請求傳送到您的 API 端點，請執行 `curl` 命令來傳遞範例事件承載當做輸入，例如：


```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

API 會傳回以下空的回應，表示成功執行：

```
{ }
```

步驟 3：驗證 Amazon SNS 應用程式和管道效能

步驟 1：驗證範例結帳管道的執行

1. 登入 [Amazon DynamoDB 主控台](#)。
2. 在導覽面板上，選擇 Tables (資料表)。
3. 搜尋 serverlessrepo-fork-example 並選擇 CheckoutTable。
4. 在表格詳細資訊頁面，選擇 Items (項目)，然後選擇建立的項目。

將會顯示存放的屬性。

步驟 2：驗證事件儲存和備份管道的執行

1. 登入 [Amazon S3 主控台](#)。
2. 在導覽面板上，選擇 Buckets (儲存貯體)。
3. 搜尋 serverlessrepo-fork-example，然後選擇 CheckoutBucket。
4. 瀏覽目錄階層，直到您找到副檔名為 .gz 的檔案為止。
5. 若要下載檔案，請選擇 Actions (動作)、Open (開啟)。
6. 基於合規理由，管道已設定 Lambda 函數來淨化信用卡資訊。

若要驗證存放的 JSON 承載不包含任何信用卡資訊，請解壓縮檔案。

步驟 3：驗證事件搜尋和分析管道的執行

1. 登入 [OpenSearch Service 主控台](#)。
2. 在導覽面板的 My domains (我的網域) 下，選擇以 serverl-analyt 為字首的網域。
3. 管道已設定 Amazon SNS 訂閱篩選政策來設定數值相符條件。

若要驗證事件編製索引是因為它參考的訂單的值高於 100 USD，請在 `serverlessrepo-abcdefgh1ijk` 頁面上選擇 Indices (索引)、`checkout_events`。

步驟 4：驗證事件重播管道的執行

1. 請登入 [Amazon SQS 主控台](#)。
2. 在佇列清單中，搜尋 `serverlessrepo-fork-example` 並選擇 `ReplayQueue`。
3. 選擇傳送及接收訊息。
4. 在 Send and receive messages in `fork-example-ecommerce-my-app...ReplayP-ReplayQueue-123ABCD4E5F6` (`fork-example-ecommerce-my-app...ReplayP-ReplayQueue-123ABCD4E5F6` 中傳送和接收訊息) 對話方塊中，選擇 Poll for messages (輪詢訊息)。
5. 若要驗證事件已排入佇列中，請在佇列中出現的訊息旁邊選擇 More Details (更多詳細資訊)。

步驟 4：模擬問題並重播事件以進行復原

步驟 1：啟用模擬問題並傳送第二個 API 請求

1. 登入 [AWS Lambda 主控台](#)。
2. 在導覽面板上，選擇 Functions (函數)。
3. 搜尋 `serverlessrepo-fork-example` 並選擇 `CheckoutFunction`。
4. 在 `fork-example-ecommerce-my-app-CheckoutFunction-ABCDEF...` 頁面的 Environment variables (環境變數) 區段中，將 `BUG_ENABLED` 變數設定為 `true`，然後選擇 Save (儲存)。
5. 將下列 JSON 複製到名為 `test_event_2.json` 的檔案。

```
{
  "id": 9917,
  "date": "2019-03-26T21:11:10-08:00",
  "status": "confirmed",
  "customer": {
    "id": 56999,
    "quantity": 1,
    "price": 75.00,
    "subtotal": 75.00
  }
}
```

- 若要將 HTTPS 請求傳送到您的 API 端點，請執行 `curl` 命令來傳遞範例事件承載當做輸入，例如：

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
```

API 會傳回以下空的回應，表示成功執行：

```
{ }
```

步驟 2：驗證模擬資料損毀

- 登入 [Amazon DynamoDB 主控台](#)。
- 在導覽面板上，選擇 Tables (資料表)。
- 搜尋 `serverlessrepo-fork-example` 並選擇 CheckoutTable。
- 在表格詳細資訊頁面，選擇 Items (項目)，然後選擇建立的項目。

將會顯示存放的屬性，有些標記為 CORRUPTED! (毀損！)

步驟 3：停用模擬問題

- 登入 [AWS Lambda 主控台](#)。
- 在導覽面板上，選擇 Functions (函數)。
- 搜尋 `serverlessrepo-fork-example` 並選擇 CheckoutFunction。
- 在 `fork-example-ecommerce-my-app-CheckoutFunction-ABCDEF...` 頁面的 Environment variables (環境變數) 區段中，將 `BUG_ENABLED` 變數設定為 `false`，然後選擇 Save (儲存)。

步驟 4：啟用重播以從問題中復原

- 在 AWS Lambda 主控台的導覽面板上，選擇 函數。
- 搜尋 `serverlessrepo-fork-example` 並選擇 ReplayFunction。
- 展開 Designer (設計工具) 區段，選擇 SQS 圖磚，然後在 SQS 區段中選擇 Enabled (已啟用)。

Note

需要大約 1 分鐘，Amazon SQS 事件來源觸發才會啟用。

4. 選擇 Save (儲存)。
5. 若要查看復原的屬性，請返回 Amazon DynamoDB 主控台。
6. 若要停用重播，請返回 AWS Lambda 主控台並停用的 Amazon SQS 事件來源觸發 `ReplayFunction`。

訂閱 Amazon SNS 主題 AWS 的事件分叉管道

為了加速事件驅動型應用程式的開發，您可以將事件處理管道訂閱 Amazon SNS 主題，由 AWS 事件分叉管道提供支援。AWS 事件分叉管道是一組開放原始碼[巢狀應用程式](#)，以[AWS 無伺服器應用程式模型](#) (AWS SAM) 為基礎，您可以直接從[AWS 事件分叉管道套件](#)（選擇顯示建立自訂 IAM 角色或資源政策的應用程式）部署到 AWS 您的帳戶。如需詳細資訊，請參閱[AWS 事件分叉管道的運作方式](#)。

本節說明如何使用 AWS Management Console 部署管道，然後將 AWS 事件分叉管道訂閱 Amazon SNS 主題。開始之前，請[建立 Amazon SNS 主題](#)。

若要刪除構成管道的資源，請在 AWS Lambda 主控台的應用程式頁面上尋找管道，展開 SAM 範本區段，選擇 CloudFormation 堆疊，然後選擇其他動作、刪除堆疊。

將事件儲存和備份管道部署和訂閱至 Amazon SNS

對於事件封存和分析，Amazon SNS 現在建議使用其與 Amazon Data Firehose 的原生整合。您可以訂閱 Firehose 交付串流至 SNS 主題，這可讓您將通知傳送至封存和分析端點，例如 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon Redshift 資料表、Amazon OpenSearch Service (OpenSearch Service) 等。搭配使用 Amazon SNS 與 Firehose 交付串流是全受管且無程式碼的解決方案，不需要您使用 AWS Lambda 函數。如需詳細資訊，請參閱[散發到 Firehose 交付串流](#)。

本教學課程說明如何部署[事件儲存和備份管道](#)，並讓管道訂閱 Amazon SNS 主題。此程序會自動將與管道相關聯的 AWS SAM 範本轉換為 AWS CloudFormation 堆疊，然後將堆疊部署到您的 AWS 帳戶。此程序還會建立和設定一組資源（構成事件儲存和備份管道），包括下列項目：

- Amazon SQS 佇列

- Lambda 函數
- Firehose 交付串流
- Amazon S3 備份儲存貯體

如需使用 Amazon S3 儲存貯體做為目的地設定串流的詳細資訊，請參閱《Amazon Data Firehose API 參考[S3DestinationConfiguration](#)》中的。

如需有關轉換事件和設定事件緩衝、事件壓縮和事件加密的詳細資訊，請參閱 [《Amazon Data Firehose 開發人員指南》](#) 中的 [建立 Amazon Data Firehose 交付串流](#)。

如需有關篩選事件的詳細資訊，請參閱本指南中的 [Amazon SNS 訂閱篩選政策](#)。

1. 登入 [AWS Lambda 主控台](#)。
2. 在導覽面板上，選擇 Functions (函數)，然後選擇 Create function (建立函數)。
3. 在 Create function (建立函數) 頁面上，執行下列動作：
 - a. 選擇瀏覽無伺服器應用程式存放庫、公有應用程式、顯示建立自訂 IAM 角色或資源政策的應用程式。
 - b. 搜尋 fork-event-storage-backup-pipeline，然後選擇應用程式。
4. 在 fork-event-storage-backup-pipeline 頁面上，執行下列動作：
 - a. 在 Application settings (應用程式設定) 區段中，輸入 Application name (應用程式名稱) (例如，my-app-backup)。

 Note

- 對於每個部署，應用程式名稱必須是唯一的。如果您重複使用應用程式名稱，部署只會更新先前部署的堆疊（而不是建立新的 AWS CloudFormation 堆疊）。

- b. (選用) 針對 BucketArn，輸入要載入傳入事件的 Amazon S3 儲存貯體 ARN。如果您未輸入值，則會在 AWS 帳戶中建立新的 Amazon S3 儲存貯體。
- c. (選用) 對於 DataTransformationFunctionArn，輸入 Lambda 函數的 ARN，以透過此函數來轉換傳入的事件。如果您不輸入值，資料轉換會停用。
- d. (選用) 輸入下列其中之一 LogLevel 設置來執行應用程序的 Lambda 函數：
 - DEBUG
 - ERROR

- INFO (default)
 - WARNING
- e. 對於 TopicArn，輸入 Amazon SNS 主題的 ARN，表示分支管道的此執行個體將訂閱此主題。
- f. (選用) 對於 StreamBufferingIntervalInSeconds 和 StreamBufferingSizeInMBs，輸入值來設定如何緩衝傳入的事件。如果您不輸入任何值，則會使用 300 秒和 5 MB。
- g. (選用) 輸入以下其中一個 StreamCompressionFormat 設定來壓縮傳入的事件：
- GZIP
 - SNAPPY
 - UNCOMPRESSED (default)
 - ZIP
- h. (選用) 對於 StreamPrefix，輸入字串字首，以命名存放在 Amazon S3 備份儲存貯體中的檔案。如果您不輸入值，則不會使用字首。
- i. (選用) 對於 SubscriptionFilterPolicy，使用 JSON 格式輸入 Amazon SNS 訂閱篩選政策，以用於篩選傳入的事件。篩選政策決定哪些事件會在 OpenSearch Service 索引中編製索引。如果您不輸入值，則不會使用篩選 (所有事件都編製索引)。
- j. (可選) 針對 SubscriptionFilterPolicyScope，請輸入字串 MessageBody 或 MessageAttributes，啟用以承載或屬性為基礎的訊息篩選。
- k. 選擇 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (我了解此應用程式會建立自訂 IAM 角色、資源政策及部署巢狀應用程式。)，然後選擇 Deploy (部署)。

在部署狀態 **my-app** 頁面上，Lambda 會顯示您的應用程式正在部署狀態。

在資源區段中，AWS CloudFormation 開始建立堆疊並顯示每個資源的 CREATE_IN_PROGRESS 狀態。程序完成時，AWS CloudFormation 會顯示 CREATE_COMPLETE 狀態。

當部署完成時，Lambda 會顯示 Your application has been deployed (您的應用程式已經完成部署) 狀態。

發佈至 Amazon SNS 主題的訊息會自動儲存在事件儲存和備份管道佈建的 Amazon S3 備份儲存貯體中。

將事件搜尋和分析管道部署及訂閱至 Amazon SNS

對於事件封存和分析，Amazon SNS 現在建議使用其與 Amazon Data Firehose 的原生整合。您可以訂閱 Firehose 交付串流至 SNS 主題，這可讓您將通知傳送至封存和分析端點，例如 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon Redshift 資料表、Amazon OpenSearch Service (OpenSearch Service) 等。搭配使用 Amazon SNS 與 Firehose 交付串流是全受管且無程式碼的解決方案，不需要您使用 AWS Lambda 函數。如需詳細資訊，請參閱[散發到 Firehose 交付串流](#)。

本教學課程說明如何部署[事件搜尋和分析管道](#)，並讓管道訂閱 Amazon SNS 主題。此程序會自動將與管道相關聯的 AWS SAM 範本轉換為 AWS CloudFormation 堆疊，然後將堆疊部署到您的 AWS 帳戶。此程序還會建立和設定一組資源 (構成事件搜尋和分析管道)，包括下列項目：

- Amazon SQS 佇列
- Lambda 函數
- Firehose 交付串流
- Amazon OpenSearch Service 網域
- Amazon S3 無法投遞儲存貯體

如需以索引做為目的地設定串流的詳細資訊，請參閱《Amazon Data Firehose API 參考[ElasticsearchDestinationConfiguration](#)》中的。

如需有關轉換事件和設定事件緩衝、事件壓縮和事件加密的詳細資訊，請參閱[《Amazon Data Firehose 開發人員指南》](#)中的[建立 Amazon Data Firehose 交付串流](#)。

如需有關篩選事件的詳細資訊，請參閱本指南中的[Amazon SNS 訂閱篩選政策](#)。

1. 登入 [AWS Lambda 主控台](#)。
2. 在導覽面板上，選擇 Functions (函數)，然後選擇 Create function (建立函數)。
3. 在 Create function (建立函數) 頁面上，執行下列動作：
 - a. 選擇瀏覽無伺服器應用程式存放庫、公有應用程式、顯示建立自訂 IAM 角色或資源政策的應用程式。
 - b. 搜尋 fork-event-search-analytics-pipeline，然後選擇應用程式。
4. 在 fork-event-search-analytics-pipeline 頁面上，執行下列動作：

- a. 在 Application settings (應用程式設定) 區段中，輸入 Application name (應用程式名稱) (例如，my-app-search)。

 Note

對於每個部署，應用程式名稱必須是唯一的。如果您重複使用應用程式名稱，部署只會更新先前部署的堆疊（而不是建立新的 AWS CloudFormation 堆疊）。

- b. (選用) 對於 DataTransformationFunctionArn，輸入 Lambda 函數的 ARN，以使用此函數來轉換傳入的事件。如果您不輸入值，資料轉換會停用。
- c. (選用) 輸入下列其中之一 LogLevel 設置來執行應用程序的 Lambda 函數：
 - DEBUG
 - ERROR
 - INFO (default)
 - WARNING
- d. (選用) 對於 SearchDomainArn，輸入叢集所在 OpenSearch Service 網域的 ARN，此叢集將設定所需的運算和儲存功能。如果您不輸入值，將會以預設組態建立新網域。
- e. 對於 TopicArn，輸入 Amazon SNS 主題的 ARN，表示分支管道的此執行個體將訂閱此主題。
- f. 對於 SearchIndexName，輸入事件搜尋和分析的 OpenSearch Service 索引名稱。

 Note

索引名稱有下列配額：

- 不得包含大寫字母
- 不得包含下列字元：\ / * ? " < > | ` , #
- 開頭不得為下列字元：- + _
- 不得為下列字元：. . .
- 長度不得超過 80 個字元
- 長度不得超過 255 個位元組
- 不得包含冒號 (從 OpenSearch Service 7.0 開始)

- g. (選用) 輸入下列其中一項 OpenSearch Service 索引輪換期間的 `SearchIndexRotationPeriod` 設定：

- `NoRotation` (default)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

索引輪換會將時間戳記附加到索引名稱，方便看出舊資料已過期。

- h. 對於 `SearchTypeName`，輸入 OpenSearch Service 類型的名稱，以組織索引中的事件。

 Note

- OpenSearch Service 類型名稱可以包含任何字元 (null 位元組除外)，但開頭不得為 `_`。
- 若是 OpenSearch Service 6.x，每個索引僅能使用一個類型。如果您為已有其他類型的現有索引指定新類型，Firehose 會傳回執行時間錯誤。

- i. (選用) 對於 `StreamBufferingIntervalInSeconds` 和 `StreamBufferingSizeInMBs`，輸入值來設定如何緩衝傳入的事件。如果您不輸入任何值，則會使用 300 秒和 5 MB。

- j. (選用) 輸入以下其中一個 `StreamCompressionFormat` 設定來壓縮傳入的事件：

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (default)
- `ZIP`

- k. (選用) 對於 `StreamPrefix`，輸入字串字首，以命名存放在 Amazon S3 無效字母儲存貯體中的檔案。如果您不輸入值，則不會使用字首。

- l. (選用) 對於 `StreamRetryDurationInSeconds`，輸入 Firehose 無法在 OpenSearch Service 索引中為事件編製索引的情況的重試持續時間。如果您不輸入值，則會使用 300 秒。

- m. (選用) 對於 `SubscriptionFilterPolicy`，使用 JSON 格式輸入 Amazon SNS 訂閱篩選政策，以用於篩選傳入的事件。篩選政策決定哪些事件會在 OpenSearch Service 索引中編製索引。如果您不輸入值，則不會使用篩選 (所有事件都編製索引)。

- n. 選擇 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (我了解此應用程式會建立自訂 IAM 角色、資源政策及部署巢狀應用程式。), 然後選擇 Deploy (部署)。

在部署狀態 **my-app ##** 頁面上, Lambda 會顯示您的應用程式正在部署狀態。

在資源區段中, AWS CloudFormation 開始建立堆疊並顯示每個資源的 CREATE_IN_PROGRESS 狀態。程序完成時, AWS CloudFormation 會顯示 CREATE_COMPLETE 狀態。

當部署完成時, Lambda 會顯示 Your application has been deployed (您的應用程式已經完成部署) 狀態。

發佈到 Amazon SNS 主題的訊息會在由事件搜尋和分析管道佈建的 OpenSearch Service 索引中自動編製索引。如果管道無法為事件編製索引, 則會將它存放在 Amazon S3 無效字母儲存貯體中。

使用 Amazon SNS 整合部署事件重播管道

本教學課程說明如何部署 [事件重播管道](#), 並讓管道訂閱 Amazon SNS 主題。此程序會自動將與管道相關聯的 AWS SAM 範本轉換為 AWS CloudFormation 堆疊, 然後將堆疊部署到您的 AWS 帳戶。此程序還會建立和設定一組資源 (構成事件重播管道), 包括 Amazon SQS 佇列和 Lambda 函數。

如需有關篩選事件的詳細資訊, 請參閱本指南中的 [Amazon SNS 訂閱篩選政策](#)。

1. 登入 [AWS Lambda 主控台](#)。
2. 在導覽面板上, 選擇 Functions (函數), 然後選擇 Create function (建立函數)。
3. 在 Create function (建立函數) 頁面上, 執行下列動作:
 - a. 選擇瀏覽無伺服器應用程式存放庫、公有應用程式、顯示建立自訂 IAM 角色或資源政策的應用程式。
 - b. 搜尋 fork-event-replay-pipeline, 然後選擇應用程式。
4. 在 fork-event-replay-pipeline 頁面上, 執行下列動作:
 - a. 在 Application settings (應用程式設定) 區段中, 輸入 Application name (應用程式名稱) (例如, my-app-replay)。

 Note

對於每個部署，應用程式名稱必須是唯一的。如果您重複使用應用程式名稱，部署只會更新先前部署的堆疊（而不是建立新的 AWS CloudFormation 堆疊）。

- b. (選用) 輸入下列其中之一 LogLevel 設置來執行應用程序的 Lambda 函數：
- DEBUG
 - ERROR
 - INFO (default)
 - WARNING
- c. (選用) 對於 ReplayQueueRetentionPeriodInSeconds，輸入 Amazon SQS 重播佇列保留訊息的時間量 (以秒為單位)。如果您不輸入值，則會使用 1,209,600 秒 (14 天)。
- d. 對於 TopicArn，輸入 Amazon SNS 主題的 ARN，表示分支管道的此執行個體將訂閱此主題。
- e. 對於 DestinationQueueName，輸入 Amazon SQS 佇列的名稱，供 Lambda 重播函數將訊息轉送到此佇列。
- f. (選用) 對於 SubscriptionFilterPolicy，使用 JSON 格式輸入 Amazon SNS 訂閱篩選政策，以用於篩選傳入的事件。篩選政策決定緩衝哪些事件來重播。如果您不輸入值，則不會使用篩選 (緩衝所有事件來重播)。
- g. 選擇 I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications. (我了解此應用程式會建立自訂 IAM 角色、資源政策及部署巢狀應用程式。)，然後選擇 Deploy (部署)。

在部署狀態 **my-app ##** 頁面上，Lambda 會顯示您的應用程式正在部署狀態。

在資源區段中，AWS CloudFormation 開始建立堆疊並顯示每個資源的 CREATE_IN_PROGRESS 狀態。程序完成時，AWS CloudFormation 會顯示 CREATE_COMPLETE 狀態。

當部署完成時，Lambda 會顯示 Your application has been deployed (您的應用程式已經完成部署) 狀態。

發佈到 Amazon SNS 主題的訊息會在由事件重播管道佈建的 Amazon SQS 佇列中自動緩衝以重播。

Note

在預設情況下會停用重播。若要啟用重播，請導覽到 Lambda 主控台的函數頁面，展開 Designer (設計工具) 區段，選擇 SQS 圖磚，然後在 SQS 區段中，選擇 Enabled (已啟用)。

使用 Amazon EventBridge 排程器搭配 Amazon SNS

[Amazon EventBridge 排程器](#) 是無伺服器排程器，可讓您從單一受管的中央服務建立、執行及管理任務。使用 EventBridge 排程器，您可以使用週期性模式的 Cron 和 Rate 運算式來建立排程，或設定一次性呼叫。您可以設定彈性的交付時段、定義重試次數上限，以及設定失敗的 API 調用的最長保留時間。

本頁面說明如何使用 EventBridge 排程器，依照排程從 Amazon SNS 主題發佈訊息。

設定執行角色

當您建立新排程時，EventBridge 排程器必須具有代表您調用其目標 API 操作的權限。您可以使用執行角色，授與 EventBridge 排程器這些許可。排程執行角色所連接的許可政策會定義哪些是必要許可。許可是否為必要權限，取決於您希望 EventBridge 排程器調用的目標 API。

您在 EventBridge 排程器主控台建立排程時 (如以下程序所述)，EventBridge 排程器會根據您選取的目標自動設定執行角色。如果您想要使用其中一個 EventBridge 排程器 SDKs、AWS CLI 或 建立排程 AWS CloudFormation，您必須擁有現有的執行角色，授予 EventBridge 排程器調用目標所需的許可。如需手動設定排程執行角色的詳細資訊，請參閱《EventBridge 排程器使用者指南》中的 [設定執行角色](#)。

建立排程

使用主控台建立排程

1. 前往 <https://console.aws.amazon.com/scheduler/home> 開啟 Amazon EventBridge 排程器。
2. 在排程頁面上，選擇建立排程。
3. 在指定排程詳細資訊頁面的排程名稱和描述區段中，執行以下動作：
 - a. 在排程名稱中，輸入排程的名稱，例如：**MyTestSchedule**。
 - b. (選用) 在描述中，輸入對排程的描述，例如：**My first schedule**。

- c. 針對排程群組，從下拉式清單中選擇排程群組。如果您沒有群組，請選擇預設值。若要建立排程群組，請選擇建立自己的排程。

您可以使用排程群組，為不同群組的排程加上標籤。

4. • 選擇排程選項。

頻率	執行此作業...	
一次性排程 一次性排程只會在您指定的日期與時間調用目標一次。	針對日期和時間執行以下動作： • 依 YYYY/MM/DD 格式輸入有效日期。 • 依 hh:mm 格式輸入時間戳記 (24 小時)。 • 針對時區選擇時區。	
週期性排程 週期性排程會依您指定的頻率，使用 cron 或 Rate 運算式調用目標。	a. 在排程模式中，執行下列其中一項動作： • 若要使用 Cron 運算式定義排程，請選擇 Cron 排程，然後輸入 Cron 運算式。 • 若要使用 Rate 表達式定義排程，請選擇 Rate 排程，然後輸入 Rate 表達式。 如需 Cron 和 Rate 運算式的詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的 EventBridge 排程器上的排程類型。	

頻率	執行此作業...	
	b. 對於彈性時段，選擇關閉可關閉此選項，或者也能選擇其中一個預先定義的時間範圍。例如，如果您選擇 15 分鐘並設定週期性排程，每小時調用目標一次，則排程會在每小時一開始的 15 分鐘內執行。	

5. (選用) 如果您在上一個步驟中選擇週期性排程，請在時間範圍區段執行以下動作：
 - a. 針對時區選擇時區。
 - b. 對於開始日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。
 - c. 對於結束日期和時間，依 YYYY/MM/DD 格式輸入有效日期，接著依 24 小時的 hh:mm 格式指定時間戳記。
6. 選擇 Next (下一步)。
7. 在選取目標頁面上，選擇 EventBridge 排程器叫用的 AWS API 操作：
 - a. 選擇 Amazon SNS 發佈。
 - b. 在發佈區段中，選取 SNS 主題或選擇建立新 SNS 主題。
 - c. (選用) 輸入 JSON 承載。如果您未輸入承載，EventBridge 排程器會使用空白事件來調用函數。
8. 選擇 Next (下一步)。
9. 在設定頁面執行以下動作：
 - a. 若要開啟排程，請在排程狀態底下切換到啟用排程。
 - b. 若要設定排程的重試政策，請在重試政策和無效字母佇列 (DLQ) 底下執行以下動作：
 - 切換到重試。
 - 針對事件的最長存留期，輸入 EventBridge 排程器保留未處理事件的最大時數和分鐘數。
 - 時間最長可設為 24 小時。

- 針對重試次數上限，輸入目標傳回錯誤時，EventBridge 排程器重新嘗試執行排程的次數上限。

最大值為重試 185 次。

設定好重試政策後，如果排程無法調用其目標，EventBridge 排程器會重新執行排程。一旦設定此功能，您就必須設定排程的最長保留時間和重試次數。

- c. 選擇 EventBridge 排程器儲存未交付事件的位置。

無效字母佇列 (DLQ) 選項	執行此作業...
不儲存	選擇無。
將事件存放在您建立排程的相同 AWS 帳戶 中	a. 選擇在 中選取 Amazon SQS 佇列 AWS 帳戶 做為 DLQ。 b. 選擇 Amazon SQS 佇列的 Amazon Resource Name (ARN)。
將事件存放在與您要建立排程 AWS 帳戶 的位置不同的 中	a. 選擇指定其他 中的 Amazon SQS 佇列 AWS 帳戶 做為 DLQ。 b. 輸入 Amazon SQS 佇列的 Amazon Resource Name (ARN)。

- d. 若要使用由客戶管理的金鑰加密您的目標輸入，請在加密底下選擇自訂加密設定 (進階)。

如果選擇此選項，請輸入現有的 KMS 金鑰 ARN，或選擇建立 AWS KMS key，以導覽至 AWS KMS 控制台。如需 EventBridge 排程器如何加密靜態資料的詳細資訊，請參閱《Amazon EventBridge 排程器使用者指南》中的[靜態加密](#)。

- e. 若要讓 EventBridge 排程器為您建立新的執行角色，請選擇為此排程建立新角色。接著輸入角色名稱。如果您選擇此選項，EventBridge 排程器會將範本化目標所需的必要許可與角色連接。

10. 選擇 Next (下一步)。

11. 在檢閱和建立排程頁面上，檢閱排程的詳細資訊。在每個區段中選擇編輯，即可返回該步驟並編輯其詳細資訊。
12. 選擇建立排程。

您可以在排程頁面檢視新建立和現有的排程。在狀態欄底下，確認您的新排程狀態為已啟用。

相關資源

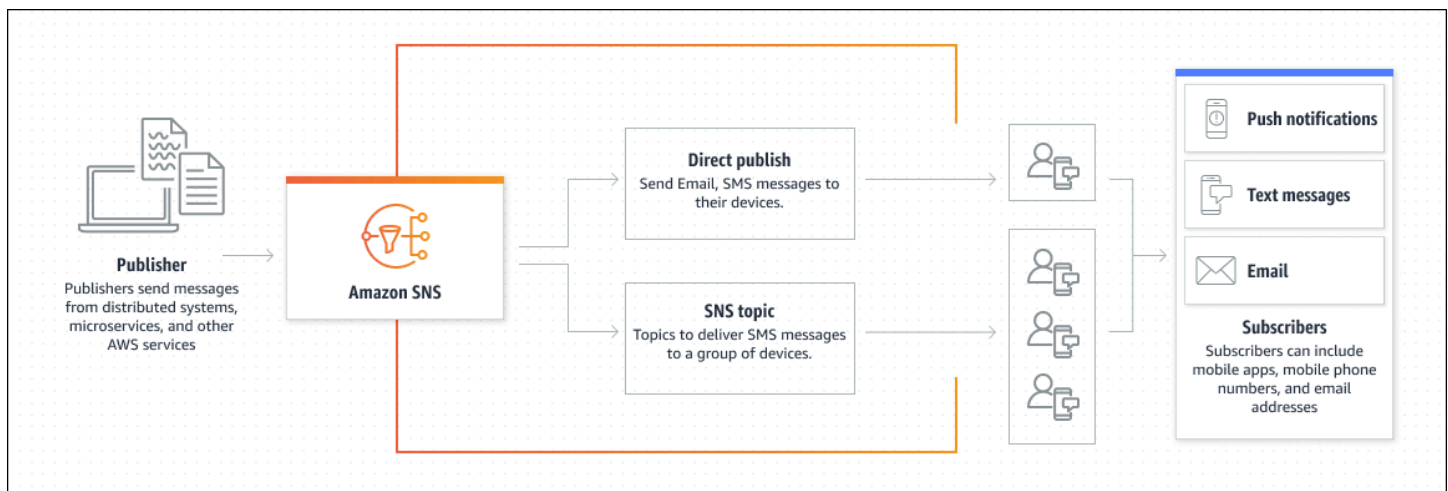
如需 EventBridge 排程器的詳細資訊，請參閱下列內容：

- [EventBridge 排程器使用者指南](#)
- [EventBridge 排程器 API 參考](#)
- [EventBridge 排程器定價](#)

將 Amazon SNS application-to-person 傳訊

Amazon SNS application-to-person(A2P) 訊息可讓您透過 SMS（簡訊服務）將通知和提醒直接傳送到客戶的行動裝置。使用此功能，您可以將推播通知傳送至行動應用程式、將簡訊傳送至行動電話號碼，以及將純文字電子郵件傳送至電子郵件地址。此外，您還可以使用主題來聯繫多個收件人，或直接將訊息發佈到個別行動端點以進行個人化通訊，藉此靈活地分配訊息。

本主題說明如何使用 Amazon SNS 與行動應用程式、行動電話號碼和電子郵件地址等訂閱者進行使用者通知。



使用 Amazon SNS 傳送行動簡訊

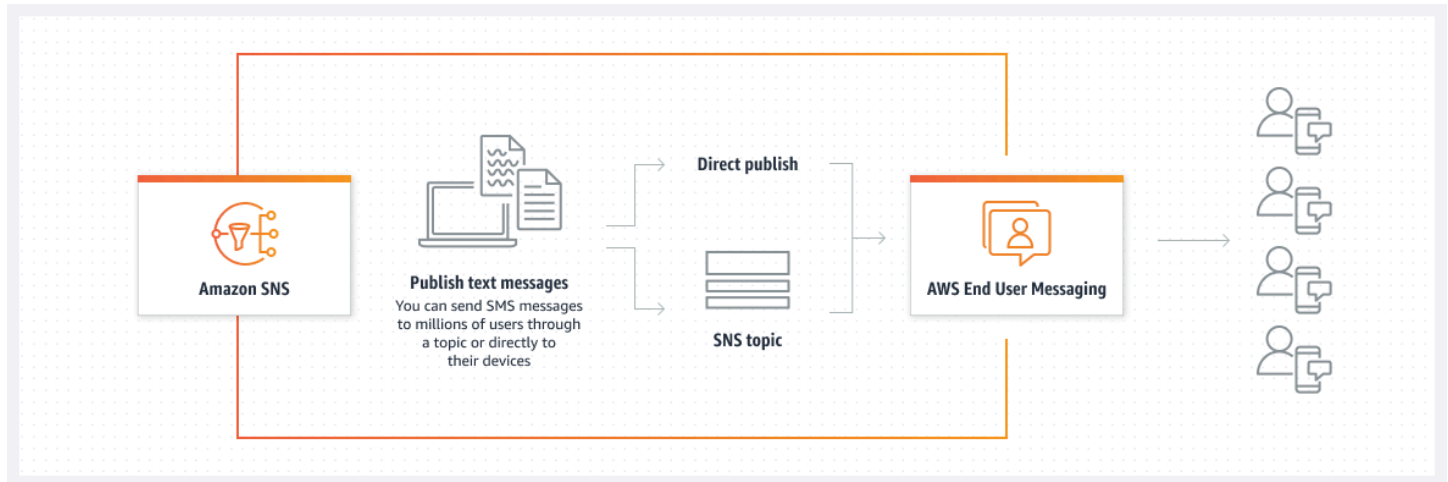
⚠ Important

Amazon SNS SMS 開發人員指南已更新。Amazon SNS 已與整合，[AWS End User Messaging SMS](#)以傳遞簡訊。本指南包含有關如何建立、設定和管理 Amazon SNS SMS 訊息的最新資訊。

Amazon SNS 行動簡訊 (SMS) 旨在協助將訊息傳遞至各種平台，例如支援 SMS 的 Web、行動和商業應用程式。使用者可以將訊息訂閱到主題，以簡化分發程序，進而將訊息傳送到一或多個電話號碼。

Amazon SNS 訊息是由傳遞 AWS End User Messaging SMS，可確保可靠的訊息傳輸。在 Amazon SNS APIs 中，您可以設定各種屬性，例如訊息類型（促銷或交易）、[每月花費限制](#)、[選擇退出清單](#)和[訊息交付最佳化](#)。

AWS End User Messaging SMS 會透過其全球簡訊供應網路，處理訊息傳輸到目的地電話號碼。它會管理路由、交付狀態，以及任何必要的區域法規合規。若要存取其他 SMS 功能，例如精細許可、電話集區、組態集、SMS 模擬器和國家/地區規則，請參閱 [AWS End User Messaging SMS 使用者指南](#)。



下列重要功能可協助您傳送可擴展且易於擴展的 Amazon SNS SMS 訊息：

[自訂訊息偏好設定](#)

根據您的預算和使用案例設定簡訊偏好設定，AWS 帳戶 為您的 自訂簡訊交付。例如，您可以選擇訊息是優先考慮成本效益還是可靠的交付。

[設定花費配額](#)

透過指定花費配額或個別訊息的交付和每月花費配額來量身打造您的簡訊交付 AWS 帳戶。依當地法律和法規（例如美國和加拿大）要求，SMS 收件人可以選擇 [不接收](#)，這表示他們選擇停止接收來自您的 SMS 訊息 AWS 帳戶。在收件人選擇不接收訊息後，您可以在有限制的情況下再次選擇加入電話號碼，以便繼續傳送訊息。

[全域傳送簡訊](#)

Amazon SNS 在多個區域中支援簡訊，可讓您傳送訊息到超過 240 個國家和地區。

Amazon SNS 如何傳遞我的簡訊？

當您請求 Amazon SNS 代表您傳送簡訊時，會使用 來傳送訊息 AWS End User Messaging SMS。Amazon SNS 與 之間的整合 AWS End User Messaging SMS 提供下列優點：

[IAM 政策](#)

您可以利用 IAM 和資源政策來控制和分配對其他 AWS 服務和區域的 SMS 資源的存取。

[AWS End User Messaging SMS 組態](#)

使用所有與原始伺服器 ID 相關的組態（建立、組態更新、佈建新的原始IDs、變更註冊範本）AWS End User Messaging SMS。

[AWS End User Messaging SMS 帳單](#)

不過，所有 SMS 帳單都會完成 AWS End User Messaging SMS。您可以合併 SMS 工作負載的 AWS 支出，同時在一個集中位置採購和管理 SMS 資源。

Amazon SNS SMS 入門

Important

Amazon SNS SMS 開發人員指南已更新。Amazon SNS 已與整合，[AWS End User Messaging SMS](#)以傳遞簡訊。本指南包含有關如何建立、設定和管理 Amazon SNS SMS 訊息的最新資訊。

本主題會引導您管理 SMS 沙盒，並設定 IAM 和資源型政策，以授予 Amazon SNS 存取和使用 AWS End User Messaging SMS APIs的必要許可。

先決條件

Amazon SNS 建議更新您的 IAM 政策，以包含下列動作，以確保對 Amazon SNS 資源的全面控制和可見性：

- [AmazonSNSFullAccess](#)
- [AmazonSNSReadOnly](#)

使用 Amazon SNS SMS 沙盒

新建立的 Amazon SNS SMS 帳戶會自動放入 SMS 沙盒中，透過降低詐騙和濫用的風險來確保 AWS 客戶和收件人的安全。此環境可做為安全的空間，用於測試和開發。在 SMS 沙盒內操作時，您可以存取所有 Amazon SNS 功能，但受到某些限制：

- SMS 訊息只能傳送到已驗證的目的地電話號碼。
- 您可以擁有多達 10 個已驗證的目的地電話號碼。
- 您只能在驗證後至少經過 24 小時或最後一次驗證嘗試之後刪除目的地電話號碼。

一旦您的帳戶從沙盒轉換出，這些限制就會移除，而且您可以將 SMS 訊息傳送給任何收件人。

首要步驟

新的 Amazon SNS SMS 帳戶會放置在 SMS 沙盒中。使用下列步驟在沙盒中建立和管理電話號碼、建立起始號碼和寄件者 IDs，以及註冊您的公司。

1. 將目的地電話號碼新增至 SMS 沙盒。如需新增、管理和移動電話號碼出 Amazon SNS SMS 沙盒的詳細資訊，請參閱 [在 Amazon SNS SMS 沙盒中新增和驗證電話號碼](#)。
2. 當您傳送簡訊給收件人時，建立收件人在其裝置上看到的起始身分。若要進一步了解起始身分，包括您可以使用的不同類型，請參閱 [Amazon SNS SMS 訊息的來源身分](#) 文件。
3. 註冊您的公司。有些國家/地區會要求您註冊公司的身分，才能購買電話號碼或寄件者 IDs，並檢閱您傳送給其國家/地區收件人的訊息。如需哪些國家/地區需要註冊的資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的 [使用 進行簡訊支援的國家/地區和區域 AWS End User Messaging SMS](#)。
4. 將您的訊息傳送到主題或行動電話。如需詳細資訊，請參閱 [使用 Amazon SNS 傳送簡訊](#)。

在 Amazon SNS SMS 沙盒中新增和驗證電話號碼

您必須先完成下列設定步驟，才能在簡訊沙盒 AWS 帳戶 中開始從 傳送簡訊。???這可確保您的帳戶已準備好接收簡訊，並正確驗證目的地電話號碼。

1. 建立 [起始 ID](#)。與 SMS 沙盒外部的帳戶類似，在傳送 SMS 訊息給某些國家或地區的收件人之前，需要一個起始 ID。
2. 在 SMS 沙盒內新增您要傳送訊息的目標電話號碼。
3. 驗證電話號碼，以確保目的地電話號碼在 SMS 訊息中有效。

新增並驗證目的地電話號碼

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台功能表中，選擇 [支援簡訊的區域](#)。
3. 在導覽窗格中，選擇 Text messaging (SMS) (簡訊 (SMS))。
4. 在沙盒目的地電話號碼區段中，選取新增電話號碼。
5. 在目的地詳細資訊下，提供以下資訊，然後選取新增電話號碼：
 - 目的地的國家/地區代碼和電話號碼。

- 您要傳送驗證訊息的語言。
6. 新增電話號碼後，Amazon SNS 會將 OTP 傳送至提供的目的地電話號碼。驗證需要此 OTP。
 7. 您會在提供的目的地電話號碼上收到標準簡訊形式的 OTP。
 - 如果您在 15 分鐘內未收到 OTP，請在 Amazon SNS 主控台中選取重新傳送驗證碼。
 - 您最多可以在 24 小時期間內重新傳送 OTP 五次。
 8. 收到 OTP 後，請在驗證碼方塊中輸入，然後選取驗證電話號碼。
 9. 檢查驗證狀態。
 - 成功驗證電話號碼後，電話號碼及其驗證狀態會顯示在沙盒目的地電話號碼區段中。
 - 如果狀態為待定，則驗證失敗。例如，如果您未正確輸入國家/地區代碼，就可能發生這種情況。
 - 您只能在自上次驗證嘗試後超過 24 小時或更長時間後刪除待定或已驗證的電話號碼。
 10. 如果您想要在其他區域使用相同的目的地電話號碼，請針對您要使用它的每個區域重複上述步驟。

未收到 OTP 文字的故障診斷

對可能阻止電話號碼接收 OTP 文字的常見問題進行故障診斷。

- Amazon SNS SMS 花費限制：如果您 AWS 帳戶的已超過傳送簡訊的花費限制，則在提高限制或解決帳單問題之前，可能無法傳送其他訊息，包括 OTP 文字。
- 未選擇接收簡訊通知的電話號碼：在某些國家或地區，收件人必須選擇接收來自短碼的簡訊，這通常用於 OTP 文字。如果收件人的電話號碼未選擇加入，他們將不會收到 OTP 文字。
- 電信業者限制或篩選：某些行動電信業者可能有限制或篩選機制，以防止傳遞特定類型的簡訊，包括 OTP 文字。這可能是由於電信業者實施的安全政策或反垃圾郵件措施。
- 無效或不正確的電話號碼：如果收件人提供的電話號碼不正確或無效，則不會傳送 OTP 文字。
- 網路問題：暫時性網路問題或中斷可能會阻止向收件人的電話傳遞簡訊，包括 OTP 文字。
- 延遲交付：在某些情況下，SMS 訊息可能會因網路擁塞或其他因素而導致交付延遲。OTP 文字最終可能會交付，但可能會延遲超過預期的時間範圍。
- 帳戶暫停或終止：如果您的發生問題 AWS 帳戶，例如未付款或違反 AWS 服務條款，Amazon SNS 訊息功能，包括 OTP 文字，可能會暫停或終止。

從 Amazon SNS SMS 沙盒刪除電話號碼

您可以從 [SMS 沙盒](#) 刪除待定和已驗證的目的地電話號碼。

⚠ Important

您只能在驗證電話號碼後 24 小時或最後一次驗證嘗試後 24 小時刪除電話號碼。

從簡訊沙盒安全執行程序中刪除目的地電話號碼

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台功能表中，選擇支援[簡訊的區域](#)，其中新增了目的地電話號碼。
3. 在導覽窗格中，選取簡訊 (SMS)。
4. 在行動簡訊 (SMS) 頁面上，導覽至沙盒目的地電話號碼區段。
5. 選擇您要刪除的特定電話號碼，然後選擇刪除電話號碼。
6. 若要確認您是否要刪除電話號碼，請輸入 **delete me**，然後選擇 Delete (刪除)。

請確定自您驗證或嘗試驗證目的地電話號碼後已經過 24 小時或更長時間，然後再繼續刪除。

7. 在您新增目的地電話號碼的每個區域中重複這些步驟，但不再計劃使用它。

移出 Amazon SNS SMS 沙盒

將 AWS 帳戶 移出 [SMS 沙盒](#)需要您先新增、驗證和測試目的地電話號碼。執行此操作後，請使用 建立案例 AWS 支援。

請求將 AWS 您的帳戶移出 SMS 沙盒

1. 驗證電話號碼
 - a. 當您的 AWS 帳戶 位於 SMS 沙盒中時，請開啟 [Amazon SNS 主控台](#)。
 - b. 在導覽窗格中，選擇行動通訊 (SMS) 下。
 - c. 在沙盒目的地電話號碼區段中，[新增並驗證](#)一或多個目的地電話號碼。此驗證可確保您可以成功傳送和接收訊息。
2. 測試簡訊發佈
 - 確認您能夠傳送和接收訊息到至少一個已驗證的電話號碼。如需如何發佈簡訊的詳細說明，請參閱 [使用 Amazon SNS 將簡訊發佈到行動電話](#)。
3. 啟動沙盒編輯

- 在 Amazon SNS 主控台的 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面，在 Account information (帳戶資訊) 下方選擇 Exit SMS sandbox (結束簡訊沙盒)。此動作會將您重新導向至 [Amazon Support Center](#)，並自動建立已選取服務配額增加選項的支援案例。

4. 填寫表單

- 在提高服務配額下的支援表單中，執行下列動作：
 - i. 選擇 SNS 簡訊做為服務。
 - ii. 提供您要傳送簡訊的網站 URL 或應用程式名稱。
 - iii. 指定您要傳送的訊息類型：一次性密碼、促銷或交易。
 - iv. 選擇您要傳送簡訊的 AWS 區域。
 - v. 列出您計劃傳送簡訊的國家或地區。
 - vi. 描述您的客戶如何選擇接收訊息。
 - vii. 包含您要使用的任何訊息範本。

5. 指定配額和區域

- 在 Requests (請求) 下，執行下列動作：
 - i. 選擇您要移動 AWS 區域的 AWS 帳戶。
 - ii. 選擇資源類型的一般限制。
 - iii. 選擇退出配額的 SMS 沙盒。
 - iv. (選用) 若要請求額外的增加或其他調整，請選擇新增另一個請求並指定必要的詳細資訊。
 - v. 針對新的配額值，輸入您請求的美元限制。

6. 其他詳細資訊

- a. 在案例描述中，提供與您的請求相關的任何其他詳細資訊。
- b. 在聯絡選項下，選擇您偏好的聯絡語言。

7. 提交請求

- 選擇提交以將您的請求傳送至 支援。

支援 團隊會在 24 小時內對您的請求提供初始回應。

為了避免使用我們的系統被用來傳送未經要求或惡意的內容，我們會仔細考慮每個請求。我們會盡量在 24 小時的期間內准許您的請求。不過，如果我們需要向您取得其他資訊，則可能需要更長的時間來處理您的請求。

如果您的使用案例不符合我們的政策，我們可能無法批准您的請求。

Amazon SNS SMS 訊息的來源身分

Important

Amazon SNS SMS 開發人員指南已更新。Amazon SNS 已與整合，[AWS End User Messaging SMS](#)以傳遞簡訊。本指南包含有關如何建立、設定和管理 Amazon SNS SMS 訊息的最新資訊。

SMS 訊息的來源身分是用來代表 SMS 訊息寄件者的識別符。您可以使用下列類型的原始身分，向收件人表明自己的身分：

起始號碼

識別簡訊寄件者電話號碼的數字字串。起始號碼有幾種類型，包括長碼（通常為 10 或多個數字的標準電話號碼）、10 位數長碼 (10DLC)、免付費電話號碼 (TFN) 和短碼（包含 4 到 7 位數的電話號碼）。

當地法律要求使用寄件者 ID 而不是來源號碼。當您使用來源號碼傳送簡訊時，收件者的裝置會將寄件號碼顯示為寄件者的電話號碼。您可以依使用案例指定不同的來源號碼。

如需詳細資訊，請參閱AWS End User Messaging SMS 《使用者指南》中的[電話號碼](#)。

Tip

若要檢視 AWS 帳戶中所有現有起始號碼的清單，請在 [Amazon SNS 主控台](#) 的導覽窗格中，選擇起始號碼。

寄件者 IDs

識別簡訊寄件者的字母名稱。當您使用寄件者 ID 傳送簡訊，且收件人位於支援寄件者 ID 身分驗證的區域時，您的寄件者 ID 會顯示在收件人的裝置上，而不是您的電話號碼。寄件者 ID 提供給簡訊收件人有關寄件者的資訊，比電話號碼、長碼或短碼提供的資訊更多。

世界各地許多國家和區域都支援寄件者 ID。在某些地方，如果您是一家將簡訊傳送給個別客戶的公司，則必須使用事先向監管機構或產業團體註冊的寄件者 ID。如需支援或需要寄件者 IDs 的國家/地區和區域的完整清單，請參閱 AWS End User Messaging SMS 《使用者指南》中的 [使用 傳送簡訊的支援國家/地區和區域 AWS End User Messaging SMS](#)。

使用寄件者 ID 無需額外費用。不過，寄件者 ID 身分驗證的支援和要求因國家/地區而異。有些主要市場 (包括加拿大、中國和美國) 不支援寄件者 ID。在某些地方，如果您是一家將簡訊傳送給個別客戶的公司，則必須使用事先向監管機構或產業團體註冊的寄件者 ID。

如需詳細資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的 [寄件者 IDs](#)。

在 Amazon SNS 中設定簡訊

Important

Amazon SNS SMS 開發人員指南已更新。Amazon SNS 已與整合，[AWS End User Messaging SMS](#) 以傳遞簡訊。本指南包含如何建立、設定和管理 Amazon SNS SMS 訊息的最新資訊。

您可以使用 Amazon SNS SMS 中的組態來設定 SMS 偏好設定以符合您的需求，例如調整支出配額和設定交付狀態記錄。本主題也提供如何使用 Amazon SNS 主控台和 AWS SDK 將 SMS 訊息發佈至主題的詳細資訊、有效處理配額，以及擷取 SMS 活動的詳細統計資料。

使用 Amazon SNS 傳送簡訊

本節說明如何使用 Amazon SNS 傳送 SMS 訊息，包括發佈至主題、訂閱電話號碼至主題、設定訊息屬性，以及直接發佈至行動電話。

將簡訊發佈至 Amazon SNS 主題

您可以透過訂閱電話號碼到 Amazon SNS 主題，一次發佈單一簡訊至多組電話號碼。SNS 主題是您可以新增訂閱者然後發佈訊息至所有那些訂閱者的通訊通道。訂閱者會收到發佈至主題的所有訊息，直到您取消訂閱，或訂閱者選擇不接收來自您 AWS 帳戶的簡訊為止。

使用 AWS 主控台傳送訊息至主題

若要建立主題

如果您尚未擁有想要傳送簡訊的主題，請完成下列步驟。

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台功能表中，選擇 [支援簡訊的區域](#)。
3. 在導覽窗格中，選擇 Topics (主題)。
4. 在 Topics (主題) 頁面上，選擇 Create topic (建立主題)。
5. 在 Create topic (建立主題) 頁面上，於 Details (詳細資料) 下方，執行下列動作：
 - a. 針對 Type (類型)，選擇 Standard (標準)。
 - b. 針對 Name (名稱)，輸入主題名稱。
 - c. (選用) 對於 Display name (顯示名稱)，輸入簡訊的自訂字首。傳送訊息至主題時，Amazon SNS 會將顯示名稱置於前面，後面加上右尖括號 (>) 和空格。顯示名稱不區分大小寫，Amazon SNS 會將顯示名稱轉換為大寫字元。例如，如果主題的顯示名稱為 MyTopic，而訊息為 Hello World!，則訊息將顯示為：

```
MYTOPIC> Hello World!
```

6. 請選擇建立主題。主題的名稱和 Amazon 資源名稱 (ARN) 會顯示在主題頁面上。

建立簡訊訂閱

訂閱可讓您只要發佈一次訊息到主題，就能傳送簡訊至多位收件人。

Note

當您開始使用 Amazon SNS 傳送簡訊時，AWS 您的帳戶位於簡訊沙盒中。簡訊沙盒為您提供了一個安全的環境，讓您嘗試 Amazon SNS 功能，而不會危及您作為簡訊寄件者的聲譽。當您的帳戶位在簡訊沙盒中時，您可以使用 Amazon SNS 的所有功能，但您只能傳送簡訊給已驗證的目的地電話號碼。如需詳細資訊，請參閱[使用 Amazon SNS SMS 沙盒](#)。

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選擇訂閱。
3. 在 Subscriptions (訂閱) 頁面，選擇 Create subscription (建立訂閱)。
4. 在 Create subscription (建立訂閱) 頁面上，於 Details (詳細資訊) 區段中，執行以下作業：
 - a. 適用於 ARN 主題中，輸入或選擇您要傳送簡訊之主題的 Amazon 資源名稱 (ARN)。
 - b. 針對 Protocol (通訊協定)，選擇 SMS。

- c. 針對 Endpoint (端點)，輸入您要訂閱主題的電話號碼。
5. 選擇建立訂閱。訂閱資訊會顯示於 Subscriptions (訂閱) 頁面上。

若要新增更多電話號碼，請重複這些步驟。您也可以新增其他類型的訂閱，例如電子郵件。

傳送訊息

當您發佈訊息至主題時，Amazon SNS 會嘗試傳遞該訊息到每個已訂閱主題的電話號碼。

1. 在 [Amazon SNS 主控台](#) 中，在主題頁面上，選擇您要傳送簡訊的主題名稱。
2. 在主題詳細資訊頁面上，選擇 Publish message (發佈訊息)。
3. 在 Publish message to topic (發佈訊息至主題) 頁面上，Message details (訊息詳細資訊) 下方，執行下列步驟：
 - a. 將 Subject (主旨) 欄位留白，除非您的主題包含電子郵件訂閱而您想要發佈至電子郵件和簡訊訂閱二者。Amazon SNS 使用您在主旨行輸入的電子郵件 Subject (主旨)。
 - b. (選擇性) 對於 Time to Live (TTL) 存留時間 (TTL)，輸入 Amazon SNS 必須將 SMS 訊息傳送給任何行動應用程式端點訂閱者的秒數。
4. 在 Message body (訊息內文) 下方，執行下列動作：
 - a. 對於 Message structure (訊息結構)，選擇 Identical payload for all delivery protocols (所有傳送通訊協定的相同酬載量)，將相同訊息傳送至訂閱主題的所有通訊協定類型。或者，選擇 Custom payload for each delivery protocol (每個傳遞通訊協定的自訂酬載) 為不同通訊協定類型的訂閱者自訂訊息。例如，您可以為電話號碼訂閱者輸入預設訊息，以及為電子郵件訂閱者輸入自訂訊息。
 - b. 對於 Message body to send to the endpoint (要傳送至端點的訊息內文)，輸入您的訊息或您對每個傳遞通訊協定的自訂訊息。

如果您的主題有顯示名稱，Amazon SNS 會將其新增至訊息，這會增加訊息的長度。顯示名稱長度是名稱的字元數加上 Amazon SNS 所新增的右尖括號 (>) 和空格兩個字元。

如需有關簡訊大小配額的詳細資訊，請參閱 [使用 Amazon SNS 將簡訊發佈到行動電話](#)。

5. (選擇性) 對於 Message attributes (訊息屬性)，新增郵件中繼資料，例如時間戳記、簽名和 ID。
6. 選擇 Publish message (發佈訊息)。Amazon SNS 傳送簡訊並顯示成功訊息。

使用 SDKs 傳送訊息 AWS 至主題

若要使用 AWS SDK，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS 開發套件和工具參考指南中的[共享的配置和認證文件](#)。

以下代碼範例顯示做法：

- 建立 Amazon SNS 主題。
- 使用手機號碼訂閱主題。
- 將簡訊發布至主題，讓所有訂閱的電話號碼一次接收訊息。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

創造一個主題並回傳其 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 "";
}
}
```

讓端點訂閱主題

```
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);
        }
    }
}

```

設定訊息的屬性，例如寄件者的 ID、最高價格及其類型。訊息屬性為選用。

```

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);
        }
    }
}

```

發布訊息至主題。訊息會傳送至每位訂閱者。

```

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);  
        }  
    }  
}
```

使用 Amazon SNS 將簡訊發佈到行動電話

您可以使用 Amazon SNS 直接將簡訊傳送到行動電話，而無需訂閱 Amazon SNS 主題的電話號碼。

Note

如果您想要一次將一個訊息傳送至多個電話號碼，訂閱電話號碼到主題仍然相當實用。如需將簡訊發布至主題的指示，請參閱 [將簡訊發佈至 Amazon SNS 主題](#)。

傳送訊息時，您可以控制是否針對成本或可靠傳遞將訊息最佳化。您也可以指定 [寄件者 ID 或來源編號](#)。如果您使用 Amazon SNS API 或 AWS SDKs 以程式設計方式傳送訊息，您可以指定訊息傳遞的最高價格。

每則簡訊最多可包含 140 個位元組，而字元配額則依編碼機制而定。例如，簡訊可包含：

- 160 個 GSM 字元
- 140 個 ASCII 字元
- 70 個 UCS-2 字元

如果您發布超過大小配額的簡訊，Amazon SNS 會將其分割為多個訊息傳送，每一個均在大小配額以內。訊息不會在字詞中間切割，而是在整個字詞的邊界切割。單一簡訊發布動作的總計大小配額是 1,600 位元組。

傳送簡訊時，您可以使用 E.164 格式指定電話號碼，這是用於國際電信的標準電話號碼結構。遵循此格式的電話號碼最多可以有 15 位數，前面加上加號 (+) 字元和國碼。例如，E.164 格式的美國電話號碼顯示為 +1XXX5550100。

傳送訊息 (主控台)

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台功能表中，選擇 [支援簡訊的區域](#)。
3. 在導覽窗格中，選擇 Text messaging (SMS) (簡訊 (SMS))。
4. 在 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面中，選擇 Publish text message (發布簡訊)。
5. 在 Publish SMS message (發布簡訊) 頁面上的 Message type (訊息類型) 下，選擇下列其中一個項目：
 - Promotional (促銷) - 非重要訊息，例如行銷訊息。
 - 交易 - 支援客戶交易的重要訊息，例如多重要素驗證用的一次性密碼。

Note

此訊息層級設定會覆寫您的帳戶層級預設訊息類型。您可以在 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面上的 Text messaging preferences (文字簡訊偏好設定) 區段設定帳戶層級的預設郵件類型。

如需有關宣傳和交易訊息的定價資訊，請參閱 [Worldwide SMS Pricing](#) (全球簡訊定價)。

6. 對於 Destination phone number (目的地電話號碼)，請輸入您要傳送訊息的電話號碼。
7. 對於 Message (訊息)，請輸入要傳送的訊息。
8. (選擇性) 在 Origination identities (來源身分) 下，指定如何向收件者識別您的身分：
 - (選用) 若要指定 Sender ID (寄件者 ID)，請輸入包含 3-11 個英數字元 (至少包括一個字母且無空格) 的自訂 ID。寄件者 ID 會在接收的裝置上顯示為訊息寄件者。例如，您可以使用您的商業品牌使訊息來源更容易辨識。

支援依國家和/或區域而異的寄件者 ID。例如，傳送到美國電話號碼的訊息不會顯示寄件者 ID。如需支援寄件者 IDs 的國家和區域，請參閱 AWS End User Messaging SMS 《使用者指南》中的 [使用 傳送簡訊的支援國家和區域 AWS End User Messaging SMS](#)。

如果您未指定寄件者 ID，下列其中一個項目會顯示為來源身分：

- 在支援長碼的國家/地區，會顯示長碼。
- 在僅支援寄件者 ID 的國家/地區，會顯示 NOTICE。

此訊息級寄件者 ID 會覆寫您在 Text messaging preferences (簡訊偏好設定) 頁面設定的預設寄件者 ID。

- 若要指定來源號碼，輸入 5-14 個號碼的字串，以在接收者裝置上顯示為寄件者的電話號碼。此字串必須符合 AWS 帳戶在目的地國家/地區中設定的起始號碼。來源號碼可以是 10DLC 號碼、免付費號碼、人對人長碼或短碼。如需詳細資訊，請參閱[Amazon SNS SMS 訊息的來源身分](#)。

如果您沒有指定來源號碼，Amazon SNS 會根據您的 AWS 帳戶組態選擇。

- 如果您要傳送簡訊給印度收件人，請展開 Country-specific attributes (特定國家屬性)，並指定下列屬性：

- 實體 ID - 傳送簡訊給印度收件人的實體 ID 或主體實體 (PE) ID。此 ID 是 1-50 個字元的特殊字串，印度電信管理局 (TRAI) 提供用來識別您在 TRAI 註冊的實體。
- 範本 ID - 傳送簡訊給印度收件人的範本 ID。此 ID 是 TRAI 提供的特殊字串，由 1-50 個字元組成，可識別您向 TRAI 註冊的範本。範本 ID 必須與您為訊息指定的寄件者 ID 相關聯。

如需傳送簡訊給印度收件人的詳細資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的[印度寄件者 ID 註冊程序](#)。

- 選擇 Publish message (發佈訊息)。

Tip

若要從原始編號傳送 SMS 訊息，您也可以選擇位於 Amazon SNS 主控台導覽面板中的 來源號碼。選擇功能欄中包含簡訊的來源號碼，然後選擇 Publish text message (發布簡訊)。

傳送訊息 (AWS SDKs)

若要使用其中一個 AWS SDKs 傳送簡訊，請使用該 SDK 中的 API 操作，該 SDK 對應至 Amazon SNS API 中的 Publish 請求。您可以透過此請求，將簡訊直接傳送到電話號碼。您也可以使用 MessageAttributes 參數，為下列屬性名稱設定值：

AWS.SNS.SMS.SenderID

包含 3-11 個英數字元或連字號 (-) 字元，且至少包括一個字母且無空格的自訂 ID。寄件者 ID 會在接收的裝置上顯示為訊息寄件者。例如，您可以使用您的商業品牌使訊息來源更容易辨識。

支援依國家或區域而異的寄件者 ID。例如，傳送到美國電話號碼的訊息不會顯示寄件者 ID。如需支援寄件者 IDs 的國家/地區或區域清單，請參閱AWS End User Messaging SMS 《使用者指南》中的[使用 傳送簡訊的支援國家/地區和區域 AWS End User Messaging SMS](#)。

如果您未指定寄件者 ID，訊息將顯示[長代碼](#)做為支援之國家或區域的寄件者 ID。針對要求字母寄件者 ID 的國家或區域，寄件者 ID 會以 NOTICE 顯示。

此訊息層級屬性會覆寫您使用 SetSMSAttributes 請求設定的帳戶層級屬性 DefaultSenderId。

AWS.MM.SMS.OriginationNumber

5-14 個數字的自訂字串，其中可以包含可選的前導加號 (+)。此數字字串會在接收裝置上顯示為寄件者的電話號碼。字串必須符合您 AWS 帳戶中為目的地國家/地區設定的起始號碼。來源號碼可以是 10DLC 號碼、免付費號碼、人對人 (P2P) 長碼或短碼。如需詳細資訊，請參閱AWS End User Messaging SMS 《使用者指南》中的[電話號碼](#)。

如果您未指定起始號碼，Amazon SNS 會根據 AWS 您的帳戶組態選擇起始號碼。

AWS.SNS.SMS.MaxPrice

您願意為傳送簡訊花費的金額上限 (單位為美元)。Amazon SNS 判定若如此做會產生超出最高價的成本，就不會傳送訊息。

如果您本月至今的簡訊成本已經超出為 MonthlySpendLimit 屬性設定的配額，此屬性不會有任何作用。您可以使用屬性 SetSMSAttributes 請求設定 MonthlySpendLimit。

若您傳送訊息至 Amazon SNS 主題，每個傳送到各個已訂閱主題電話號碼的訊息都會套用最高價。

AWS.SNS.SMS.SMSType

所要傳送訊息的類型：

- **Promotional** (預設) – 非重要訊息，例如行銷訊息。
- **Transactional**– 支援客戶交易的重要訊息，例如多重要素驗證用的一次性密碼。

此訊息層級屬性會覆寫您使用 SetSMSAttributes 請求設定的帳戶層級屬性 DefaultSMSType。

AWS.MM.SMS.EntityId

此屬性僅適用於傳送簡訊給印度收件人。

這是您的實體 ID 或主體實體 (PE) 識別碼，用來傳送簡訊給印度收件者。此 ID 是 1-50 個字元的特殊字串，印度電信管理局 (TRAI) 提供用來識別您在 TRAI 註冊的實體。

AWS.MM.SMS.TemplateId

此屬性僅適用於傳送簡訊給印度收件人。

這是傳送簡訊給印度收件人的範本。此 ID 是 TRAI 提供的特殊字串，由 1-50 個字元組成，可識別您向 TRAI 註冊的範本。範本 ID 必須與您為訊息指定的寄件者 ID 相關聯。

傳送訊息

下列程式碼範例示範如何使用 Amazon SNS 發佈簡訊。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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}");
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [發佈](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * 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();
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的[發佈](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[發佈](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[發佈](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}

```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#publish-to-a-text-message-sms-message>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[發佈](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[發佈](#)。

在 Amazon SNS 中設定簡訊偏好設定

使用 Amazon SNS 指定 SMS 訊息的偏好設定。例如，您可以指定是否針對成本或可靠性最佳化傳遞、您的每月花費限制、如何記錄傳遞，以及是否訂閱每日簡訊用量報告。

這些喜好設定會對您從帳戶傳送的每個簡訊生效，但是您可以在傳送個別訊息時覆寫部分設定。如需詳細資訊，請參閱[使用 Amazon SNS 將簡訊發佈到行動電話](#)。

使用 AWS Management Console 設定簡訊喜好設定

1. 登入 [Amazon SNS 主控台](#)。
2. 選擇一個[支援簡訊的區域](#)。
3. 在導覽面板上，選擇行動，然後選擇簡訊 (SMS)。
4. 在 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面上，於 Text messaging preference (簡訊喜好設定) 區段中，選擇 Edit (編輯)。
5. 在 Edit text messaging preferences (編輯簡訊喜好設定) 頁面上，在 Details (詳細資訊) 中，執行以下作業：

- a. 針對 Default message type (預設訊息類型)，選擇以下其中一個項目：
- 行銷活動 - 非關鍵訊息 (例如行銷)。Amazon SNS 會將訊息傳遞最佳化為產生最低的成本。
 - 交易 (預設) - 支援客戶交易的重要訊息，例如多重要素驗證用的一次性密碼。Amazon SNS 會將訊息傳遞最佳化為達成最高的可靠性。

如需有關宣傳和交易訊息的定價資訊，請參閱[全球簡訊定價](#)。

- b. (選用) 針對 Account spend limit (帳戶費用限制)，請輸入您每個月要花費在簡訊上的金額上限 (單位為 USD)。

 Important

- 根據預設，費用配額會設為 1.00 USD。如果您想要提高服務配額，[請提交請求](#)。
- 若主控台中設定的金額超過您的服務配額，Amazon SNS 會停止發布簡訊。
- 因為 Amazon SNS 是分散式的系統，它會在超過費用配額的數分鐘內停止傳送簡訊。在此間隔期間，若您繼續傳送簡訊，您可能會產生超過您配額的成本。

6. (選用) 針對 Default sender ID (預設寄件者 ID)，輸入自訂 ID (例如您的商業品牌)，該 ID 會顯示為接收裝置的寄件者。

 Note

寄件者 ID 的支援情況會因國家或區域而不同。

7. (選用) 輸入 Amazon S3 儲存貯體用量報告名稱。

 Note

Amazon S3 儲存貯體政策必須授予 Amazon SNS 的寫入存取權。

8. 選擇 Save changes (儲存變更)。

設定偏好設定AWS SDKs)

若要使用其中一個 AWS SDKs 設定您的 SMS 偏好設定，請在該 SDK 中使用與 Amazon SNS API 中 `SetSMSAttributes` 請求對應的動作。透過此請求，您可以指派值給不同簡訊屬性，例如您的每月費用配額以及您的預設簡訊類型 (促銷或交易)。如需所有 SMS 屬性，請參閱 Amazon Simple Notification Service API 參考 中的 [SetSMSAttributes](#)。

下列程式碼範例示範如何使用 `SetSMSAttributes`。

C++

SDK for C++

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

如何使用 Amazon SNS 設定 `DefaultSMSType` 屬性。

```
#!/ 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();  
}
```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [SetSMSAttributes](#)。

CLI

AWS CLI

若要設定簡訊屬性

下列 `set-sms-attributes` 範例會將簡訊的預設寄件者 ID 設定為 `MyName`。

```
aws sns set-sms-attributes \  
    --attributes DefaultSenderId=MyName
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetSMSAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [SetSMSAttributes](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [SetSMSAttributes](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
$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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#set-sms-attributes>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [SetSMSAttributes](#)。

設定特定國家/地區交付的簡訊偏好設定

您可以僅將訊息傳送至特定目的地國家/地區，來管理和控制 SMS 流量。這可確保您的訊息只會傳送到核准的國家/地區，避免不必要的簡訊費用。下列指示使用 Amazon Pinpoint 的 Protect 組態來指定您要允許或封鎖的國家/地區。

1. 開啟 AWS SMS 主控台，網址為 <https://console.aws.amazon.com/sms-voice/>：//。
2. 在導覽窗格的概觀下，於快速入門區段中，選擇建立保護組態。
3. 在保護組態詳細資訊下，輸入保護組態的易用名稱（例如 Allow-Only-AU）。
4. 在 SMS 國家/地區規則下，選取區域/國家/地區核取方塊，封鎖傳送訊息至所有支援的國家/地區。
5. 取消選取要傳送訊息的國家/地區的核取方塊。例如，若要僅允許傳送至澳洲的訊息，請取消選取澳洲的核取方塊。
6. 在保護組態關聯區段的關聯類型下，選取帳戶預設。這將確保 AWS End User Messaging SMS 保護組態會影響透過 Amazon SNS、[Amazon Cognito](#) 和 Amazon Pinpoint [SendMessages](#) API 呼叫傳送的所有訊息。
7. 選擇建立保護組態以儲存您的設定。

隨即顯示下列確認訊息：

```
Success Protect configuration protect-abc0123456789 has been created.
```

8. 登入 [Amazon SNS 主控台](#)。

9. [發佈訊息](#)到其中一個封鎖的國家/地區，例如印度。

訊息將不會傳送。您可以使用 [CloudWatch](#) 在交付失敗日誌中驗證這一點。搜尋日誌群組 `sns/region/AccountID/DirectPublishToPhoneNumber/Failure`，以取得類似下列範例的回應：

```
{
  "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"
}
```

管理 Amazon SNS 電話號碼和訂閱

Amazon SNS 提供數個選項管理誰從您的帳戶接收簡訊。藉由有限的頻率，您可以加入已停止從您的帳戶接收簡訊的電話號碼。若要停止傳送訊息到簡訊訂閱，您可以移除訂閱或發布到訂閱的主題。

停止接收簡訊

如果當地法律法規（例如美國和加拿大）要求，SMS 收件人可以使用其裝置，透過下列任何方式回覆訊息來選擇退出：

- ARRET (法文)
- 取消
- 結束
- 停止接收
- 停止接收
- 結束
- 移除
- 停止
- TD

- 取消訂閱

若要選擇退出，收件人必須回覆與 Amazon SNS 用來傳遞訊息的相同[起始號碼](#)。選擇不接收後，收件人將不再收到來自的簡訊，AWS 帳戶 除非您選擇接收電話號碼。

如果電話號碼已訂閱 Amazon SNS 主題，選擇退出不會移除訂閱，但除非您選擇加入電話號碼，否則簡訊將無法傳遞至該訂閱。

使用 Amazon SNS 主控台管理電話號碼和訂閱

您可以使用 Amazon SNS 主控台來從您的帳戶控制哪個電話號碼來接收簡訊。

選擇加入已選擇退出 Amazon SNS 主控台的電話號碼

您可以檢視哪些電話號碼選擇不接收來自您帳戶的簡訊，也可以選擇加入這些電話號碼以繼續傳送訊息給他們。

您每 30 天只能選擇加入電話號碼一次。

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台選單中，設定區域選擇器為[支援簡訊的區域](#)。
3. 在導覽面板上，選擇 Text messaging (SMS) (簡訊 (SMS))。
4. 在行動簡訊 (SMS) 頁面的選擇退出電話號碼區段中，會顯示選擇退出的電話號碼。
5. 選取您要加入的電話號碼的核取方塊，然後選擇加入。該電話號碼不再選擇退出，且將會收到您傳送給它的簡訊。

刪除 Amazon SNS 主控台的簡訊訂閱

刪除簡訊訂閱以停止傳送簡訊至您發布到主題的電話號碼。

1. 在導覽面板上，選擇 Subscriptions (訂閱)。
2. 選擇您要刪除之訂閱的核取方塊。然後選擇 Actions (動作)，再選擇 Delete Subscriptions (刪除訂閱)。
3. 在 Delete (刪除) 視窗中，選擇 Delete (刪除)。Amazon SNS 會刪除訂閱並顯示成功訊息。

刪除 Amazon SNS 主控台的主題

當您不再想要發布訊息其訂閱的端點，請刪除該主題。

1. 在導覽面板上，選擇 Topics (主題)。
2. 選擇您要刪除之主題的核取方塊。然後選擇 Actions (動作)，再選擇 Delete Topics (刪除主題)。
3. 在 Delete (刪除) 視窗中，選擇 Delete (刪除)。Amazon SNS 會刪除主題並顯示成功訊息。

使用 AWS SDK 管理電話號碼和訂閱

您可以使用 AWS SDKs 向 Amazon SNS 提出程式設計請求，並管理哪些電話號碼可以從您的帳戶接收簡訊。

若要使用 AWS SDK，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS SDKs [和工具參考指南中的共用組態和登入資料檔案](#)。

使用 AWS SDK 檢視所有選擇退出的電話號碼

若要檢視所有選擇退出的電話號碼，請使用 Amazon SNS API 提交 `ListPhoneNumbersOptedOut` 請求。

下列程式碼範例示範如何使用 `ListPhoneNumbersOptedOut`。

CLI

AWS CLI

列出停止接收簡訊

下列 `list-phone-numbers-opted-out` 範例會列出選擇停止接收簡訊的電話號碼。

```
aws sns list-phone-numbers-opted-out
```

輸出：

```
{
  "phoneNumbers": [
    "+15555550100"
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ListPhoneNumbersOptedOut](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ListPhoneNumbersOptedOut](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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());  
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#list-opted-out-phone-numbers>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [ListPhoneNumbersOptedOut](#)。

使用 AWS SDK 檢查電話號碼是否選擇退出

若要檢查電話號碼是否已選擇退出，請使用 Amazon SNS API 提交 `CheckIfPhoneNumberIsOptedOut` 請求。

下列程式碼範例示範如何使用 `CheckIfPhoneNumberIsOptedOut`。

.NET

適用於 .NET 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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}");  
}  
}  
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

CLI

AWS CLI

檢查電話號碼是否停止接收簡訊

下列 `check-if-phone-number-is-opted-out` 範例會檢查指定的電話號碼是否選擇不接收來自目前 AWS 帳戶的簡訊。

```
aws sns check-if-phone-number-is-opted-out \  
--phone-number +1555550100
```

輸出：

```
{  
  "isOptedOut": false  
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CheckIfPhoneNumberIsOptedOut](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#check-if-a-phone-number-has-opted-out>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

選擇使用 Amazon SNS API 選擇退出的電話號碼

若要選擇加入電話號碼，請使用 Amazon SNS API 提交 `OptInPhoneNumber` 請求。

您每 30 天只能選擇加入電話號碼一次。

使用 AWS SDK 刪除簡訊訂閱

若要從 Amazon SNS 主題刪除簡訊訂閱，請使用 Amazon SNS API 提交 `ListSubscriptions` 請求來取得訂閱 ARN，然後傳遞 ARN 至 `Unsubscribe` 請求。

下列程式碼範例示範如何使用 `Unsubscribe`。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

透過訂閱 ARN 取消訂閱主題。

```
/// <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;
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [取消訂閱](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! 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();
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的[取消訂閱](#)。

CLI

AWS CLI

取消訂閱主題

下列 unsubscribe 範例會從主題中刪除指定的訂閱。

```
aws sns unsubscribe \
  --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
  topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [Unsubscribe](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[取消訂閱](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;  
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [取消訂閱](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun unSub(subscriptionArnVal: String) {  
    val request =  
        UnsubscribeRequest {  
            subscriptionArn = subscriptionArnVal  
        }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient.unsubscribe(request)  
        println("Subscription was removed for ${request.subscriptionArn}")  
    }  
}
```

- 如需 API 詳細資訊，請參閱 [《適用於 Kotlin 的 AWS SDK API 參考》](#) 中的 [取消訂閱](#)。

PHP

SDK for PHP

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-subscribing-unsubscribing-topics.html#unsubscribe-from-a-topic>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[取消訂閱](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[取消訂閱](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的[取消訂閱](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.")
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的[取消訂閱](#)。

使用 SDK AWS 刪除主題

若要刪除主題及其所有訂閱，請使用 Amazon SNS API 提交 ListTopics 請求來取得主題 ARN，然後傳遞 ARN 至 DeleteTopic 請求。

下列程式碼範例示範如何使用 DeleteTopic。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

藉由主題 ARN 刪除該主題。

```
/// <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;
}

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [DeleteTopic](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! 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();
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

CLI

AWS CLI

刪除 SNS 主題

下列 delete-topic 範例會刪除指定的 SNS 主題。

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

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteTopic](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}

```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [DeleteTopic](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DeleteTopic](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```

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({});

```

匯入 SDK 和用戶端模組，然後呼叫 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
  //   }
  // }
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

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

```
}  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 DeleteTopic。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}

```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [DeleteTopic](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [DeleteTopic](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import AWSSNS
```

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

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

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [DeleteTopic](#)。

Amazon SNS SMS 活動監控

您可以透過監控簡訊活動來追蹤目標電話號碼、成功或失敗的傳送、失敗的原因、成本和其他資訊。Amazon SNS 透過在主控台中摘要統計資料、傳送訊息至 Amazon CloudWatch 和傳送每日簡訊用量報告到您指定的 Amazon S3 儲存貯體來提供協助。

檢視 Amazon SNS SMS 交付統計資料

您可以使用 Amazon SNS 主控台檢視最近簡訊傳遞的統計。

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台選單中，設定區域選擇器為[支援簡訊的區域](#)。
3. 在導覽面板上，選擇 Text messaging (SMS) (簡訊 (SMS))。
4. 在 Text messaging (SMS) (簡訊 (SMS)) 頁面上，檢視 Account stats (帳戶統計資訊) 區段中的有關您的交易和宣傳簡訊傳遞的圖表。每個圖表顯示前面 15 天的下列資料：
 - 傳送率 (成功傳送的百分比)
 - 已傳送 (嘗試傳送的次數)
 - 已失敗 (傳送失敗的次數)

在此頁面上，您也可以選擇 Usage (用量) 按鈕，前往您存放每日用量報告的 Amazon S3 儲存貯體。如需詳細資訊，請參閱[訂閱 Amazon SNS 每日簡訊用量報告](#)。

使用 Amazon CloudWatch 指標和日誌的 Amazon SNS SMS 交付監控 Amazon CloudWatch

您可以使用 Amazon CloudWatch 和 Amazon CloudWatch Logs 監控您的簡訊傳遞。

檢視 Amazon CloudWatch 指標

Amazon SNS 會自動收集有關您的簡訊傳遞的指標並將其推送至 Amazon CloudWatch。您可以使用 CloudWatch 監控這些指標，並建立提醒在指標超過閾值時提醒您。例如，您可以監控 CloudWatch 指標，以得知您的簡訊傳送率和您本月至今的簡訊費用。

如需有關監控 CloudWatch 指標、設定 CloudWatch 提醒和可用指標類型的資訊，請參閱 [使用 Amazon CloudWatch 監控 Amazon SNS 主題](#)。

檢視 CloudWatch Logs

您可以啟用 Amazon SNS 寫入 Amazon CloudWatch Logs 來收集有關成功和未成功傳送簡訊的資訊。對於您傳送的每一則簡訊，Amazon SNS 都會寫入日誌，其中包括訊息價格、狀態為成功或失敗、失敗的原因 (如果訊息失敗的話)、訊息駐留時間和其他資訊。

為簡訊啟用和檢視 CloudWatch Logs

1. 登入 [Amazon SNS 主控台](#)。
2. 在主控台選單中，設定區域選擇器為[支援簡訊的區域](#)。
3. 在導覽面板上，選擇 Text messaging (SMS) (簡訊 (SMS))。
4. 在 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面上，於 Text messaging preference (簡訊喜好設定) 區段中，選擇 Edit (編輯)。
5. 在下一個頁面上，展開 Delivery status logging (交付狀態記錄日誌) 區段。
6. 對於 Success sample rate (成功取樣率)，指定簡訊成功傳遞的百分比，且 Amazon SNS 將在 CloudWatch Logs 中寫入日誌。例如：
 - 若要僅針對失敗的傳遞寫入日誌，將此值設為 0。
 - 若要針對 10% 的成功傳遞寫入日誌，將其設為 10。

如果不指定百分比，Amazon SNS 會將所有成功的傳遞寫入日誌。

7. 若要提供必要的許可，請進行下列其中一個動作：
 - 若要建立新的服務角色，請選擇 Create new service role (建立新服務角色) 然後 Create new roles (建立新角色)。在下一頁上，選擇 Allow (允許)，為您的帳戶資源提供 Amazon SNS 寫入存取權。
 - 若要使用現有的服務角色，請選擇 Use existing service role (使用現有的服務角色)，然後將 ARN 名稱貼到 IAM role for successful and failed deliveries (成功和失敗交付的 IAM 角色) 方塊。

您指定的服務角色必須允許對帳戶資源的寫入存取權限。如需建立 IAM 角色的詳細資訊，請參閱《IAM 使用者指南》中的[為 AWS 服務建立角色](#)。

8. 選擇 Save changes (儲存變更)。
9. 回到 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面上，移至 Delivery status logs (傳遞狀態記錄) 區段以檢視任何可用的記錄檔。

 Note

根據目的地電話號碼的電信業者，Amazon SNS 主控台最多可能需要 72 小時的時間才會出現在 Amazon SNS 主控台中。

成功傳送簡訊的範例日誌

成功傳送簡訊的傳遞狀態日誌類似以下範例：

```
{
  "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"
}
```

傳送簡訊失敗的範例日誌

傳送簡訊失敗的傳遞狀態日誌類似以下範例：


```
{
  "notification": {
    "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
    "timestamp": "2016-06-28 00:40:34.559"
  },
  "delivery": {
    "mnc": 0,
    "numberOfMessageParts": 1,
    "destination": "+1XXX5550100",
    "priceInUSD": 0.00645,
    "smsType": "Transactional",
    "mcc": 0,
    "providerResponse": "Unknown error attempting to reach phone",
    "dwellTimeMs": 1420,
    "dwellTimeMsUntilDeviceAck": 1692
  },
  "status": "FAILURE"
}
```

簡訊傳遞失敗的原因

提供具有 `providerResponse` 屬性的失敗原因。簡訊可能因為下列原因傳遞失敗：

- 被電話的電信業者視為垃圾郵件而封鎖
- 目的地位於封鎖清單上
- 電話號碼無效
- 訊息內文無效
- 電話電信業者已封鎖此訊息
- 電話電信業者目前無法聯絡上/無法使用
- 電話已封鎖簡訊
- 手機在封鎖清單上
- 電話目前無法聯絡上/無法使用
- 電話號碼退訂
- 此傳送會超出價格上限
- 嘗試聯絡電話時發生未知的錯誤

訂閱 Amazon SNS 每日簡訊用量報告

您可以從 Amazon SNS 訂閱每日用量報告來監控 SMS 訊息傳送。如果您每天傳送至少一則 SMS 訊息，Amazon SNS 就會將用量報告 CSV 檔傳遞到指定的 Amazon S3 儲存貯體。Amazon S3 儲存貯體中需要 24 小時才能使用簡訊用量報告。

每日用量報告資訊

用量報告針對每則透過您的帳戶傳送的 SMS 訊息，包含下列資訊。

此報告不包含收件人已停止接收的訊息。

- 訊息的發佈時間 (以 UTC 表示)
- 訊息 ID
- 目的地電話號碼
- 訊息類型
- 傳遞狀態
- 訊息價格 (USD)
- 分段編號 (如果訊息內容太長，就會分成多個分段)
- 分段總數

Note

如果 Amazon SNS 沒有收到組件編號，我們將其值設定為零。

訂閱每日用量報告

若要訂閱每日用量報告，您必須使用適當的許可建立 Amazon S3 儲存貯體。

為每日用量報告建立 Amazon S3 儲存貯體

1. 從傳送簡訊 AWS 帳戶的，登入 [Amazon S3 主控台](#)。
2. 選擇 Create Bucket (建立儲存貯體)。
3. 針對 Bucket Name (儲存貯體名稱)，建議您輸入帳戶和組織唯一的名稱。例如，使用樣式 <my-bucket-prefix>-<account_id>-<org-id>。

如需儲存貯體名稱慣例和限制的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的[儲存貯體命名規則](#)。

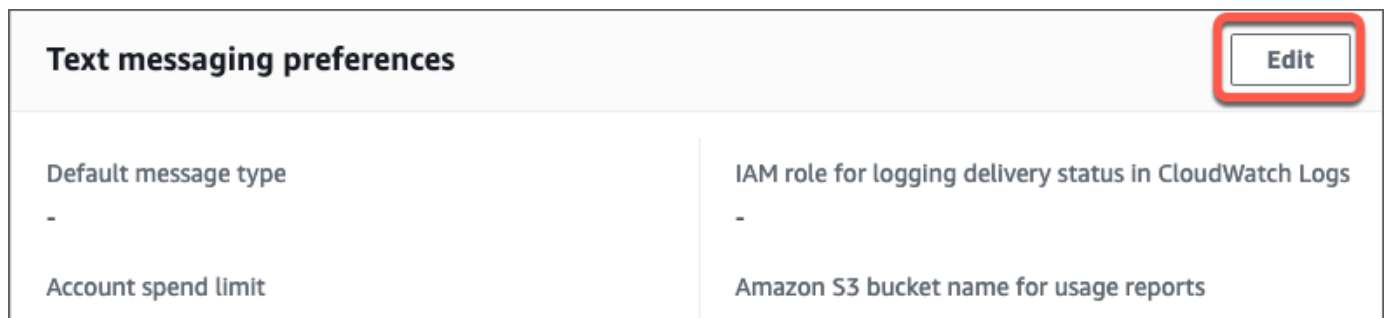
4. 選擇建立。
5. 在 All Buckets (所有儲存貯體) 表格中，選擇儲存貯體。
6. 在 Permissions (許可) 索引標籤中，選擇 Bucket policy (儲存貯體政策)。
7. 在 Bucket Policy Editor (儲存貯體政策編輯器) 視窗中，提供允許 Amazon SNS 服務委託人寫入您的儲存貯體的政策。如需範例，請參閱「[儲存貯體政策的範例](#)」。

如果您使用範例原則，請記得將 *my-s3-bucket* 取代為您在步驟 3 中選擇的儲存貯體名稱。

8. 選擇儲存。

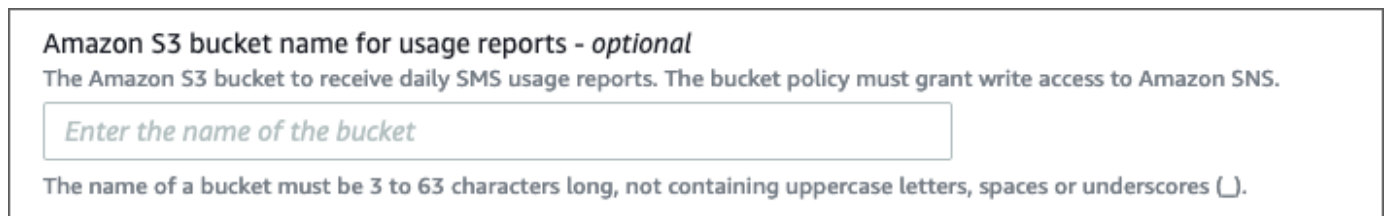
訂閱每日用量報告

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Text messaging (SMS) (簡訊 (SMS))。
3. 在 Text messaging (SMS) (簡訊 (SMS)) 頁面上，於 Text messaging preference (簡訊喜好設定) 區段中，選擇 Edit (編輯)。



The screenshot shows the 'Text messaging preferences' section of the Amazon SNS console. It includes an 'Edit' button in the top right corner, which is highlighted with a red rectangular box. Below the title, there are 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'.

4. 在 Edit text messaging preferences (編輯文字簡訊喜好設定) 頁面上，於 Details (詳細資訊) 區段中，指定 Amazon S3 bucket name for usage reports (用量報告的 Amazon S3 儲存貯體名稱)。



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. 選擇儲存變更。

儲存貯體政策的範例

下列政策允許 Amazon SNS 服務委託人執行 `s3:PutObject`、`s3:GetBucketLocation` 和 `s3:ListBucket` 動作。

AWS 為具有已授予您帳戶中資源存取權之服務主體的所有 服務提供工具。當 Amazon S3 儲存貯體政策陳述式中的主體是[混淆代理人問題](#)時。若要限制儲存貯體可以接收每日用量報告報告的區域和帳戶，請使用 `aws:SourceArn` (如以下範例所示)。如果您不希望限制可以產生這些報告的區域，請使用 `aws:SourceAccount` 來根據產生報告的帳戶進行限制。如果您不知道資源的 ARN，請使用 `aws:SourceAccount`。

在建立用以接收來自 Amazon SNS 每日 SMS 用量報告的 Amazon S3 儲存貯體時，請使用下列包含混淆代理人保護的範例。

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Sid": "AllowPutObject",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "account_id"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:region:account_id:*"
        }
      }
    },
    {
      "Sid": "AllowGetBucketLocation",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "s3:GetBucketLocation",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
```

```

        "StringEquals": {
            "aws:SourceAccount": "account_id"
        },
        "ArnLike": {
            "aws:SourceArn": "arn:aws:sns:region:account_id:*"
        }
    },
    {
        "Sid": "AllowListBucket",
        "Effect": "Allow",
        "Principal": {
            "Service": "sns.amazonaws.com"
        },
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "account_id"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:sns:region:account_id:*"
            }
        }
    }
]
}

```

Note

您可以發佈用量報告到 Amazon S3 政策中 Condition 元素指定的 AWS 帳戶所擁有的 Amazon S3 儲存貯體。若要將用量報告發佈至另一個 AWS 帳戶擁有的 Amazon S3 儲存貯體，請參閱[如何從另一個儲存貯體複製 Amazon S3 物件 AWS 帳戶？](#)。

每日用量報告的範例

在您訂閱每日用量報告後，Amazon SNS 每天都會將具有用量資料的 CSV 檔案存放在下列位置：

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

每個檔案可包含最多 50,000 筆記錄。如果一日的記錄量超過此配額，Amazon SNS 會新增多個檔案。報告範例如下所示：

```
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
```

請求支援 Amazon SNS SMS 訊息

Important

Amazon SNS SMS 開發人員指南已更新。Amazon SNS 已與 [整合](#)，[AWS End User Messaging SMS](#) 以傳遞簡訊。本指南包含如何建立、設定和管理 Amazon SNS SMS 訊息的最新資訊。

除非您聯絡，否則 AWS 您的帳戶無法使用具有 Amazon SNS 的特定簡訊選項支援。在 [AWS 支援中心](#) 開立一個案例，以請求以下任何項目：

- 提高您的每月簡訊花費閾值

在預設情況下，每月花費閾值為 1.00 USD (美元)。您的花費閾值會決定您可以使用 Amazon SNS 傳送的訊息量。您可以請求符合簡訊使用案例所預期每月訊息量的花費閾值。

- 從 [簡訊沙盒](#) 移動，讓您不受限制地傳送簡訊。如需詳細資訊，請參閱 [移出 Amazon SNS SMS 沙盒](#)。
- 專用 [來源號碼](#)
- 專用 [寄件者 ID](#)。寄件者 ID 是在收件人的裝置上顯示為寄件者的自訂 ID。例如，您可以使用您的商業品牌使訊息來源更容易辨識。支援依國家或區域而異的寄件者 ID。如需詳細資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的 [使用 傳送簡訊的支援國家和地區 AWS End User Messaging SMS](#)。

請求增加您的每月 Amazon SNS SMS 花費配額

Amazon SNS 提供花費配額，幫助您管理每月使用您的帳戶傳送簡訊所產生的成本上限。花費配額可降低您遭受惡意攻擊的風險，並防止您的上游應用程式傳送的訊息數量超出預期。當 Amazon SNS 判定傳送簡訊會致使成本超過當月的花費配額時，您可以設定 Amazon SNS 停止發布簡訊。

為了確保您的營運不受影響，我們建議您所請求的花費配額必須足以支援您的生產工作負載。如需詳細資訊，請參閱[步驟 1：開啟 Amazon SNS 簡訊案例](#)。收到配額後，您可以套用完整配額或較小的值來管理您的風險，如[步驟 2：更新您的簡訊設定](#)所述。套用較小的值可讓您控制每月花費，並視需要選擇擴展。

Important

因為 Amazon SNS 是分散式系統，如果超過花費配額，它會在數分鐘內停止傳送簡訊。在此期間，若您繼續傳送簡訊，則可能會產生超過您配額的成本。

我們將所有新帳戶的花費配額設定為每月 1.00 USD (美元)。此配額旨在讓您測試 Amazon SNS 的訊息傳送功能。若要請求提高帳戶的簡訊花費配額，請在 AWS 支援中心開立配額增加案例。

主題

- [步驟 1：開啟 Amazon SNS 簡訊案例](#)
- [步驟 2：在 Amazon SNS 主控台上更新您的簡訊設定](#)

步驟 1：開啟 Amazon SNS 簡訊案例

您可以在 AWS 支援中心開啟配額增加案例，以請求提高每月花費配額。

Note

申請表中的某些欄位標示為「選用」。不過，支援需要以下步驟中提及的所有資訊，以便處理您的要求。如果您不提供所有必要的資訊，處理您的請求可能會發生延遲。

1. 登入，AWS Management Console 網址為 <http://www.https://console.aws.amazon.com/>。
2. 在支援功能表上，選擇支援中心。
3. 在您的支援案例窗格中，選擇建立案例。
4. 選擇尋找增加服務限制額度？連結，然後完成以下操作：

- 針對限制類型，選擇 SNS 簡訊。
 - (選用) 對於提供將傳送簡訊的網站或應用程式連結，提供有關將傳送簡訊之網站、應用程式或服務的資訊。
 - (選用) 對於您打算傳送的訊息類型，請選擇您計劃使用長碼傳送的訊息類型。
 - 一次性密碼 - 提供密碼給客戶以向網站或應用程式進行身分驗證的訊息。
 - Promotional (促銷) - 提升您的業務或服務的非重要訊息，例如，特殊優惠或公告。
 - Transactional (交易) - 支援客戶交易的重要資訊訊息，例如訂單確認或帳戶提醒。交易訊息不得包含促銷或行銷內容。
 - (選用) 針對 AWS 您要傳送訊息的區域，選擇您要傳送訊息的區域。
 - (選用) 對於您打算傳送訊息的國家/地區，輸入您要購買短碼的國家或區域。
 - (選用) 在您的客戶如何選擇接收您發送的訊息中，提供有關您選擇加入程序的詳細資訊。
 - (選用) 在請提供您計劃用來傳送訊息給客戶的訊息範本欄位中，包含您將使用的範本。
5. 在請求下，填寫以下部分：
- 在區域中，選擇您要從中傳送訊息的區域。

 Note

在要求區段中必須選擇區域。即使您在案例詳情區段中提供了此資訊，也必須在此處將其納入。

- 針對資源類型，選擇一般限制。
 - 對於限制，選擇提高帳戶花費閾值。
6. 對於新限制值，輸入您每個日曆月能花費在簡訊的金額上限 (單位為 USD)。
7. 在案例說明 下，針對使用案例說明提供下列詳細資訊：
- 傳送簡訊的公司或服務的網站或應用程式。
 - 您的網站或應用程式所提供的服務，以及您的簡訊如何協助該服務。
 - 使用者如何註冊以自願接收您的網站上、應用程式或其他位置的簡訊。

如果您請求的花費配額 (對新配額值指定的值) 超過 10,000 USD (美元)，則需針對您要傳送訊息的每個國家/地區，提供以下的其他詳細資訊：

- 您是否使用寄件者 ID 或短碼。如果您使用寄件者 ID，請提供：
 - 寄件者 ID。
 - 該寄件者 ID 是否已向國家/地區中的無線電信業者註冊。
 - 您的訊息預期的每秒交易上限 (TPS)。
 - 平均訊息大小。
 - 您傳送至該國家/地區的訊息範本。
 - (選用) 字元編碼需求 (如果有)。
8. (選用) 如果您要提交任何進一步的要求，請選擇新增其他要求。如果您包含多個請求，請提供每個請求所需的資訊。如需所需的資訊，請參閱 [請求支援 Amazon SNS SMS 訊息](#) 中的其他小節。
 9. 在聯絡選項下，針對偏好的聯絡語言選擇您希望針對此案例收到的通訊。
 10. 完成後，請選擇提交。

支援 團隊會在 24 小時內對您的請求提供初始回應。

為了避免使用我們的系統被用來傳送未經要求或惡意的內容，我們會仔細考慮每個請求。我們會盡量在 24 小時的期間內准許您的請求。不過，如果我們需要向您取得其他資訊，則可能需要更長的時間來處理您的請求。

如果您的使用案例不符合我們的政策，我們可能無法批准您的請求。

步驟 2：在 Amazon SNS 主控台上更新您的簡訊設定

在我們通知您的每月花費配額提高之後，您必須在 Amazon SNS 主控台上調整帳戶的花費配額。

Important

您必須完成以下步驟，否則您的簡訊消費限制將不會增加。

在主控台上調整花費配額

1. 登入 [Amazon SNS 主控台](#)。
2. 開啟左側導覽功能表，展開行動裝置，然後選擇簡訊 (SMS)。
3. 在 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面上，於 Text messaging preference (簡訊喜好設定) 區段中，選擇 Edit (編輯)。

- 在編輯簡訊偏好設定頁面的詳細資料區段，在帳戶消費限制欄位中輸入新的 SMS 消費限制。

 Note

注意：您可能會收到警告，指出輸入的值大於預設支出限制。您可以忽略此資訊。

- 選擇 Save changes (儲存變更)。

 Note

如果您收到「Invalid Parameter」(無效參數) 錯誤，請檢查 AWS Support 的聯絡人，並確認您輸入了正確的新 SMS 支出限制。如果您仍然遇到問題，請在 AWS 支援中心開啟案例。

當您在 支援 中心建立案例時，請務必包含您要提交之請求類型的所有必要資訊。否則，在繼續之前，支援 必須與您聯絡以取得此資訊。透過提交詳細的案例，您可幫助確保您的案例可實現，而不會有延遲。如需特定簡訊請求類型所需的詳細資訊，請參閱下列主題。

如需寄件者 IDs 的詳細資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的下列文件：

AWS End User Messaging SMS 主題	描述
請求增加花費配額	您的花費配額決定您 AWS End User Messaging SMS 每個月可以花多少錢傳送簡訊。
在支援中心為寄件者 ID 開啟案例	如果您計劃將訊息傳送給需要寄件者 IDs 的國家/地區，您可以透過在 支援 中心建立新案例來請求寄件者 ID。

Amazon SNS SMS 訊息的最佳實務

 Important

Amazon SNS SMS 開發人員指南已更新。Amazon SNS 已與 整合，[AWS End User Messaging SMS](#) 以傳遞簡訊。本指南包含如何建立、設定和管理 Amazon SNS SMS 訊息的最新資訊。

行動電話使用者都很難忍受來路不明的簡訊。對於來路不明簡訊行銷活動的回覆率幾乎都很低，因此投資報酬會很差。

此外，行動電話業者會持續稽核大量簡訊寄件者。如果他們認定號碼在傳送來路不明的訊息，則他們會阻擋或封鎖來自這些號碼的訊息。

傳送來路不明的內容也違反 [AWS 可接受的使用政策](#)。Amazon SNS 團隊會定期稽核簡訊行銷活動，如果發現您似乎在傳送來路不明的訊息，就可能會阻擋或禁止您傳送訊息。

最後，在許多國家、區域和轄區，對於傳送來路不明簡訊的情況，有很嚴厲的處罰。例如，在美國，《電話消費者保護法案》(TCPA) 規定，按消費者收到的每一則來路不明訊息，消費者可獲得 500-1,500 USD 的損壞賠償 (由寄件者支付)。

本節描述幾個最佳實務，可能有助於您提升客戶參與度及避免昂貴的罰款。不過，請注意，本節不含法律建議。請一律諮詢律師以取得法律建議。

遵守法律、法規和電信業者要求

如果您違反客戶所在地的法律與法規，您可能會面臨巨額的罰款和處罰。因此，務必了解您經營業務的每個國家或區域的簡訊相關法律。

以下清單包含適用於世界各地主要市場的簡訊通訊的重要法律連結。

- 美國：《1991 年電話消費者保護法案》(也稱為 TCPA) 適用於某些類型的簡訊。如需詳細資訊，請參閱聯邦通訊委員會網站上的 [法條和法規](#)。
- 英國：《2003 年隱私與電子通訊 (EC 指令) 法規》(也稱為 PECR) 適用於某些類型的簡訊。如需詳細資訊，請參閱英國資訊委員辦公室網站上的 [什麼是 PECR ?](#)。
- 歐盟：《2002 年隱私與電子通訊指令》(有時稱為 ePrivacy 指令) 適用於某些類型的簡訊。如需詳細資訊，請參閱 Europa.eu 網站上的 [法律全文](#)。
- 加拿大：《打擊網際網路及無線垃圾郵件法案》(更常稱為《加拿大反垃圾郵件法》或 CASL) 適用於某些類型的簡訊。如需詳細資訊，請參閱加拿大國會網站上的 [法律全文](#)。
- 日本：《特定電子郵件傳輸法規法案》可能適用於某些類型的簡訊。如需詳細資訊，請參閱日本總務省網站上的 [日本反垃圾郵件對策](#)。

身為寄件人，即使您的公司或組織不在這些國家/地區，這些法律仍可能適用於您。這份清單中的某些法律當初制定是為了對付來路不明的電子郵件或電話，但經過解釋或擴充之後，也適用於簡訊。其他國家或區域可能有自己的法律來規範簡訊的傳輸。請向客戶所在每個國家或區域的律師諮詢，以獲得法律建議。

在許多國家/地區，當地電信業者最終有權確定哪種流量在其網路上流動。這表示電信業者可能會對超出當地法律最低要求的 SMS 內容施加限制。

取得許可

切勿將訊息傳送給尚未明確要求接收您計劃傳送之特定訊息類型的收件者。即使在同一家公司內的組織中，也不要共享選擇加入名單。

如果收件人可以使用線上表單來註冊接收您的訊息，請新增系統以避免自動化指令碼未告知使用者就強迫訂閱。您也應該限制使用者在單一工作階段中提交電話號碼的次數。

當您收到選擇接收簡訊請求時，請傳送訊息給收件人，請他們確認希望接收您的訊息。在收件人確認訂閱之前，請勿傳送任何額外的訊息給他們。訂閱確認訊息可能類似以下範例：

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may  
apply. Reply HELP for help, STOP to cancel.
```

維護記錄，包括每個選擇接收請求和確認的日期、時間和來源。這在電信業者或法規機構需要時可能非常有用，也可協助您對客戶清單執行例行稽核。

選擇加入工作流程

在某些情況下 (例如美國免付費電話或短碼註冊)，行動電信業者會要求您提供整個選擇加入工作流程的圖樣或螢幕擷取畫面。圖樣或螢幕擷取畫面必須與收件者將完成的選擇加入工作流程非常相似。

您的圖樣或螢幕擷取畫面應包含下列所有必要的揭露資訊，以維持最高的合規程度。

必要的披露

- 您將透過程式傳送之訊息使用案例的說明。
- 短語「可能適用訊息和傳輸費率」。
- 指示收件者收到您的訊息的頻率。例如，週期性傳訊程式可能會顯示「每週一則訊息」。一次性密碼或多重要素驗證使用案例可能會顯示「訊息頻率不同」或「每次登入嘗試一則訊息」。
- 您的條款和條件以及隱私權政策文件連結。

不合規選擇加入的常見拒絕原因

- 如果提供的公司名稱不符合圖樣或螢幕擷取畫面中所提供的名稱。任何非明顯的關係都應在選擇加入工作流程描述中進行說明。

- 如果看起來訊息將傳送給收件者，但在執行此動作之前並未明確收集同意。明確同意是所有訊息傳遞的要求。
- 如果看起來需要接收簡訊才能註冊服務。如果工作流程不提供任何以其他形式 (例如電子郵件或語音通話) 接收選擇加入訊息的替代方法，則此選項不符合規定。
- 如果選擇加入的語言完全顯示在服務條款中。這些披露應始終在選擇加入時向收件人提交，而不是放在連結的政策文件中。
- 如果客戶同意接收來自您的某種類型的訊息，而您向他們傳送其他類型的簡訊。例如，他們同意接收一次性密碼，但也收到投票和調查訊息。
- 如果未向收件人提供所需的披露 (上面列出)。

下列範例符合行動電信業者對多重要素驗證使用案例的要求。

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*'. A 'Next >' button is at the bottom.

The second screen, also titled 'examplecorp', asks the user if they want to enable Multi-Factor Authentication (MFA). It offers two 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 offers three options: 'Email' (selected), 'Phone call', and 'Text message'. A note states: 'Message and data rates may apply. If you choose to receive MFA passwords as text messages, we'll send you one text message per login attempt. To stop receiving messages, text "STOP" to 98765. For more information, text "HELP."'. Links for 'Terms & Conditions' and 'Privacy Policy' are provided. A 'Mobile number' field is present, with a note: 'When you press the Next button, we'll send you an MFA password to verify your phone number.' A 'Next >' button is at the bottom.

A bracket on the right side of the third screen indicates: 'This section only appears when "Text message" is selected'.

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 a text message from '67876' with the content: 'Your ExampleCorp Multi-factor Authentication code is 918273. Text HELP for more info or STOP to opt out.' A 'Send' button is 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.' A 'Resend code' link is provided. Below the message is a numeric keypad with digits 1-9, 0, and symbols for backspace and call. 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.

多重驗證使用案例的模擬

它包含已完成的文字和圖像，並顯示完整的選擇加入流程，並帶有註釋。在選擇加入流程中，客戶必須採取明確的有意行動來表示同意接收簡訊，並包含所有必要的披露。

其他選擇加入工作流程類型

如果電信業者符合上述內容，則行動電信業者還將接受應用程式和網站之外的選擇加入工作流程，例如口頭或書面選擇加入。合規的選擇加入工作流程和口頭或書面腳本將收到收件人的明確同意，以接收特定的訊息類型。範例包括支援代理在錄製到服務資料庫之前用來收集同意的口頭指令碼，或是宣傳單上列出的電話號碼。若要提供這些選擇加入工作流程類型的模擬，您可以提供選擇加入指令碼、行銷資料或收集號碼的資料庫的螢幕擷取畫面。如果選擇加入不清楚或使用案例超過特定數量，行動電信業者可能會對這些使用案例有其他問題。

不要傳送至舊清單

人們經常更改電話號碼。您兩年前同意聯繫的電話號碼現在可能屬於其他人。不要在新的訊息傳遞程序中使用舊的電話號碼清單；如果這樣做，則可能會有一些訊息失敗，因為該號碼不再使用，有些人因為他們不記得首先徵求您的同意而選擇退出。

稽核您的客戶清單

如果您傳送週期性簡訊行銷活動，請定期稽核客戶清單。稽核客戶清單可確保僅有興趣接收訊息的客戶才會收到您的訊息。

當您稽核清單時，請傳送訊息給每個選擇接收的客戶來提醒他們已訂閱，並提供有關取消訂閱的資訊給他們。提醒訊息可能類似以下範例：

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply  
HELP for help, STOP to unsubscribe.
```

保留記錄

保留記錄，以顯示每個客戶何時請求接收您的簡訊，以及您傳送什麼訊息給每個客戶。世界各地許多國家和區域會要求簡訊寄件者以可輕鬆擷取的方式維護這些記錄。行動電信業者也可能隨時向您請求此資訊。您必須提供的確切資訊隨國家或區域而不同。如需有關記錄保留需求的詳細資訊，請檢閱客戶所在每個國家或區域的商業簡訊法規。

有時，電信業者或監管機構會要求我們證明客戶已選擇接收您的訊息。在這些情況下，支援會與您聯絡，提供電信業者或機構所需的資訊清單。如果您無法提供所需的資訊，我們可能會暫停您傳送更多簡訊。

訊息保持清晰、誠懇和簡潔

簡訊是一種獨特的媒介。每則訊息 160 個字元的限制，表示您的訊息必須簡潔。您可能在其他通訊管道 (例如電子郵件) 中使用的技巧可能不適用於 SMS 管道，並且在與簡訊搭配使用時，甚至可能看起來

不誠實或具欺騙性。如果訊息中的內容不符合最佳做法，收件人可能會忽略您的訊息；在最壞的情況下，行動電信業者可能會將您的訊息識別為垃圾郵件，並封鎖未來來自您電話號碼的訊息。

本節提供一些建立有效簡訊內文的提示和想法。

將您自己識別為寄件者

您的收件人應該能夠立即得知訊息來自您。遵循此最佳作法的寄件者會在每封郵件的開頭加上識別名稱(以下稱「程式名稱」)。

請勿執行此作業：

```
Your account has been accessed from a new device. Reply Y to confirm.
```

或改用這項優惠：

```
ExampleCorp Financial Alerts: You have logged in to your account from a  
new device. Reply Y to confirm, or STOP to opt-out.
```

不要試圖使您的訊息看起來像是個人訊息

有些行銷人員會想讓訊息看似來自個人，來為其簡訊添加個人風格。但是，此技術可能會使您的郵件看起來像是網路釣魚。

請勿執行此作業：

```
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.
```

或改用這項優惠：

```
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.
```

談論金錢時要小心

詐騙者經常利用人們想獲取金錢的慾望。不要讓報價看起來好的不像真的。不要用金錢的誘惑來欺騙人。不要使用貨幣符號來表示金錢。

請勿執行此作業：

```
Save big $$$ on your next car repair by going to https://  
www.example.com.
```


或改用這項優惠：

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.

僅使用必要的字元

品牌通常傾向於透過在其訊息中包含商標符號 (例如 ™ 或 ®) 來保護其商標。但是，這些符號不屬於可包含在 160 個字元簡訊中的標準字元集 (稱為 GSM 字母)。當您傳送包含這些字元之一的訊息時，會使用不同的字元編碼系統自動傳送郵件，每個訊息部分僅支援 70 個字元。因此，您的訊息可能分為幾個部分。由於您會針對傳送的每個訊息部分收費，因此傳送整封訊息所花費的成本可能會超過預期的費用。此外，您的收件者可能會收到來自您的數個連續訊息，而不是單一訊息。如需簡訊字元編碼的詳細資訊，請參閱 [Amazon SNS 中的簡訊字元限制](#)。

請勿執行此作業：

ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at example.com and use the promo code WIDGET.

或改用這項優惠：

ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at example.com and use the promo code WIDGET.

Note

前面的兩個範例幾乎相同，但第一個範例包含一個註冊商標符號 (®)，它不屬於 GSM 字母。因此，第一個範例會以兩則訊息的形式傳送，而第二個範例則以一則訊息傳送。

使用有效、安全的連結

如果您的訊息包含連結，請再次檢查這些連結以確定這些連結可運作。在公司網路外的裝置上測試連結，以確保連結能正確解析。由於簡訊的限制為 160 個字元，因此過長 URL 會分割為數則訊息。您應該使用重新導向網域來提供簡化的 URL。不過，您不應該使用免費連結縮短服務，如 tinyurl.com 或 bitly.com，因為電信業者傾向於篩選掉包含這些網域連結的訊息。不過，只要您的連結指向貴公司或組織專用的網域，您就可以使用付費連結縮短服務。

請勿執行此作業：

Go to <https://tinyurl.com/4585y8mr> today for a special offer!

或改用這項優惠：

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.

限制您使用的縮寫數

簡訊管道的 160 個字元限制會導致一些寄件者認為需要在訊息中廣泛使用縮寫。但是，對於許多讀者來說，過度使用縮寫看起來並不專業，並且可能導致某些使用者將您的訊息回報為垃圾郵件。其實可以在不使用過多縮寫的情況下編寫一致的訊息。

請勿執行此作業：

Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.

或改用這項優惠：

ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets at example.com. Text STOP to opt-out.

適當回應

當收件人回覆您的訊息時，請務必回覆有用的資訊。例如，當客戶以關鍵字「HELP」回應您的其中一個訊息時，請傳送有關他們所訂閱方案的相關資訊、您每個月將傳送的訊息數量，以及他們可聯絡您以取得更多資訊的方式。HELP 回應可能類似以下範例：

HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2 msgs/month. Msg & data rates may apply. Reply STOP to cancel.

當客戶以關鍵字「STOP」回覆時，請告知他們不會再收到任何進一步的訊息。STOP 回應可能類似以下範例：

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.

根據參與度來調整傳送

客戶的優先順序可能隨著時間而變更。如果客戶已覺得您的訊息沒有用處，他們可能選擇完全不要您的訊息，或甚至將您的訊息回報為來路不明。基於這些原因，您必須根據客戶參與度來調整傳送實務。

如果客戶很少與您的訊息互動，您應該調整訊息的頻率。例如，如果您傳送每週訊息給參與的客戶，您可以為較少參與的客戶建立單獨的每月摘要。

最後，從您的客戶清單中移除完全不參與的客戶。此步驟可避免客戶對您的訊息感到厭煩。還可節省您的成本，並協助保護您身為寄件者的評價。

適時傳送

只在正常白天上班時間傳送訊息。如果您在晚餐時間或午夜傳送訊息，客戶很可能會取消訂閱您的清單，以避免受到打擾。此外，當您的客戶無法立即回應時，則傳送簡訊也沒有意義。

如果您向非常大的受眾傳送行銷活動或旅程，請仔細檢查起始數量的輸送率。將收件者數量除以輸送率，以決定將訊息傳送給所有收件者所需的時間。

避免跨管道疲勞轟炸

在您的行銷活動中，如果您使用多個通訊管道 (例如電子郵件、簡訊和推送訊息)，請勿在每個管道中傳送相同的訊息。當您同時在多個管道中傳送相同的訊息時，客戶可能覺得您的傳送行為很討厭，而沒有用處。

使用專用短碼

如果您使用短碼，請為每個品牌和每個類型的訊息維護單獨的短碼。例如，如果您的公司有兩個品牌，請對每個品牌使用單獨的短碼。同樣地，如果您同時傳送交易訊息和促銷訊息，請對每種訊息使用單獨的短碼。若要進一步了解如何請求短碼，請參閱AWS End User Messaging SMS 《使用者指南》中的[使用 請求簡訊的短碼 AWS End User Messaging SMS](#)。

驗證您的目的地電話號碼

當您透過 Amazon SNS 傳送簡訊時，需按傳送的每個訊息部分向您收取費用。您根據訊息部分支付的價格，因收件人的國家或地區而異。如需簡訊定價的詳細資訊，請參閱[AWS 全球簡訊定價](#)。

當 Amazon SNS 接受傳送簡訊的請求時 (由於呼叫 [SendMessage](#) API，或由於促銷活動或旅程啟動的結果)，您需要支付傳送該訊息的費用。即使預定的收件者實際上並未收到訊息，此聲明也是正確的。例如，如果收件人的電話號碼已停止使用，或者您傳送訊息的電話號碼不是有效的行動電話號碼，您仍需支付傳送訊息的費用。

Amazon SNS 接受傳送簡訊的有效請求，並嘗試傳送簡訊。因此，您應驗證發送訊息的電話號碼是否為有效的手機號碼。您可以使用 AWS End User Messaging SMS 傳送測試訊息，以判斷電話號碼是否有效，以及電話號碼的類型（例如行動、有線或 VoIP）。如需詳細資訊，請參閱 AWS End User Messaging SMS 《使用者指南》中的[使用 SMS 模擬器傳送測試訊息](#)。

考量備援的設計

對於關鍵任務簡訊計劃，我們建議您在多個 AWS 區域中設定 Amazon SNS。Amazon SNS 提供數個 AWS 區域。如需可使用 Amazon SNS 的區域清單，請參閱[AWS 一般參考](#)。

您用於 SMS 訊息的電話號碼 (包括短碼、長碼、免費電話號碼和 10DLC 號碼) 無法跨 AWS 區域複製。因此，若要在多個區域使用 Amazon SNS，您必須在要使用 Amazon SNS 的每個區域中要求個別的電話號碼。例如，如果您使用短碼傳送文字訊息給美國收件人，則需要在您計劃使用的每個短碼中請求個別 AWS 區域的短碼。

在某些國家/地區，您也可以使用多種類型的電話號碼來增加備援。例如，在美國，您可以要求短碼、10DLC 號碼和免費電話號碼。這些電話號碼類型中的每一種都採用不同的路由到收件人。有多個電話號碼類型可用，無論是在相同 AWS 區域或分散在多個 AWS 區域，都會提供額外的備援層，這有助於改善彈性。

簡訊限制和規定

如需 SMS 限制和約束，請參閱 AWS End User Messaging SMS 《使用者指南》中的[SMS 和 MMS 限制和約束](#)。

管理退出關鍵字

SMS 收件者可以回覆關鍵字，以使用其裝置選擇退出訊息。如需詳細資訊，請參閱[停止接收簡訊](#)。

CreatePool

使用 CreatePool API 動作可建立新集區，並將指定的起始身分與集區關聯。如需詳細資訊，請參閱《AWS End User Messaging SMS API 參考》中的[CreatePool](#)。

PutKeyword

使用 PutKeyword API 動作在起始電話號碼或集區上建立或更新關鍵字組態。如需詳細資訊，請參閱《AWS End User Messaging SMS API 參考》中的[PutKeyword](#)。

管理號碼設定

若要管理您向 AWS Support 請求並指派給帳戶的專用短碼和長碼設定，請參閱[使用中的變更電話號碼的功能 AWS CLI](#)[AWS End User Messaging SMS](#)。

Amazon SNS 中的簡訊字元限制

單一簡訊最多可包含 140 個位元組的資訊。您在單一簡訊中可以包含的字元數，取決於訊息所包含的字元類型。

如果訊息只使用 [GSM 03.38 字元集的字元](#) (也稱為 GSM 7 位元字母)，則最多可包含 160 個字元。如果訊息包含 GSM 03.38 字元集以外的任何字元，則最多可有 70 個字元。當您傳送簡訊時，Amazon SNS 會自動決定最有效率的編碼來使用。

當訊息超過字元數上限時，訊息會分割成多個部分。當訊息分割成多個部分時，每個部分都會包含有關其前面訊息部分的其他資訊。當收件人的裝置收到以這種方式分隔的訊息部分時，它會使用這個其他資訊來確保所有訊息部分都以正確的順序顯示。根據收件人的行動電信業者和裝置，多則訊息可能會顯示為單一訊息或一系列個別訊息。因此，每個訊息部分的字元數會降至 153 個 (若訊息只包含 GSM 03.38 字元) 或 67 個 (若訊息包含其他字元)。您可以使用 SMS 長度計算器工具 (可在線上取得數個這類工具)，在傳送訊息之前，預估訊息中包含多少個訊息部分。任何訊息的最大支援大小為 1600 GSM 字元或 630 個非 GSM 字元。有關輸送量和訊息大小的更多資訊，請參閱《Amazon Pinpoint 使用者指南》中的 [Amazon Pinpoint 中的 SMS 字元限制](#)。

若要檢視您傳送的每則訊息的訊息部分數目，您應該先啟用 [事件串流設定](#)。當您這麼做時，Amazon SNS 會在訊息傳遞給收件人的行動服務供應商時產生 `_SMS.SUCCESS` 事件。`_SMS.SUCCESS` 事件記錄包含稱為 `attributes.number_of_message_parts` 的屬性。此屬性指定訊息所包含的訊息部分數目。

Important

當您傳送包含多個訊息部分的訊息時，將按訊息中包含的訊息部分數目向您收費。

GSM 03.38 字元集

下表列出 GSM 03.38 字元集的所有字元。如果您傳送的訊息只包含下表所示的字元，則訊息最多可包含 160 個字元。

GSM 03.38 標準字元												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
\$	=	!	>	#	-	ı	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

除了上表所示的字元，GSM 03.38 字元集還包含幾個符號。不過，這些字元還包含一個隱形的逸出字元，所以都以兩個字元計算：

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

最後，GSM 03.38 字元集還包含以下非列印字元：

- 空白字元。
- 換行控制，表示一行文字結束，並開始另一行。
- 歸位控制，移到一行文字的開頭 (通常接在換行字元後)。
- 逸出控制，自動新增到前面清單中的字元。

範例訊息

本節包含數則範例簡訊。針對每個範例，此區段會顯示字元總數，以及訊息的訊息部分數目。

範例 1：長訊息，只包含 GSM 03.38 字母表中的字元

下列訊息只包含 GSM 03.38 字母表中的字元。

```
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/bill11.
```

上述訊息包含 180 個字元，因此必須將其分割成多個訊息部分。當一則訊息分成多個訊息部分時，每個部分都可包含 153 個 GSM 03.38 字元。因此，此訊息會當成 2 個訊息部分傳送。

範例 2：包含多位元組字元的訊息

下列訊息包含幾個中文字元，所有這些字元都在 GSM 03.38 字母表之外。

```
#####.#####1994#7#####
```

上述訊息包含 71 個字元。不過，因為訊息中的所有字元幾乎都在 GSM 03.38 字母表之外，所以它會當成兩個訊息部分傳送。其中每個訊息部分最多可包含 67 個字元。

範例 3：包含單一非 GSM 字元的訊息

下列訊息包含不是 GSM 03.38 字母表一部分的單一字元。在此範例中，字元是結束單引號 (')，它是與一般撇號 (') 不同的字元。文書處理應用程式 (例如 Microsoft Word) 通常會自動以結束單引號取代撇號。如果您在 Microsoft Word 中撰寫簡訊草稿並將其貼入 Amazon SNS，則應該移除這些特殊字元，並以撇號取代之。

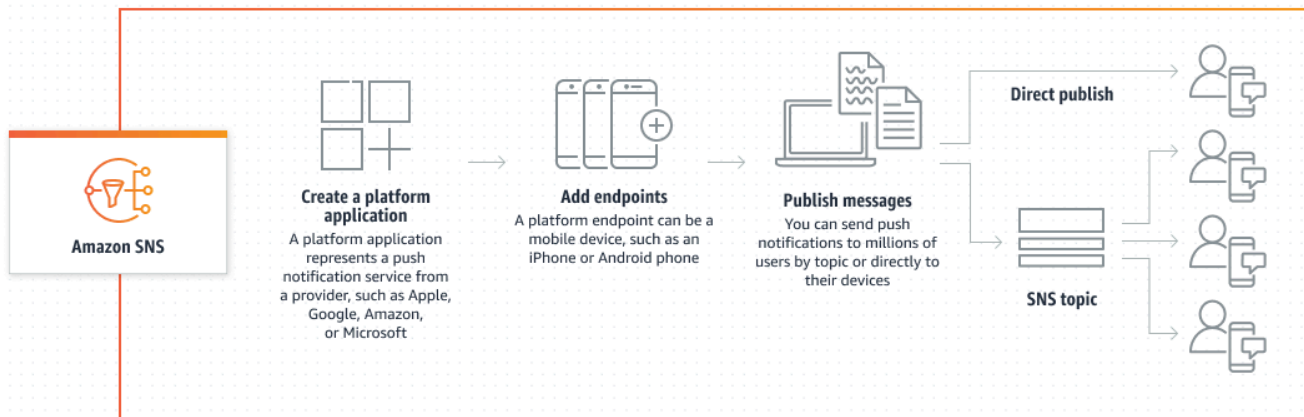
```
John: Your appointment with Dr. Salazar's office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

上述訊息包含 130 個字元。不過，由於它包含了不是 GSM 03.38 字母表一部分的結束單引號字元，因此它會當成兩個訊息部分傳送。

如果以撇號 (這是 GSM 03.38 字母表的一部分) 取代此訊息中的結束單引號字元，則該訊息會當成單一訊息部分傳送。

使用 Amazon SNS 傳送行動推播通知

您可以使用 Amazon SNS 將推播通知訊息直接傳送到行動裝置上的應用程式。傳送至行動端點的推播通知訊息，可能會在行動應用程式中顯示為訊息提醒、徽章更新或音效提醒。



主題

- [Amazon SNS 使用者通知的運作方式](#)
- [使用 Amazon SNS 設定推送通知](#)
- [在 Amazon SNS 中設定行動應用程式](#)
- [使用 Amazon SNS 傳送行動推播通知](#)
- [Amazon SNS 行動應用程式屬性](#)
- [行動應用程式的 Amazon SNS 應用程式事件通知](#)
- [行動推送 API 動作](#)
- [常見的 Amazon SNS 行動推送 API 錯誤](#)
- [使用 Amazon SNS 時間來即時訊息屬性，以進行行動推播通知](#)
- [Amazon SNS 行動應用程式支援的區域](#)
- [管理 Amazon SNS 行動推播通知的最佳實務](#)

Amazon SNS 使用者通知的運作方式

您使用以下其中一個支援的推送通知服務，傳送推送通知訊息至行動裝置和桌上型電腦。

- Amazon Device Messaging (ADM)
- 適用於 iOS 和 Mac OS X 的 Apple 推送通知服務 (APN)
- 百度雲端推送 (百度)
- Firebase Cloud Messaging (FCM)
- 適用於 Windows 手機的微軟推送通知服務 (MPNS)
- Windows 推送通知服務 (WNS)

諸如 APN 和 FCM 的推送通知服務，會保持與每個應用程式，以及已註冊使用其服務關聯行動裝置的連線。當應用程式和行動裝置註冊時，推送通知服務會傳回裝置字符。Amazon SNS 使用裝置字符來建立行動端點，您可傳送直接推送通知訊息至該端點。為使 Amazon SNS 與不同推送通知服務通訊，您會將推送通知服務憑證提交至要代表您使用的 Amazon SNS。如需詳細資訊，請參閱[使用 Amazon SNS 設定推送通知](#)。

除了直接傳送推送通知訊息外，您還可使用 Amazon SNS 來傳送訊息到已訂閱主題的行動端點。如[什麼是 Amazon SNS？](#)中所述，此概念與將其他端點類型 (例如 Amazon SQS、HTTP/S、電子郵件和簡訊) 訂閱到主題相同。差別是 Amazon SNS 使用推送通知服務通訊，以便訂閱的行動端點接收傳送至主題的通知訊息。

使用 Amazon SNS 設定推送通知

1. 如為您想要支援的行動平台，您必須先[取得登入資料和裝置字符](#)。
2. 使用登入資料以建立使用 Amazon SNS 的平台應用程式物件 (PlatformApplicationArn)。如需詳細資訊，請參閱[建立 Amazon SNS 平台應用程式](#)。
3. 使用傳回的登入資料，從推送通知服務為您的行動應用程式與裝置要求裝置字符。您收到的字符代表您的行動應用程式和裝置。
4. 使用裝置字符和 PlatformApplicationArn 來建立使用 Amazon SNS 的平台端點物件 (EndpointArn)。如需詳細資訊，請參閱[設定行動通知的 Amazon SNS 平台端點](#)。
5. 使用 EndpointArn 來[發佈訊息到行動裝置的應用程式](#)。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 [Direct Amazon SNS 行動裝置傳訊](#) 及 [發佈](#) API。

在 Amazon SNS 中設定行動應用程式

本主題說明如何 AWS Management Console 使用中所述的資訊在 中設定行動應用程式[Amazon SNS 使用者通知的先決條件](#)。

Amazon SNS 使用者通知的先決條件

若要開始使用 Amazon SNS 行動推播通知，您需要下列項目：

- 一組登入資料，用於連線到其中一個支援的推送通知服務：ADM、APN、百度、FCM、MPNS 或 WNS。
- 裝置字符或註冊 ID，用於行動應用程式和裝置。
- 已設定為傳送推送通知訊息到行動端點的 Amazon SNS。
- 已註冊和設定為使用其中一個支援的推送通知服務的行動應用程式。

將您的應用程式註冊推送通知服務需要執行數個步驟。Amazon SNS 需要您提供的一些資訊來推送通知服務，以便傳送直接的通知訊息至行動端點。一般而言，您需要用於連線到推送通知服務的必要憑證、從推送通知服務接收的裝置字符或註冊 ID (代表您的行動裝置和行動應用程式)，以及已註冊推送通知服務的行動應用程式。

憑證的確切形式在行動平台間有所不同，但是在每種狀況中，這些憑證都必須在連線到平台時提交。對於每個行動應用程式都會發行一組憑證，並且其必須用來傳送訊息到該應用程式的每個執行個體。

特定名稱會依所使用的推送通知服務而有所不同。例如，使用 APN 做為推送通知服務時，您需要裝置字符。或者，使用 FCM 時，對等的裝置字符稱為註冊 ID。裝置字符或註冊 ID 是由行動裝置的作業系統傳送至應用程式的字串。它可唯一識別在特定行動裝置上執行之行動應用程式的執行個體，並且可將被視為此應用程式與裝置配對的識別符。

Amazon SNS 將憑證 (加上幾個其他設定) 儲存為平台應用程式資源。裝置權杖 (同樣具有一些額外設定) 表示為稱為平台端點的物件。每個平台端點都屬於一個特定的平台應用程式，而每個平台端點也都能使用存放在其對應平台應用程式中的憑證進行通訊。

下列各節包括每個支援的推送通知服務的事前準備。取得先決條件資訊後，您可以使用 AWS Management Console 或 Amazon SNS 行動推播 APIs 傳送推播通知訊息。如需詳細資訊，請參閱[使用 Amazon SNS 設定推送通知](#)。

建立 Amazon SNS 平台應用程式

若要從 Amazon SNS 傳送通知至行動端點，無論是直接傳送或透過主題的訂閱傳送，您必須先建立平台應用程式。向註冊應用程式後 AWS，您需要為應用程式和行動裝置建立端點。此端點允許 Amazon SNS 傳送訊息至裝置。

建立平台應用程式

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選取推送通知。
3. 於 Platform applications (平台應用程式) 區段中，選擇 Create platform application (建立平台應用程式)。
4. 選擇您的 AWS 區域。如需可在建立行動應用程式的 AWS 區域清單，請參閱 [Amazon SNS 行動應用程式支援的區域](#)。
5. 輸入下列應用程式詳細資訊：
 - 應用程式名稱 – 為您的平台應用程式提供名稱。名稱必須介於 1 到 256 個字元之間，且可包含大寫字母和小寫字母、數字、底線、連字號和句點。
 - 推播通知平台 – 選取應用程式註冊的適當通知服務（例如 Apple Push Notification Service (APNs)、Firebase Cloud Messaging (FCM)）。
6. 根據您選取的平台，您需要提供特定登入資料：
 - 對於 APNs(Apple 推播通知服務) – 選擇字符型或憑證型身分驗證。
 - 對於字符型身分驗證，請上傳 .p8 檔案（透過 Keychain Access 產生）。
 - 對於憑證型身分驗證，請上傳 .p12 檔案（也從 Keychain Access 匯出）。
 - 針對 FCM (Firebase Cloud Messaging) – 從 Firebase 主控台輸入伺服器金鑰。
 - 對於其他平台（例如 ADM 或 GCM）– 輸入個別的 API 金鑰或登入資料。
7. 輸入必要的詳細資訊後，選擇建立平台應用程式。此動作向 Amazon SNS 註冊應用程式，並建立對應的平台應用程式物件。
8. 建立時，Amazon SNS 會產生並傳回 [PlatformApplicationArn](#)(Amazon Resource Name)。此 ARN 可唯一識別您的平台應用程式，並在為行動裝置建立端點時使用。

設定行動通知的 Amazon SNS 平台端點

當應用程式和行動裝置向推送通知服務（例如 APNs 或 Firebase Cloud Messaging）註冊時，推送通知服務會傳回裝置字符。Amazon SNS 使用此裝置字符來建立平台端點，做為傳送直接推送通知訊息至裝置上的應用程式的目標。平台端點可做為橋接器，將 Amazon SNS 傳送的訊息路由至推播通知服務，以傳送至對應的行動裝置。如需詳細資訊，請參閱 [Amazon SNS 使用者通知的先決條件](#) 和 [使用 Amazon SNS 設定推送通知](#)。

了解裝置字符和平台端點

裝置字符可唯一識別向推送通知服務註冊的行動裝置（例如 APNs、Firebase Cloud Messaging）。當應用程式向推送通知服務註冊時，會產生該應用程式和裝置專屬的裝置字符。Amazon SNS 使用此裝置字符在對應的平台應用程式中建立平台端點。

平台端點允許 Amazon SNS 透過推送通知服務傳送推送通知訊息至裝置，以維護應用程式與使用者裝置之間的連線。

建立平台端點

若要使用 Amazon SNS 將通知推送至應用程式，該應用程式的裝置字符首先必須透過呼叫建立平台端點動作向 Amazon SNS 註冊。此動作將平台應用程式的 Amazon 資源名稱 (ARN) 和裝置字符當作參數，並傳回所建立平台端點的 ARN。

[CreatePlatformEndpoint](#) 動作會執行下列動作：

- 如果平台端點已存在，請勿再次建立。將現有平台端點的 ARN 傳回呼叫者。
- 如果具有相同裝置權杖但不同設定的平台端點已存在，請勿再次建立它。將例外擲回呼叫者。
- 如果平台端點不存在，請建立它。將新建立平台端點的 ARN 傳回呼叫者。

您不應每次應用程式啟動時立即建立平台端點動作，因為此方法並不會一直提供運作中的端點。這可能會發生，例如應用程式在同一個裝置和端點上解除安裝並重新安裝時，因為其已存在但被停用。成功註冊程序應達成以下項目：

1. 確保平台端點存在，供此應用程式和裝置組合使用。
2. 確保平台端點中的裝置字符是最新的有效裝置字符。
3. 確保平台端點已啟用且隨時可供使用。

虛擬程式碼

以下虛擬程式碼說明在各種起始條件中建立可運作、最新的已啟用平台端點的建議實務。此方法不論這是否為應用程式首次註冊、此應用程式是否存在平台端點，以及平台端點是否啟用、具有正確的裝置字符等等，均有效用。可安全在一列中多次呼叫它，因為如果它已經是最新且啟用，不會建立重複的平台端點或變更現有平台端點。

```
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
endif
```

此方法在應用程式任何時候想要註冊或重新註冊時都可使用。也可用於通知 Amazon SNS 裝置字符變更事項。若是此狀況，您可以只呼叫具有最新字符值的動作。關於此方法有幾點要注意：

- 有兩種情況它可能呼叫建立平台端點動作。這可在最開始即應用程式不知道其自身的平台端點 ARN 時呼叫，在首次註冊期間發生。如果初始 `GetEndpointAttributes` 動作呼叫失敗，但沒有找到例外狀況，也會呼叫它，就像應用程式知道其端點 ARN 但已刪除一樣。
- 即使剛建立平台端點，也會呼叫 `GetEndpointAttributes` 動作來驗證平台端點的狀態。這會在平台端點已經存在但被停用時發生。在此情況下，建立平台端點動作會成功但不會啟用平台端點，所以您必須在傳回成功之前複查平台端點的狀態。

AWS SDK 範例

下列程式碼示範如何使用 AWS SDKs 提供的 Amazon SNS 用戶端實作先前的虛擬程式碼。

若要使用 AWS 開發套件，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS SDK 和工具參考指南中的 [共享的配置和認證文件](#)。

CLI

AWS CLI

建立平台應用程式端點

下列 `create-platform-endpoint` 範例會使用指定的字符，為指定的平台應用程式建立端點。

```
aws sns create-platform-endpoint \
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/MyApplication \
  --token EXAMPLE12345...
```

輸出：

```
{
  "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
}
```

如需詳細資訊，請參閱[行動推送 API 動作](#)。

故障診斷

反複使用過時的裝置字符呼叫建立平台端點

特別是 FCM 端點，您可能會認為最好存放應用程式發出的第一個裝置字符，然後在每次啟動應用程式時，使用該裝置字符呼叫建立平台端點。這可能看起來正確，因為應用程式不必管理裝置字符的狀態，Amazon SNS 會自動將裝置字符更新至其最新值。然而，此解決方案有幾個嚴重問題：

- Amazon SNS 仰賴 FCM 的回饋來將過期的裝置字符更新為新的裝置字符。FCM 有時會保留舊裝置字符的資訊，但非無限期。一旦 FCM 忘記舊裝置字符和新裝置字符的連線，Amazon SNS 就無法再將存放在平台端點中的裝置字符更新至其正確的值；只能改為停用平台端點。
- 平台應用程式將包含對應至相同裝置字符的多個平台端點。
- Amazon SNS 會對可以相同裝置字符建立的幾個平台端點強加配額。最後，建立新的端點將因為無效的參數例外狀況而失敗，並且出現下列錯誤訊息：「此端點已經註冊不同字符」。

如需管理 FCM 端點的詳細資訊，請參閱[Firebase Cloud Messaging 端點的 Amazon SNS 管理](#)。

重新啟用與無效裝置字符關聯的平台端點

當行動平台 (例如 APN 或 FCM) 通知 Amazon SNS 用於發佈要求的裝置字符無效，Amazon SNS 會停用與該裝置字符關聯的平台端點。然後 Amazon SNS 會拒絕後續發佈至該裝置字符。您可能會想最好只要重新啟用平台端點並持續發佈，但在大多數情況中，如此做並沒有作用：發佈的訊息不會傳遞而平台端點會在之後很快再次停用。

這是因為與平台端點關聯的裝置字符真的無效 傳遞到平台端點無法成功，因為其不再對應到任何安裝的應用程式。下次發佈時，行動平台將再次通知 Amazon SNS 裝置字符無效，Amazon SNS 會再次停用平台端點。

若要重新啟用已停用的平台端點，其需要與有效的裝置字符關聯 (使用一組端點屬性動作呼叫)，然後啟用。只有這樣傳遞到該平台端點才會成功。重新啟用平台端點而不更新其裝置字符，唯有在裝置字符與原本無效但再次生效的該端點關聯時才有作用。這可能會發生，例如應用程式在同一個行動裝置上解除安裝然後重新安裝，並且接收相同的裝置字符時。以上所述的方法，確保僅在確認裝置字符與其關聯後重新啟用平台端點，是最新可用的方法。

將裝置權杖與 Amazon SNS 整合以用於行動通知

當您第一次向通知服務註冊應用程式和行動裝置時，例如 Apple Push Notification Service (APNs) 和 Firebase Cloud Messaging (FCM)，服務會傳回裝置字符或註冊 IDs。這些字符/IDs 會新增至 Amazon SNS，以使用 [PlatformApplicationArn](#) API 為應用程式和裝置建立端點。建立端點後，[EndpointArn](#) 會傳回，Amazon SNS 會使用它將通知導向至正確的應用程式/裝置。

您可以透過下列方式將裝置字符或註冊 IDs 新增至 Amazon SNS：

- 透過 手動新增單一字符 AWS Management Console
- 使用 [CreatePlatformEndpoint](#) API 上傳多個字符
- 註冊未來裝置的字符

手動新增裝置字符或註冊 ID

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選取推送通知。
3. 在平台應用程式區段中，選取您的應用程式，然後選擇編輯。如果您尚未建立平台應用程式，請遵循 [建立 Amazon SNS 平台應用程式](#) 指南立即建立。
4. 選擇建立端點。

5. 在端點字符方塊中，輸入字符或註冊 ID，視您使用的通知服務而定（例如 FCM 註冊 ID）。
6. （選用）在使用者資料欄位中輸入其他資料。此資料必須為 UTF-8 編碼且小於 2 KB。
7. 選擇建立端點。

建立端點後，您可以直接傳送訊息至行動裝置或訂閱 Amazon SNS 主題的行動裝置。

使用 **CreatePlatformEndpoint** API 上傳多個字符

下列步驟說明如何使用提供的範例 Java 應用程式 (bulkupload 套件) AWS，將數個字符（裝置字符或註冊 IDs）上傳至 Amazon SNS。您可以使用此範本應用程式，來協助您開始上傳現有字符。

Note

下列步驟使用 Eclipse Java IDE。這些步驟假設您已安裝，適用於 Java 的 AWS SDK 而且您擁有 AWS 的安全登入資料 AWS 帳戶。如需詳細資訊，請參閱[適用於 Java 的 AWS SDK](#)。如需登入資料的詳細資訊，請參閱《IAM 使用者指南》中的[AWS 安全登入資料](#)。

1. 下載並解壓縮 [snsmobilepush.zip](#) 檔案。
2. 在 Eclipse 中建立新的 Java 專案，並將 SNSSamples 資料夾匯入專案。
3. 下載 [OpenCSV 程式庫](#)並將其新增至建置路徑。
4. 在 BulkUpload.properties 檔案中，指定下列項目：
 - 您的 ApplicationArn（平台應用程式 ARN）。
 - 包含字符的 CSV 檔案的絕對路徑。
 - 記錄成功和失敗字符的檔案名稱。例如，goodTokens.csv 和 badTokens.csv。
 - （選用）分隔符號、引號字元和要使用的執行緒數量的組態。

您完整的 BulkUpload.properties 看起來應與下列類似：

```
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. 執行 `BatchCreatePlatformEndpointSample.java` 應用程式，將權杖上傳至 Amazon SNS。成功上傳的權杖將登入 `goodTokens.csv`，而格式不正確的權杖將登入 `badTokens.csv`。

從裝置註冊權杖以供未來應用程式安裝

此程序有兩個選項：

使用 Amazon Cognito 服務

您的行動應用程式可以使用臨時安全登入資料來建立端點。建議 Amazon Cognito 產生臨時登入資料。如需詳細資訊，請參閱 [Amazon Cognito 開發人員指南](#)

若要追蹤應用程式[註冊](#)，請使用 Amazon SNS 事件在建立新的端點 ARNs時接收通知。

或者，您可以使用 [ListEndpointByPlatformApplication](#) API 來擷取已註冊端點的清單。

使用代理伺服器

如果您的應用程式基礎設施已在安裝時支援裝置註冊，您可以使用伺服器做為代理。它會透過 [CreatePlatformEndpoint](#) API 將裝置字符轉送至 Amazon SNS。

Amazon SNS 建立的端點 ARN 將會傳回，並且可由您的伺服器存放以供日後訊息發佈使用。

Amazon SNS Apple 推播通知身分驗證方法

您可以透過提供將您識別為應用程式開發人員的資訊，授權 Amazon SNS 將推送通知傳送到您的 iOS 或 macOS 應用程式。若要進行身分驗證，請在[建立平台應用程式時](#)，提供金鑰或憑證，您可以從 Apple 開發人員帳戶取得這兩項資料。

字符簽署金鑰

Amazon SNS 用來簽署 Apple 推送通知服務 (APNS) 身分驗證字符的私有簽署金鑰。

如果您提供簽署金鑰，Amazon SNS 將使用字符，針對您傳送的每個推送通知驗證 APN。您可以使用簽署金鑰，傳送推送通知至 APN 生產和沙盒環境。

簽署金鑰不會過期，您可以將相同的簽署金鑰用於多個應用程式。如需詳細資訊，請參閱 Apple 網站開發人員帳戶說明區段中的[使用身分驗證字符與 APNS 通訊](#)。

憑證

當您傳送推送通知時，Amazon SNS 用來對 APNS 進行身分驗證的 TLS 憑證。您可以從您的 Apple 開發人員帳戶取得憑證。

憑證將會在一年後過期。發生這種情況時，您必須建立新的憑證並將該憑證提供給 Amazon SNS。如需詳細資訊，請參閱 Apple 開發人員網站上的[建立與 APNS 的憑證型連線](#)。

使用 管理主控台 AWS 管理 APNs設定

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選取推送通知。
3. 在平台應用程式區段中，選取要編輯其 APNs設定的應用程式，然後選擇編輯。如果您尚未建立平台應用程式，請遵循[建立 Amazon SNS 平台應用程式](#)指南立即建立。
4. 選擇編輯以修改平台應用程式的設定。
5. 在驗證類型區段中，選擇下列其中一個選項：
 - 字符型身分驗證（建議用於現代 APNs整合）
 - 憑證型身分驗證（較舊的方法）
6. 根據身分驗證類型設定您的憑證：
 - 對於字符型身分驗證：
 - 上傳 .p8 檔案，這是您從 Apple 開發人員帳戶下載的身分驗證字符簽署金鑰。
 - 輸入您在 Apple 開發人員帳戶中找到的簽署金鑰 ID。導覽至憑證、IDs和設定檔、金鑰，然後選取您要使用的金鑰。
 - 從您的 Apple 開發人員帳戶提供團隊識別符。您可以在 成員資格 頁面上找到。
 - 輸入指派給應用程式的套件識別符。您可以在憑證、IDs和設定檔、應用程式 IDs。
 - 對於憑證型身分驗證：
 - 上傳 TLS 憑證的 .p12 檔案。從 Apple 開發人員帳戶下載憑證後，即可從 macOS 上的 Keychain Access 匯出此檔案。
 - 如果您已將密碼指派給 .p12 憑證，請在此處輸入密碼。
7. 輸入必要的登入資料後，請選擇儲存變更以更新設定。

Amazon SNS 與 Firebase Cloud Messaging 身分驗證設定整合

本主題說明如何從 Google 取得必要的 FCM API (HTTP v1) 登入資料，以搭配 AWS API AWS CLI 和使用 AWS Management Console。

⚠ Important

2024 年 3 月 26 日 – Amazon SNS 支援適用於 Apple 裝置和 Webpush 目的地的 FCM HTTP v1 API。建議您在 2024 年 6 月 1 日或之前將現有的行動推送應用程式遷移至最新的 FCM HTTP v1 API，以避免應用程式中斷。

2024 年 1 月 18 日 – Amazon SNS 推出對 FCM HTTP v1 API 的支援，以將行動推播通知交付至 Android 裝置。

2023 年 6 月 20 日 – Google 已棄用其 Firebase Cloud Messaging (FCM) 舊版 HTTP API。Amazon SNS 現在支援使用 FCM HTTP v1 API 交付至所有裝置類型。建議您在 2024 年 6 月 1 日或之前將現有的行動推送應用程式遷移至最新的 FCM HTTP v1 API，以避免中斷。

您可以透過提供將您識別為應用程式開發人員的資訊，授權 Amazon SNS 將推播通知傳送到您的應用程式。如要進行身份驗證，請在[建立平台應用程式時](#)提供 API 金鑰或權杖。您可以從[Firebase 應用程式主控台](#)取得下列資訊：

API 金鑰

API 金鑰是呼叫 Firebase 舊版 API 時使用的憑證。2024 年 6 月 20 日，Google 將刪除 FCM 舊版 API。如果您目前正在使用 API 金鑰作為平台憑證，則可以通過選擇權杖作為選項，並上傳 Firebase 應用程式的相關 JSON 檔案來更新平台憑據。

Token (字符)

呼叫 HTTP v1 API 時，會使用簡短的即時存取權杖。這是 Firebase 建議用來傳推送通知的 API。為了生成存取權杖，Firebase 以私有金鑰文件 (也稱為 service.json 檔案) 的形式提供給開發人員一組憑證。

先決條件

您必須先取得 FCM service.json 憑證，然後才能開始在 Amazon SNS 中管理 FCM 設定。如要獲取您的 service.json 憑證，請參閱 Google Firebase 文件中的[從舊版 FCM API 遷移到 HTTP v1](#)。

使用 CLI 管理 FCM 設定

您可以使用 AWS API 建立 FCM 推送通知。AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱《AWS 一般參考指南》中的[Amazon Simple Notification Service 端點和配額](#)。

建立 FCM 推送通知與 Amazon SNS 主題 (AWS API)

使用金鑰憑證時，PlatformCredential 是 API key。使用權杖憑證時，PlatformCredential 是 JSON 格式的私有金鑰檔案：

- [CreatePlatformApplication](#)

擷取現有 Amazon SNS 主題 (AWS API) 的 FCM 憑證類型

擷取憑證類型 "AuthenticationMethod": "Token" 或 "AuthenticationMethod": "Key"：

- [GetPlatformApplicationAttributes](#)

如要設置現有 Amazon SNS 主題的 FCM 屬性 (AWS API)

設置 FCM 屬性：

- [SetPlatformApplicationAttributes](#)

使用主控台管理 FCM 設定

您可以使用 AWS Command Line Interface (CLI) 建立 FCM 推送通知。AWS 帳戶中 Amazon SNS 資源的數量和大小有限。如需詳細資訊，請參閱 [Amazon Simple Notification Service 端點和配額](#)。

若要與 Amazon SNS 主題一起建立 FCM 推送通知 (AWS CLI)

使用金鑰憑證時，PlatformCredential 是 API key。使用權杖憑證時，PlatformCredential 是 JSON 格式的私有金鑰檔案。使用 AWS CLI 時，檔案必須為字串格式，且必須忽略特殊字元。若要正確格式化檔案，Amazon SNS 建議您使用下列命令：SERVICE_JSON=`jq @json <<< cat service.json`：

- [create-platform-application](#)

如要擷取現有 Amazon SNS 主題的 FCM 憑證類型 (AWS CLI)。

擷取憑證類型 "AuthenticationMethod": "Token" 或 "AuthenticationMethod": "Key"：

- [get-platform-application-attributes](#)

如要設置現有 Amazon SNS 主題的 FCM 屬性 (AWS CLI)

設置 FCM 屬性：

- [set-platform-application-attributes](#)

管理 FCM 設定 (主控台)

使用下列步驟在 Amazon SNS 中輸入和管理 Firebase Cloud Messaging (FCM) 登入資料。

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選取推送通知。
3. 在平台應用程式區段中，選取您要編輯其登入資料的 FCM 平台應用程式，然後選擇編輯。
4. 在 Firebase Cloud Messaging Credentials 區段中，選擇下列其中一個選項：
 - 字符型身分驗證（建議的方法）– 上傳您從 Firebase 主控台下載的私有金鑰檔案 (JSON)。此檔案包含為 FCM 通知產生短期存取字串所需的登入資料。若要取得此檔案：
 1. 前往 [Firebase 應用程式主控台](#)。
 2. 在專案設定中，選取雲端傳訊。
 3. 下載私有金鑰 JSON 檔案（用於字符型身分驗證方法）。
 - API 金鑰身分驗證 – 如果您偏好使用較舊的 API 金鑰身分驗證方法，請在提供的欄位中輸入 Google API 金鑰。若要取得此檔案：
 1. 前往 [Firebase 應用程式主控台](#)。
 2. 在專案設定中，選取雲端傳訊。
 3. 複製用於傳送通知的伺服器金鑰 (API 金鑰)。
5. 完成後，請選擇儲存變更。

相關主題

- [在 Amazon SNS 中使用 Google Firebase Cloud Messaging v1 承載](#)

Firebase Cloud Messaging 端點的 Amazon SNS 管理

管理和維護裝置字串

您可以遵循下列步驟，確保行動應用程式推播通知的可交付性：

1. 將所有裝置字符、對應的 Amazon SNS 端點 ARNs 和時間戳記存放在應用程式伺服器上。
2. 移除所有過時的字符，並刪除對應的 Amazon SNS 端點 ARNs。

在您應用程式的初始啟動時，您會收到裝置的裝置權杖（也稱為註冊權杖）。此裝置字符是由裝置的作業系統開採，並與您的 FCM 應用程式繫結。收到此裝置字符後，您可以將它註冊到 Amazon SNS 作為平台端點。建議您將裝置字符、Amazon SNS 平台端點 ARN 和時間戳記儲存到您的應用程式伺服器或其他持久性存放區。若要設定 FCM 應用程式以擷取和存放裝置字符，請參閱[擷取並存放 Google Firebase 文件中的註冊字符](#)。

請務必維護 up-to-date 字符。在下列情況下，您使用者的裝置字符可能會變更：

1. 行動應用程式會在新裝置上還原。
2. 使用者解除安裝或更新應用程式。
3. 使用者會清除應用程式資料。

當您的裝置字符變更時，建議您使用新的字符更新對應的 Amazon SNS 端點。這可讓 Amazon SNS 繼續與已註冊的裝置通訊。您可以在行動應用程式中實作下列虛擬程式碼來執行此操作。它描述了建立和維護已啟用平台端點的建議實務。此方法可以在每次行動應用程式啟動時執行，或在背景中做為排程任務執行。

虛擬程式碼

使用下列 FCM 虛擬程式碼來管理和維護裝置字符。

```
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
```

若要進一步了解權杖更新需求，請參閱 Google Firebase 文件中的[定期更新權杖](#)。

偵測無效的字符

當訊息以無效的裝置字符分派至 FCM v1 端點時，Amazon SNS 將收到下列其中一個例外狀況：

- UNREGISTERED (HTTP 404) – 當 Amazon SNS 收到此例外狀況時，您會收到 `FailureType` 的交付失敗事件 `InvalidPlatformToken`，而與端點相關聯的 `FailureMessage` 平台字符無效。當交付失敗時，Amazon SNS 會停用您的平台端點，但有此例外。
- INVALID_ARGUMENT (HTTP 400) – 當 Amazon SNS 收到此例外狀況時，表示裝置字符或訊息承載無效。如需詳細資訊，請參閱 Google Firebase 文件中的 [ErrorCode](#)。

由於 `INVALID_ARGUMENT` 可以在這些情況下傳回，因此 Amazon SNS 會傳回 `FailureType` 的 `InvalidNotification`，而通知內文 `FailureMessage` 的是無效的。當您收到此錯誤時，請確認您的承載正確。如果正確，請確認裝置字符是 up-to-date。當交付失敗時，Amazon SNS 不會停用您的平台端點，但有此例外狀況。

另一個您會遇到 `InvalidPlatformToken` 交付失敗事件的情況是，註冊的裝置字符不屬於嘗試傳送該訊息的應用程式。在此情況下，Google 會傳回 `SENDER_ID_MISMATCH` 錯誤。當交付失敗時，Amazon SNS 會停用您的平台端點，但有此例外。

當您為應用程式設定[交付狀態記錄](#)時，您可以在 CloudWatch 中使用從 FCM v1 API 收到的所有觀察到的錯誤代碼。

若要接收應用程式的交付事件，請參閱 [可用的應用程式事件](#)。

移除過時的字符

傳遞至端點裝置的訊息開始失敗時，字符會被視為過時。Amazon SNS 會將這些過時字符設定為平台應用程式的已停用端點。當您發佈至已停用的端點時，Amazon SNS 會傳回具有 `FailureType` 的 `EventDeliveryFailure` 事件 `EndpointDisabled`，且端點 `FailureMessage` 的已停用。若要接收應用程式的交付事件，請參閱 [可用的應用程式事件](#)。

當您從 Amazon SNS 收到此錯誤時，您需要移除或更新平台應用程式中的過時字符。

使用 Amazon SNS 傳送行動推播通知

本節說明使用如何傳送推送通知訊息到您的行動裝置。

發佈到主題

您也可以使用 Amazon SNS 向已訂閱主題的行動端點傳送訊息。如 [什麼是 Amazon SNS？](#) 中所述，此概念與將其他端點類型 (例如 Amazon SQS、HTTP/S、電子郵件和簡訊) 訂閱到主題相同。不同之處在於，Amazon SNS 透過通知服務，如 Apple Push Notification Service (APNS) 和 Google Firebase Cloud Messaging (FCM) 進行通訊。透過通知服務，以讓訂閱的行動端點接收傳送至主題的通知。

Direct Amazon SNS 行動裝置傳訊

您可將 Amazon SNS 推送通知訊息直接傳送到代表行動裝置上應用程式的端點。

傳送直達訊息

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板中，選擇 Push notifications (推送通知)。
3. 在 Mobile push notifications (行動裝置推送通知) 頁面上的 Platform applications (平台應用程式) 區段，選擇應用程式名稱，例如 **MyApp**。
4. 在 **MyApp** 頁面的 Endpoints (端點) 部分，請選擇端點，然後選擇 Publish message (發佈訊息)。
5. 在 Publish message to endpoint (發佈訊息至端點) 頁面上，輸入將顯示在行動裝置應用程式上的訊息，然後選擇 Publish message (發佈訊息)。

Amazon SNS 將通知訊息傳送至平台通知服務，該服務會再將訊息傳送至應用程式。

使用平台特定承載發佈 Amazon SNS 通知

您可以使用 AWS Management Console 或 Amazon SNS APIs，將具有平台特定承載的自訂訊息傳送至行動裝置。如需使用 Amazon SNS API 的詳細資訊，請參閱 [snsmobilepush.zip](#) 中的 [行動推送 API 動作](#) 與 SNSMobilePush.java 檔案。

傳送 JSON 格式訊息

當您傳送平台特定承載時，資料格式必須為 JSON 鍵值對字串，並搭配引號逸出。

以下範例說明 FCM 平台的自訂訊息。

```
{
  "GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
    \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}\"
}
```

傳送平台特定訊息

除了將自訂資料做為鍵值對傳送，您也可傳送平台特定的鍵值對。

以下範例顯示在 FCM data 參數中自訂資料鍵值對之後納入的 FCM 參數 `time_to_live` 及 `collapse_key`。

```
{
  "GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
    \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\": {\"time_to_live\": 3600, \"collapse_key\": \"deals\"}}}}\"
}
```

如需 Amazon SNS 中支援的每個推送通知服務所支援的鍵值對清單，請參閱：

Important

Amazon SNS 現在支援 Firebase Cloud Messaging (FCM) HTTP v1 API，可將行動推播通知傳送至 Android 裝置。

2024 年 3 月 26 日 – Amazon SNS 支援適用於 Apple 裝置和 Webpush 目的地的 FCM HTTP v1 API。我們建議您在 2024 年 6 月 1 日或之前將現有的行動推送應用程式遷移至最新的 FCM HTTP v1 API，以避免應用程式中斷。

- 在 APN 文件中的 [酬載金鑰參考](#)
- FCM 說明文件中的 [Firebase Cloud Messaging HTTP 通訊協定](#)
- 在 ADM 說明文件中 [傳送訊息](#)

在多個平台上向應用程式傳送訊息

若要為多個平台 (如 FCM 和 APN) 向裝置上安裝的應用程式傳送訊息，您必須先將行動裝置端點訂閱 Amazon SNS 中的某個主題，然後向該主題發佈訊息。

以下範例顯示要傳送到 APN、FCM 和 ADM 上已訂閱行動裝置端點的訊息：

```
{
  "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\"}}"}
}
```

將訊息做為提醒或背景通知傳送到 APN

Amazon SNS 可以將訊息傳送到 APN 作為 alert 或 background 通知 (如需詳細資訊，請參閱 APN 文件中的[將背景更新推送到您的應用程式](#))。

- alert APN 通知會顯示提醒訊息、播放音效，或將徽章新增到您應用程式的圖示，來通知使用者。
- background APN 通知會喚醒或指示您的應用程式來對通知內容採取行動，而不會通知使用者。

指定自訂 APN 標頭數值

建議您使用 Amazon SNS Publish API 動作、AWS SDKs 或指

定AWS.SNS.MOBILE.APNS.PUSH_TYPE預留訊息屬性的自訂值 AWS CLI。下列 CLI 範例會針對指定的主題將 content-available 設為 1，並將 apns-push-type 設為 background。

```
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

確保 JSON 結構有效。在每個鍵/值對之後新增逗號，最後一個鍵/值對除外。

從酬載推論 APN 推送類型標頭

如果您沒有設定 `apns-push-type` APN 標頭，依 `content-available` 金鑰而定，Amazon SNS 會在您 JSON 格式 APN 酬載組態的 `aps` 字典中，將標頭設為 `alert` 或 `background`。

 Note

雖然 `apns-push-type` 標頭可設為其他值，但 Amazon SNS 僅可推論 `alert` 或 `background` 標頭。

- `apns-push-type` 已設定為 `alert`
 - 如果 `aps` 字典僅包含設定為 `content-available` 的 1，且一個以上金鑰可觸發使用者互動。
 - 如果 `aps` 字典包含設定為 `content-available` 的 0，或者 `content-available` 金鑰不存在。
 - 如果 `content-available` 金鑰的值不是整數或布林值。
- `apns-push-type` 已設定為 `background`
 - 如果 `aps` 字典僅包含設定為 `content-available` 的 1，且沒有其他金鑰可觸發使用者互動。

 Important

如果 Amazon SNS 以僅限背景通知的方式傳送 APN 的原始組態物件，那麼您就必須在 `aps` 字典中加入設為 `content-available` 的 1。雖然您可以加入自定金鑰，`aps` 字典絕對不能包含可觸發使用者互動的任何金鑰 (例如：通知、徽章或聲音)。

以下是原始組態物件範例。

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

在此範例中，Amazon SNS 會將訊息的 apns-push-type APN 標頭設為 background。當 Amazon SNS 偵測到 apn 字典包含設定為 1 的 content-available 金鑰，且不包含任何其他設定標頭為可觸發使用者互動的金鑰時，會將標頭設為 background。

在 Amazon SNS 中使用 Google Firebase Cloud Messaging v1 承載

Amazon SNS 支援使用 FCM HTTP v1 API 將通知傳送至 Android、iOS 和 Webpush 目的地。本主題提供使用 CLI 或 Amazon SNS API 發佈行動推播通知時的承載結構範例。

傳送 FCM 通知時，您可以在承載中包含下列訊息類型：

- 資料訊息 – 資料訊息由用戶端應用程式處理，並包含自訂金鑰/值對。建構資料訊息時，您必須包含 JSON 物件做為值的 data 金鑰，然後輸入您的自訂金鑰值對。
- 通知訊息或顯示訊息 – 通知訊息包含一組由 FCM SDK 處理的預先定義金鑰。這些金鑰會根據您要交付的裝置類型而有所不同。如需平台特定通知金鑰的詳細資訊，請參閱下列內容：
 - [Android 通知金鑰](#)
 - [APNS 通知金鑰](#)
 - [Webpush 通知金鑰](#)

如需 FCM 訊息類型的詳細資訊，請參閱 Google Firebase 文件中的 [訊息類型](#)。

使用 FCM v1 承載結構傳送訊息

如果您是第一次建立 FCM 應用程式，或想要利用 FCM v1 功能，您可以選擇加入以傳送 FCM v1 格式的承載。若要這樣做，您必須包含最上層金鑰 fcmV1Message。如需建構 FCM v1 承載的詳細資訊，請參閱 Google Firebase 文件中的 [從舊版 FCM APIs 遷移至 HTTP v1](#) 和 [跨平台自訂訊息](#)。

傳送至 Amazon SNS 的 FCM v1 範例承載：

Note

使用 Amazon SNS 發佈通知時，下列範例中使用的 GCM 金鑰值必須編碼為字串。

```
{
  "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\": {

```

```
\nbadge\\": \"5\",  
  \n\"title\\\": \\\"string\\\",  
  \n\"body\\\": \\\"string\\\"  
},  
  \n\"data\\\": {  
    \n\"dataWeb\\\": \\\"priority message\\\"  
  }  
}  
}  
}  
}\"  
}
```

傳送 JSON 承載時，請務必在請求中包含 `message-structure` 屬性，並將其設定為 `json`。

CLI 範例：

```
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
```

如需傳送 FCM v1 格式承載的詳細資訊，請參閱 Google Firebase 文件中的以下內容：

- 從舊版 FCM APIs 遷移至 HTTP v1
- 關於 FCM 訊息
- REST 資源：`projects.messages`

使用舊版承載結構將訊息傳送至 FCM v1 API

遷移至 FCM v1 時，您不需要變更新版憑證的承載結構。Amazon SNS 會將您的承載轉換為新的 FCM v1 承載結構，並傳送至 Google。

輸入訊息承載格式：

 $\{$


```
"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\\\"}}"
}
```

傳送至 Google 的訊息：

```
{
  "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"
}
}
}

```

潛在風險

- 舊版 v1 映射不支援 Apple Push Notification Service (APNS) headers 或 fcm_options 金鑰。如果您想要使用這些欄位，請傳送 FCM v1 承載。
- 在某些情況下，FCM v1 需要訊息標頭，才能將靜音通知傳送至 APNs。如果您目前正在傳送無提示通知到您的 APNs 裝置，它們將無法與舊版方法搭配使用。反之，我們建議您使用 FCM v1 承載，以避免非預期的問題。若要尋找 APNs 標頭及其用途的清單，請參閱 Apple 開發人員指南中的 [與 APNs 通訊](#)。

- 如果您在傳送通知時使用 TTL Amazon SNS 屬性，則只會在 android 欄位中更新。如果您想要設定 TTL APNS 屬性，請使用 FCM v1 承載。
- android、apns 和 webpush 金鑰將映射，並填入提供的所有相關金鑰。例如，如果您提供 title，這是這三個平台之間共用的金鑰，FCM v1 映射會將您提供的標題填入這三個平台。
- 平台之間的某些共用金鑰預期值類型不同。例如，傳遞給的 badge 金鑰 apns 預期整數值，而傳遞給的 badge 金鑰 webpush 預期字串值。如果您提供 badge 金鑰，FCM v1 映射只會填入您提供有效值的金鑰。

FCM 交付失敗事件

下表提供 Amazon SNS 失敗類型，對應於從 Google for FCM v1 通知請求收到的錯誤/狀態碼。當您為應用程式設定 [交付狀態記錄](#) 時，您可以在 CloudWatch 中使用從 FCM v1 API 收到的所有觀察錯誤代碼。

FCM 錯誤/狀態碼	Amazon SNS 失敗類型	失敗訊息	原因和緩解措施
UNREGISTERED	InvalidPlatformToken	與端點相關聯的平台字符無效。	連接至端點的裝置字符已過時或無效。Amazon SNS 已停用您的端點。將 Amazon SNS 端點更新為最新的裝置字符。
INVALID_ARGUMENT	InvalidNotification	通知內文無效。	裝置字符或訊息承載可能無效。確認您的訊息承載有效。如果訊息承載有效，請將 Amazon SNS 端點更新為最新的裝置字符。
SENDER_ID_MISMATCH	InvalidPlatformToken	與端點相關聯的平台字符無效。	與裝置字符相關聯的平台應用程式沒有傳送至裝置字符的許可。確認您在 Amazon SNS 平台應用程式中

FCM 錯誤/狀態碼	Amazon SNS 失敗類型	失敗訊息	原因和緩解措施
			使用正確的 FCM 登入資料。
UNAVAILABLE	DependencyUnavailable	相依性不可用。	FCM 無法及時處理請求。Amazon SNS 執行的所有重試都失敗。您可以將這些訊息存放在無效字母佇列 (DLQ) 中，稍後再重新驅動。
INTERNAL	UnexpectedFailure	意外失敗；請聯絡 Amazon。失敗片語【內部錯誤】。	FCM 伺服器在嘗試處理您的請求時發生錯誤。Amazon SNS 執行的所有重試都失敗。您可以將這些訊息存放在無效字母佇列 (DLQ) 中，稍後再重新驅動。
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	平台應用程式登入資料無效。	無法傳送以 iOS 裝置或 Webpush 裝置為目標的訊息。確認您的開發和生產登入資料有效。
QUOTA_EXCEEDED	Throttled	請求由【gcm】調節。	已超過訊息速率配額、裝置訊息速率配額或主題訊息速率配額。如需如何解決此問題的資訊，請參閱 Google Firebase 文件中的 Error Code 。

FCM 錯誤/狀態碼	Amazon SNS 失敗類型	失敗訊息	原因和緩解措施
PERMISSION_DENIED	InvalidNotification	通知內文無效。	如果發生 PERMISSION_DENIED 例外狀況，發起人（您的 FCM 應用程式）沒有在承載中執行指定操作的許可。導覽至 FCM 主控台，並確認您的登入資料已啟用必要的 API 動作。

Amazon SNS 行動應用程式屬性

Amazon Simple Notification Service (Amazon SNS) 提供支援記錄推送通知訊息的傳遞狀態。在您設定應用程式屬性後，對於從 Amazon SNS 傳送至行動端點的訊息，日誌項目將會傳送至 CloudWatch Logs。記錄訊息傳遞狀態有助於提供深入的營運見解，例如下列項目：

- 得知推送通知訊息是否是從 Amazon SNS 傳送至推送通知服務。
- 識別從推送通知服務傳送至 Amazon SNS 的回應。
- 判定訊息暫留時間 (發佈時間戳記到就在遞交至推送通知服務前的時間)。

若要設定訊息傳遞狀態的應用程式屬性，您可以使用 AWS Management Console、AWS 軟體開發套件 (SDKs) 或查詢 API。

使用 設定訊息傳遞狀態屬性 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板中，選擇 Mobile (行動裝置)、Push notifications (推送通知)。
3. 從平台應用程式區段中，選擇應用程式，而此應用程式包含您想要接收 CloudWatch Logs 的端點。
4. 選擇 Application Actions (應用程式動作)，然後選擇 Delivery Status (交付狀態)。
5. 在 Delivery Status (交付狀態) 對話方塊中，選擇 Create IAM Roles (建立 IAM 角色)。

然後您會被重新引導至 IAM 主控台。

6. 選擇 Allow (允許) 以提供代表您使用 CloudWatch Logs 的 Amazon SNS 寫入存取權。
7. 現在，返回 Delivery Status (傳遞狀態) 對話方塊，在 Percentage of Success to Sample (0-100) (成功取樣的百分比 (0-100)) 欄位中針對您要接收 CloudWatch Logs 的成功傳送訊息的百分比輸入數字。

 Note

在您設定訊息傳遞狀態的應用程式屬性後，所有訊息傳送失敗的都會產生 CloudWatch Logs。

8. 最後，選擇 Save Configuration (儲存組態)。您現在可以檢視和剖析包含訊息傳遞狀態的 CloudWatch Logs。如需使用 CloudWatch 的詳細資訊，請參閱 [CloudWatch 文件](#)。

Amazon SNS 訊息傳遞狀態 CloudWatch Logs 範例

在您設定應用程式端點的訊息傳遞狀態屬性後，將會產生 CloudWatch Logs。JSON 格式的範例日誌顯示如下：

成功

```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
    "token": "ExampleIei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-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"
  }
}
```

失敗

```
{
  "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"
  }
}
```

如需推送通知服務回應代碼的清單，請參閱 [平台回應代碼](#)。

使用 AWS 開發套件設定訊息傳遞狀態屬性

[AWS 開發套件](#) 提供數種語言的 API，以搭配 Amazon SNS 使用訊息傳遞狀態屬性。

以下 Java 範例顯示如何使用 `SetPlatformApplicationAttributes` API，為推送通知訊息的訊息傳遞狀態設定應用程式屬性。您可以使用下列訊息傳遞狀態的屬性：`SuccessFeedbackRoleArn`、`FailureFeedbackRoleArn` 和 `SuccessFeedbackSampleRate`。`SuccessFeedbackRoleArn` 和 `FailureFeedbackRoleArn` 屬性是用來提供 Amazon SNS 寫入存取權，以代表您使用 CloudWatch Logs。`SuccessFeedbackSampleRate` 屬性用於指定成功傳送訊息的取樣率百分比 (0-100)。在您設定 `FailureFeedbackRoleArn` 屬性後，則所有傳送失敗的訊息都會產生 CloudWatch Logs。

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
    SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
```

```
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

如需適用於 Java 的開發套件詳細資訊，請參閱 [適用於 Java 的 AWS SDK 入門](#)。

平台回應代碼

以下是推送通知服務回應代碼的連結清單：

推送通知服務	回應代碼
Amazon Device Messaging (ADM)	請參閱 ADM 說明文件中的 回應格式 。
Apple Push Notification Service (APN)	請參閱本機與遠端通知程式設計指南的 與 APN 通訊 中的來自 APN 的 HTTP/2 回應。
Firebase Cloud Messaging (FCM)	請參閱 Firebase Cloud Messaging 文件中 下游訊息錯誤回應代碼 。
適用於 Windows 手機的微軟推送通知服務 (MPNS)	請參閱 Windows 8 開發文件的 Windows Phone 8 的推送通知服務回應代碼 。
Windows 推送通知服務 (WNS)	請參閱在 Windows 8 開發文件的 推送通知服務請求和回應標頭 (Windows Runtime Apps) 中的「回應代碼」。

行動應用程式的 Amazon SNS 應用程式事件通知

Amazon SNS 提供發生特定應用程式事件時觸發通知的支援。然後您可以對該事件採取一些程式設計動作。您的應用程式必須包含對於推送通知服務的支援，例如 Apple Push Notification Service (APN)、Firebase Cloud Messaging (FCM) 和 Windows 推送通知服務 (WNS)。您可以使用 Amazon SNS 主控台 AWS CLI 或 AWS SDKs 設定應用程式事件通知。

可用的應用程式事件

應用程式事件通知會在建立、刪除和更新個別平台端點，以及傳遞失敗時，進行追蹤。以下是應用程式事件的屬性名稱。

屬性名稱	通知觸發
EventEndpointCreated	新的平台端點已新增到您的應用程式。
EventEndpointDeleted	任何與您應用程式相關聯的平台端點已遭到刪除。
EventEndpointUpdated	任何與您應用程式相關聯的平台端點屬性已產生變更。
EventDeliveryFailure	傳遞到任何與您應用程式相關聯的平台端點時發生永久性失敗。  Note 若要追蹤平台應用程式端的交付失敗，請為應用程式的訂閱訊息傳遞狀態事件。如需詳細資訊，請參閱使用訊息傳遞狀態的 Amazon SNS 應用程式屬性。

您可以將任何屬性與應用程式建立關聯，讓應用程式接收這些事件通知。

傳送行動裝置推送通知

若要傳送應用程式事件通知，您可以指定主題接收每個事件類型的通知。當 Amazon SNS 傳送通知時，主題可將其路由到將採取程式設計動作的端點。

Important

高容量應用程式會建立大量的應用程式事件通知 (例如成千上萬)，而佔用供人使用的端點，例如電子郵件地址、電話號碼和行動應用程式。傳送應用程式事件通知到主題時請考慮下列準則：

- 接收通知的每個主題應僅包含程式設計端點的訂閱，例如 HTTP 或 HTTPS 端點、Amazon SQS 佇列或 AWS Lambda 函數。
- 若要減少由通知所觸發的處理次數，請將每個主題的訂閱數限制在較低的數字 (例如五個或更少)。

您可以使用 Amazon SNS 主控台、AWS Command Line Interface (AWS CLI) 或 AWS SDKs 傳送應用程式事件通知。

AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板中，選擇 Mobile (行動裝置)、Push notifications (推送通知)。
3. 在行動推播通知頁面的平台應用程式區段中，選擇應用程式，然後選擇編輯。
4. 展開 Event notifications (事件通知) 區段。
5. 選擇 Actions (動作)、Configure events (設定事件)。
6. 輸入要用於下列事件的主題 ARN：
 - 端點建立
 - 端點刪除
 - 端點更新
 - 傳遞失敗
7. 選擇 Save changes (儲存變更)。

AWS CLI

執行 [set-platform-application-attributes](#) 命令。

下面的範例為所有四個應用程式事件設定相同的 Amazon SNS 主題。

```
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

使用 AWS SDK 透過 Amazon SNS API 提交 `SetPlatformApplicationAttributes` 請求，以設定應用程式事件通知。

如需 AWS 開發套件開發人員指南和程式碼範例的完整清單，包括入門說明和先前版本的相關資訊，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。

行動推送 API 動作

若要使用 Amazon SNS 行動裝置推送 API，您必須先滿足推送通知服務 (例如 Apple Push Notification (APN) 和 Firebase Cloud Messaging (FCM)) 的事前準備。如需有關先決條件的詳細資訊，請參閱 [Amazon SNS 使用者通知的先決條件](#)。

若要使用 API 傳送推送通知訊息到行動應用程式和裝置，首先您必須使用會傳回 `PlatformApplicationArn` 屬性的 `CreatePlatformApplication` 動作。`PlatformApplicationArn` 屬性然後會被傳回 `EndpointArn` 屬性的 `CreatePlatformEndpoint` 使用。然後您可以使用 `EndpointArn` 屬性搭配 `Publish` 動作來傳送通知訊息到行動應用程式和裝置，或者您可以對於訂閱主題使用 `EndpointArn` 屬性和 `Subscribe` 動作。如需詳細資訊，請參閱 [使用 Amazon SNS 設定推送通知](#)。

Amazon SNS 行動推送 API 如下所示：

[CreatePlatformApplication](#)

針對裝置和行動應用程式可能註冊的其中一個支援的推送通知服務 (例如 APN 和 FCM)，建立平台應用程式物件。傳回 `CreatePlatformEndpoint` 動作所使用的 `PlatformApplicationArn` 屬性。

[CreatePlatformEndpoint](#)

為裝置和行動應用程式的其中一個支援的推送通知服務建立端點。`CreatePlatformEndpoint` 使用從 `CreatePlatformApplication` 動作傳回的 `PlatformApplicationArn` 屬性。使用 `CreatePlatformEndpoint` 時傳回之 `EndpointArn` 屬性會搭配使用 `Publish` 動作，來傳送通知訊息到行動應用程式和裝置。

[CreateTopic](#)

建立可發佈訊息的主題。

[DeleteEndpoint](#)

為裝置和行動應用程式的其中一個支援的推送通知服務刪除端點。

[DeletePlatformApplication](#)

刪除平台應用程式物件

[DeleteTopic](#)

刪除主題及其所有訂閱。

[GetEndpointAttributes](#)

為裝置和行動應用程式擷取端點屬性。

[GetPlatformApplicationAttributes](#)

擷取平台應用程式物件的屬性。

[ListEndpointsByPlatformApplication](#)

列出支援的推送通知服務中裝置和行動應用程式的端點和端點屬性。

[ListPlatformApplications](#)

列出支援的推送通知服務的平台應用程式物件。

[Publish](#)

傳送通知訊息到所有主題的訂閱端點。

[SetEndpointAttributes](#)

為裝置和行動應用程式設定端點的屬性。

[SetPlatformApplicationAttributes](#)

設定平台應用程式物件的屬性。

[Subscribe](#)

透過傳送端點確認訊息，來準備訂閱端點。若要實際建立訂閱，端點擁有者必須使用字符從確認訊息呼叫 `ConfirmSubscription` 動作。

[Unsubscribe](#)

刪除訂閱。

常見的 Amazon SNS 行動推送 API 錯誤

行動推送的 Amazon SNS API 所傳回的錯誤列於下表中。如需行動推送的 Amazon SNS API 的詳細資訊，請參閱 [行動推送 API 動作](#)。

錯誤	描述	HTTPS 狀態碼	API 動作
應用程式名稱為 null 字串	所需的應用程式名稱設定為 null。	400	CreatePlatformApplication
平台名稱為 null 字串	所需的平台名稱設定為 null。	400	CreatePlatformApplication
平台名稱無效	為平台名稱提供了無效或超出範圍的值。	400	CreatePlatformApplication
APN - 主體不是有效的憑證	提供給 APN 主體的憑證無效，其為 SSL 憑證。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 CreatePlatformApplication 。	400	CreatePlatformApplication
APN - 主體是有效的憑證，但不是 .pem 格式	提供給 APN 主體的有效憑證不是 .pem 格式，其為 SSL 憑證。	400	CreatePlatformApplication
APN - 主體是過期的憑證	提供給 APN 主體的憑證過期，其為 SSL 憑證。	400	CreatePlatformApplication
APN - 主體不是 Apple 發行的憑證	提供給 APN 主體的憑證不是 Apple 發行，其為 SSL 憑證。	400	CreatePlatformApplication
APN - 主體未提供	未提供 APN 主體，其為 SSL 憑證。	400	CreatePlatformApplication

錯誤	描述	HTTPS 狀態碼	API 動作
APN - 登入資料未提供	未提供 APN 登入資料，其為私密金鑰。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 CreatePlatformApplication 。	400	CreatePlatformApplication
APN - 登入資料不是有效的 .pem 格式	APN 登入資料不是有效的 .pem 格式，其為私密金鑰。	400	CreatePlatformApplication
FCM - 未提供 serverAPIKey	未提供 FCM 憑證 (API 金鑰)。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 CreatePlatformApplication 。	400	CreatePlatformApplication
FCM - serverAPIKey 為空白	FCM 憑證 (API 金鑰) 為空白。	400	CreatePlatformApplication
FCM - serverAPIKey 為 null 字串	FCM 憑證 (API 金鑰) 為 null。	400	CreatePlatformApplication
FCM - serverAPIKey 無效	FCM 憑證 (API 金鑰) 無效。	400	CreatePlatformApplication
ADM - clientsecret 未提供	所需的用戶端密碼未提供。	400	CreatePlatformApplication

錯誤	描述	HTTPS 狀態碼	API 動作
ADM - clientsecret 為 null 字串	用戶端密碼所需的字串為 null。	400	CreatePlatformApplication
ADM - client_secret 為空白字串	用戶端密碼所需的字串為空白。	400	CreatePlatformApplication
ADM - client_secret 無效	用戶端密碼所需的字串無效。	400	CreatePlatformApplication
ADM - client_id 為空白字串	用戶端 ID 所需的字串為空白。	400	CreatePlatformApplication
ADM - clientId 未提供	用戶端 ID 所需的字串未提供。	400	CreatePlatformApplication
ADM - clientid 為 null 字串	用戶端 ID 所需的字串為 null。	400	CreatePlatformApplication
ADM - client_id 無效	用戶端 ID 所需的字串無效。	400	CreatePlatformApplication
EventEndpointCreated 具有無效的 ARN 格式	EventEndpointCreated 具有無效的 ARN 格式。	400	CreatePlatformApplication
EventEndpointDeleted 具有無效的 ARN 格式	EventEndpointDeleted 具有無效的 ARN 格式。	400	CreatePlatformApplication

錯誤	描述	HTTPS 狀態碼	API 動作
EventEndpointUpdated 具有無效的 ARN 格式	EventEndpointUpdated 具有無效的 ARN 格式。	400	CreatePlatformApplication
EventDeliveryAttemptFailure 具有無效的 ARN 格式	EventDeliveryAttemptFailure 具有無效的 ARN 格式。	400	CreatePlatformApplication
EventDeliveryFailure 具有無效的 ARN 格式	EventDeliveryFailure 具有無效的 ARN 格式。	400	CreatePlatformApplication
EventEndpointCreated 不是現有的主題	EventEndpointCreated 不是現有的主題。	400	CreatePlatformApplication
EventEndpointDeleted 不是現有的主題	EventEndpointDeleted 不是現有的主題。	400	CreatePlatformApplication
EventEndpointUpdated 不是現有的主題	EventEndpointUpdated 不是現有的主題。	400	CreatePlatformApplication
EventDeliveryAttemptFailure 不是現有的主題	EventDeliveryAttemptFailure 不是現有的主題。	400	CreatePlatformApplication
EventDeliveryFailure 不是現有的主題	EventDeliveryFailure 不是現有的主題。	400	CreatePlatformApplication
平台 ARN 無效	平台 ARN 無效。	400	SetPlatformAttributes
平台 ARN 有效但不屬於使用者	平台 ARN 有效但不屬於使用者。	400	SetPlatformAttributes

錯誤	描述	HTTPS 狀態碼	API 動作
APN - 主體不是有效的憑證	提供給 APN 主體的憑證無效，其為 SSL 憑證。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 CreatePlatformApplication 。	400	SetPlatformAttributes
APN - 主體是有效的憑證，但不是 .pem 格式	提供給 APN 主體的有效憑證不是 .pem 格式，其為 SSL 憑證。	400	SetPlatformAttributes
APN - 主體是過期的憑證	提供給 APN 主體的憑證過期，其為 SSL 憑證。	400	SetPlatformAttributes
APN - 主體不是 Apple 發行的憑證	提供給 APN 主體的憑證不是 Apple 發行，其為 SSL 憑證。	400	SetPlatformAttributes
APN - 主體未提供	未提供 APN 主體，其為 SSL 憑證。	400	SetPlatformAttributes
APN - 登入資料未提供	未提供 APN 登入資料，其為私密金鑰。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 CreatePlatformApplication 。	400	SetPlatformAttributes
APN - 登入資料不是有效的 .pem 格式	APN 登入資料不是有效的 .pem 格式，其為私密金鑰。	400	SetPlatformAttributes

錯誤	描述	HTTPS 狀態碼	API 動作
FCM - 未提供 serverAPIKey	未提供 FCM 憑證 (API 金鑰)。如需詳細資訊，請參閱 Amazon Simple Notification Service API 參考中的 CreatePlatformApplication 。	400	SetPlatformAttributes
FCM - serverAPIKey 為 null 字串	FCM 憑證 (API 金鑰) 為 null。	400	SetPlatformAttributes
ADM - clientId 未提供	用戶端 ID 所需的字串未提供。	400	SetPlatformAttributes
ADM - clientId 為 null 字串	用戶端 ID 所需的字串為 null。	400	SetPlatformAttributes
ADM - clientsecret 未提供	所需的用戶端密碼未提供。	400	SetPlatformAttributes
ADM - clientsecret 為 null 字串	用戶端密碼所需的字串為 null。	400	SetPlatformAttributes
EventEndpointUpdated 具有無效的 ARN 格式	EventEndpointUpdated 具有無效的 ARN 格式。	400	SetPlatformAttributes
EventEndpointDeleted 具有無效的 ARN 格式	EventEndpointDeleted 具有無效的 ARN 格式。	400	SetPlatformAttributes
EventEndpointUpdated 具有無效的 ARN 格式	EventEndpointUpdated 具有無效的 ARN 格式。	400	SetPlatformAttributes

錯誤	描述	HTTPS 狀態碼	API 動作
EventDeliveryAttemptFailure 具有無效的 ARN 格式	EventDeliveryAttemptFailure 具有無效的 ARN 格式。	400	SetPlatformAttributes
EventDeliveryFailure 具有無效的 ARN 格式	EventDeliveryFailure 具有無效的 ARN 格式。	400	SetPlatformAttributes
EventEndpointCreated 不是現有的主題	EventEndpointCreated 不是現有的主題。	400	SetPlatformAttributes
EventEndpointDeleted 不是現有的主題	EventEndpointDeleted 不是現有的主題。	400	SetPlatformAttributes
EventEndpointUpdated 不是現有的主題	EventEndpointUpdated 不是現有的主題。	400	SetPlatformAttributes
EventDeliveryAttemptFailure 不是現有的主題	EventDeliveryAttemptFailure 不是現有的主題。	400	SetPlatformAttributes
EventDeliveryFailure 不是現有的主題	EventDeliveryFailure 不是現有的主題。	400	SetPlatformAttributes
平台 ARN 無效	平台 ARN 無效。	400	GetPlatformApplicationAttributes
平台 ARN 有效但不屬於使用者	平台 ARN 有效但不屬於使用者。	403	GetPlatformApplicationAttributes
指定的字符無效	指定的字符無效。	400	ListPlatformApplications

錯誤	描述	HTTPS 狀態碼	API 動作
平台 ARN 無效	平台 ARN 無效。	400	ListEndpointsByPlatformApplication
平台 ARN 有效但不屬於使用者	平台 ARN 有效但不屬於使用者。	404	ListEndpointsByPlatformApplication
指定的字符無效	指定的字符無效。	400	ListEndpointsByPlatformApplication
平台 ARN 無效	平台 ARN 無效。	400	DeletePlatformApplication
平台 ARN 有效但不屬於使用者	平台 ARN 有效但不屬於使用者。	403	DeletePlatformApplication
平台 ARN 無效	平台 ARN 無效。	400	CreatePlatformEndpoint
平台 ARN 有效但不屬於使用者	平台 ARN 有效但不屬於使用者。	404	CreatePlatformEndpoint
未指定字符	未指定字符。	400	CreatePlatformEndpoint
字符不是正確的長度	字符不是正確的長度。	400	CreatePlatformEndpoint

錯誤	描述	HTTPS 狀態碼	API 動作
客戶使用者資料過大	客戶使用者資料使用 UTF-8 編碼時長度不可超過 2048 位元組。	400	CreatePlatformEndpoint
端點 ARN 無效	端點 ARN 無效。	400	DeleteEndpoint
端點 ARN 有效但不屬於使用者	端點 ARN 有效但不屬於使用者。	403	DeleteEndpoint
端點 ARN 無效	端點 ARN 無效。	400	SetEndpointAttributes
端點 ARN 有效但不屬於使用者	端點 ARN 有效但不屬於使用者。	403	SetEndpointAttributes
未指定字符	未指定字符。	400	SetEndpointAttributes
字符不是正確的長度	字符不是正確的長度。	400	SetEndpointAttributes
客戶使用者資料過大	客戶使用者資料使用 UTF-8 編碼時長度不可超過 2048 位元組。	400	SetEndpointAttributes
端點 ARN 無效	端點 ARN 無效。	400	GetEndpointAttributes
端點 ARN 有效但不屬於使用者	端點 ARN 有效但不屬於使用者。	403	GetEndpointAttributes
目標 ARN 無效	目標 ARN 無效。	400	Publish
目標 ARN 有效但不屬於使用者	目標 ARN 有效但不屬於使用者。	403	Publish
訊息格式無效	訊息格式無效。	400	Publish

錯誤	描述	HTTPS 狀態碼	API 動作
訊息大小大於協定/終端服務所支援的大小	訊息大小大於協定/終端服務所支援的大小。	400	Publish

使用 Amazon SNS 時間來即時訊息屬性，以進行行動推播通知

Amazon Simple Notification Service (Amazon SNS) 支援為行動推送通知訊息設定存留時間 (TTL) 訊息屬性。除了在支援此功能的行動推播通知服務 Amazon SNS 訊息內文中設定 TTL 的現有功能之外，例如 Amazon Device Messaging (ADM) 和 Firebase Cloud Messaging (FCM)，在傳送至 Android 時。

TTL 訊息屬性用於指定有關訊息的過期中繼資料。這可讓您指定推送通知服務 (例如 Apple Push Notification Service (APN) 或 FCM) 傳遞訊息到端點所需的時間量。如果基於某些理由 (例如行動裝置電源關閉)，訊息在指定的 TTL 之內無法傳送，則系統會放棄訊息且不會再嘗試傳送訊息。若要在訊息屬性中指定 TTL，您可以使用 AWS Management Console AWS 軟體開發套件 (SDKs) 或查詢 API。

推送通知服務的 TTL 訊息屬性

以下是推送通知服務的 TTL 訊息屬性清單，您可以在使用 AWS SDKs 或查詢 API 時用來設定：

推送通知服務	TTL 訊息屬性
Amazon Device Messaging (ADM)	<code>AWS.SNS.MOBILE.ADM.TTL</code>
Apple Push Notification Service (APN)	<code>AWS.SNS.MOBILE.APNS.TTL</code>
Apple Push Notification Service Sandbox (APNs_SANDBOX)	<code>AWS.SNS.MOBILE.APNS_SANDBOX.TTL</code>
百度雲端推送 (百度)	<code>AWS.SNS.MOBILE.BAIDU.TTL</code>
Firebase Cloud Messaging (傳送至 Android 時為 FCM)	<code>AWS.SNS.MOBILE.FCM.TTL</code>
Windows 推送通知服務 (WNS)	<code>AWS.SNS.MOBILE.WNS.TTL</code>

每個推送通知服務均會以不同方式處理 TTL。Amazon SNS 提供了涵蓋所有推送通知服務的 TTL 摘要視圖，使您能夠更方便地指定 TTL。當您使用 AWS Management Console 指定 TTL（以秒為單位）時，您只需輸入一次 TTL 值，Amazon SNS 就會在發佈訊息時計算每個所選推播通知服務的 TTL。

TTL 與發佈時間相對。在將推送通知訊息遞交至特定推送通知服務時，Amazon SNS 會為推送通知計算暫留時間（介於發佈時間戳記與遞交至推送通知服務前之間的時間），並將剩餘 TTL 傳送到特定的推送通知服務。如果 TTL 短於暫留時間，Amazon SNS 就不會嘗試發佈。

如果您為推送通知訊息指定 TTL，則 TTL 值必須是正整數，除非的值對推送通知服務具有特定意義，例如使用 APNs 和 FCM（當傳送至 Android 時）。如果 TTL 值設定為 0，且推送通知服務對 0 沒有特定含意，則 Amazon SNS 不會放棄該訊息。如需有關在使用 APN 時將 TTL 參數設為 0 的詳細資訊，請參閱 Binary Provider API 說明文件中的[表格 A-3 遠端通知適用的項目識別碼](#)。

決定 TTL 的優先順序

Amazon SNS 根據以下優先順序決定推送通知訊息的 TTL，其中最小數字具有最優先順序：

1. 訊息屬性 TTL
2. 訊息內文 TTL
3. 推送通知服務預設 TTL (每個服務有所不同)
4. Amazon SNS 預設 TTL (4 週)

如果您為同一則訊息設定不同的 TTL 值（一個值位於訊息屬性，另一個位於訊息內文），則 Amazon SNS 會修改訊息內文中的 TTL，以和訊息屬性中指定的 TTL 相符。

使用指定 TTL AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板中，選擇 Mobile (行動裝置)、Push notifications (推送通知)。
3. 在 Mobile push notifications (行動裝置推送通知) 頁面上，在 Platform applications (平台應用程式) 區段內，選擇應用程式。
4. 在 **MyApplication** (我的應用程式) 頁面上，於 Endpoints (端點) 區段內，選擇應用程式端點，然後選擇 Publish message (發佈訊息)。
5. 在 Message details (訊息詳細資訊) 區段，輸入 TTL (推送通知服務需將訊息傳遞至端點的秒數)。
6. 選擇 Publish message (發佈訊息)。

Amazon SNS 行動應用程式支援的區域

您目前只可以在以下區域中建立行動應用程式：

- 美國東部 (俄亥俄)
- 美國東部 (維吉尼亞北部)
- 美國西部 (加利佛尼亞北部)
- 美國西部 (奧勒岡)
- 非洲 (開普敦)
- Asia Pacific (Hong Kong)
- 亞太區域 (雅加達)
- 亞太區域 (孟買)
- 亞太區域 (大阪)
- 亞太區域 (首爾)
- 亞太區域 (新加坡)
- 亞太區域 (雪梨)
- 亞太區域 (東京)
- 加拿大 (中部)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國西部)

管理 Amazon SNS 行動推播通知的最佳實務

本節描述可能有助於您提升客戶參與度的最佳實務。

端點管理

交付問題可能發生在下列情況中：使用者因裝置動作（例如，在裝置上重新安裝應用程式）而變更裝置權杖，或[憑證更新](#)影響在特定 iOS 版本上執行的裝置。Apple 推薦的最佳實務是在您的應用程式每次啟動時[註冊](#) APN。

由於裝置字符不會在每次使用者開啟應用程式時變更，因此可以使用等冪 [CreatePlatformEndpoint](#) API。不過，如果字符本身無效，或者端點有效但已停用（例如生產和沙盒環境不相符），這可能會為相同的裝置引入重複項目。

可以使用裝置字符管理機制，例如[虛擬程式碼](#)。

如需管理和維護 FCM v1 裝置字符的資訊，請參閱 [Firebase Cloud Messaging 端點的 Amazon SNS 管理](#)。

交付狀態記錄

若要監控推送通知交付狀態，我們建議您為 Amazon SNS 平台應用程式啟用交付狀態記錄。這有助於您疑難排解交付失敗問題，因為日誌包含從推送平台服務傳回的提供者[回應代碼](#)。如需啟用交付狀態記錄的詳細資訊，請參閱[如何存取 Amazon SNS 主題交付日誌以取得推送通知？](#)。

事件通知

若要以事件驅動的方式管理端點，您可以使用[事件通知](#)功能。這樣，設定的 Amazon SNS 主題就可以將事件擴散給訂閱者（例如 Lambda 函數），以處理端點建立、刪除、更新和交付失敗的平台應用程式事件。

Amazon SNS 電子郵件訂閱設定和管理

您可以使用 AWS Management Console 適用於 Java 的 AWS SDK 或來訂閱[電子郵件地址](#)至 Amazon SNS 主題 適用於 .NET 的 AWS SDK。

備註

- 不支援自訂電子郵件訊息內文。電子郵件傳遞功能旨在提供內部系統警示，而非行銷訊息。
- 僅標準主題支援直接訂閱電子郵件端點。
- 電子郵件交付輸送量會受到調節。如需詳細資訊，請參閱 [Amazon SNS 配額](#)。

⚠ Important

如要禁止郵寄清單收件人取消訂閱 Amazon SNS 主題電子郵件的所有收件人，請前往 AWS Support 參閱[設定須通過身分驗證才能取消訂閱的電子郵件訂閱](#)。

使用 訂閱電子郵件地址至 Amazon SNS 主題 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 在左導覽窗格中，選擇訂閱。
3. 在訂閱頁面，選擇建立訂閱。
4. 在建立訂閱頁面上，於詳細資訊區段中，執行以下作業：
 - a. 對於ARN 主題，選擇主題的 Amazon Resource Name (ARN)。
 - b. 對於通訊協定，選擇電子郵件。
 - c. 針對 Endpoint (端點)，輸入您的電子郵件位址。
 - d. (選用) 若要設定篩選政策，請展開 Subscription filter policy (訂閱篩選政策) 區段。如需詳細資訊，請參閱[Amazon SNS 訂閱篩選政策](#)。
 - e. (選用) 若要啟用以承載為基礎的篩選，請將 Filter Policy Scope 設定為 MessageBody。如需詳細資訊，請參閱[Amazon SNS 訂閱篩選政策範圍](#)。
 - f. (選用) 若要設定訂閱的無效字母佇列，請展開 Redrive policy (dead-letter queue) (重新磁碟機政策 (無效字母佇列)) 區段。如需詳細資訊，請參閱[Amazon SNS 無效字母佇列](#)。
 - g. 選擇建立訂閱。

主控台會建立訂閱並開啟訂閱的 Details (詳細資訊) 頁面

您必須確認訂閱，電子郵件地址才會開始接收訊息。

確認訂閱

1. 檢查您的電子郵件收件匣並選擇確認訂閱 Amazon SNS 發送的電子郵件。
2. Amazon SNS 會開啟您的 web 瀏覽器，並顯示含有您的訂閱 ID 的訂閱確認。

使用 AWS SDK 訂閱電子郵件地址至 Amazon SNS 主題

若要使用 AWS 開發套件，您必須使用 登入資料進行設定。如需詳細資訊，請參閱 AWS SDK和工具參考指南中的[共享的配置和認證文件](#)。

下列程式碼範例示範如何使用 Subscribe。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#)中設定和執行。

訂閱主題的電子郵件地址。

```
/// <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;
    }
```

使用篩選條件訂閱主題的佇列。

```
/// <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;
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的[訂閱](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
//! 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();
}

```

訂閱行動應用程式至主題。

```

//! 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();
}
```

訂閱 Lambda 函數至主題。

```
//! 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();
}
```

```
}

```

訂閱主題的 SQS 佇列。

```

    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;
    }

```


使用篩選條件訂閱主題。

```

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;
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的[訂閱](#)。

CLI

AWS CLI

訂閱主題

下列 `subscribe` 命令會將電子郵件地址訂閱至指定的主題。

```
aws sns subscribe \
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
  --protocol email \
  --notification-endpoint my-email@example.com
```

輸出：

```
{
  "SubscriptionArn": "pending confirmation"
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的[Subscribe](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

使用篩選條件訂閱主題的佇列。

```
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
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的[訂閱](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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);
    }
}
}

```

訂閱 HTTP 端點至主題。

```

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);
    }
}
}

```

訂閱 Lambda 函數至主題。

```
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 "";
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的[訂閱](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 API。

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

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 * a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 * topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

訂閱行動應用程式至主題。

```

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;
};

```

訂閱 Lambda 函數至主題。

```

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;
};

```

訂閱主題的 SQS 佇列。

```

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;
};

```

使用篩選條件訂閱主題。

```

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;
};

```

- 如需詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/sns-examples-managing-topics.html#sns-examples-subscribing-email>。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的[訂閱](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```

suspend fun subEmail(
    topicArnVal: String,
    email: String,

```



```
) : String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

訂閱 Lambda 函數至主題。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[訂閱](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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());  
}
```

訂閱 HTTP 端點至主題。

```
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());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[訂閱](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[訂閱](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```

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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-create-subscription.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的[訂閱](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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(())
}

```

- 如需 API 詳細資訊，請參閱適用於 Rust API 的AWS SDK 參考中的[訂閱](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```

TRY.
    oo_result = lo_sns->subscribe(
        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.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的AWS SDK API 參考》中的[訂閱](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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.")
```

訂閱主題的電話號碼，以透過 SMS 接收通知。

```
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.")
```

- 如需 API 詳細資訊，請參閱 [《適用於 Swift 的 SDK API 參考》](#) 中的訂閱。AWS

使用 AWS SDKs Amazon SNS 程式碼範例

下列程式碼範例示範如何使用 Amazon SNS 搭配 AWS 軟體開發套件 (SDK)。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境中查看內容中的動作。

案例是向您展示如何呼叫服務中的多個函數或與其他 AWS 服務組合來完成特定任務的程式碼範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

開始使用

您好 Amazon SNS

下列程式碼範例示範如何開始使用 Amazon SNS。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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();
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [ListTopics](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

CMakeLists.txt CMake 檔案的程式碼。

```
# 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})
```

hello_sns.cpp 來源檔案的程式碼。

```
#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;
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [ListTopics](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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)
        }
    }
}
```



```
}  
}  
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [ListTopics](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ListTopics](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

初始化 SNS 用戶端並列出您帳戶中的主題。

```
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"));
};
```

- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [ListTopics](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient
```

```
        .listTopicsPaginated(ListTopicsRequest { })
        .transform { it.topics?.forEach { topic -> emit(topic) } }
        .collect { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[ListTopics](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

Package.swift 檔案。

```
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")
    ]
)

```

主要 Swift 程式。

```

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)
        }
    }
}
```

```
}  
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [ListTopics](#)。

程式碼範例

- [使用 AWS SDKs Amazon SNS 基本範例](#)
 - [您好 Amazon SNS](#)
 - [使用 AWS SDKs 的 Amazon SNS 動作](#)
 - [CheckIfPhoneNumberIsOptedOut 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmSubscription 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateTopic 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteTopic 搭配 AWS SDK 或 CLI 使用](#)
 - [GetSMSAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [GetTopicAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [ListPhoneNumbersOptedOut 搭配 AWS SDK 或 CLI 使用](#)
 - [ListSubscriptions 搭配 AWS SDK 或 CLI 使用](#)
 - [ListTopics 搭配 AWS SDK 或 CLI 使用](#)
 - [Publish 搭配 AWS SDK 或 CLI 使用](#)
 - [SetSMSAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [SetSubscriptionAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [SetSubscriptionAttributesRedrivePolicy 搭配 AWS SDK 使用](#)
 - [SetTopicAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [Subscribe 搭配 AWS SDK 或 CLI 使用](#)
 - [TagResource 搭配 AWS SDK 或 CLI 使用](#)
 - [Unsubscribe 搭配 AWS SDK 或 CLI 使用](#)
- [使用 AWS SDKs Amazon SNS 案例](#)
 - [建置應用程式以將資料提交至 DynamoDB 資料表](#)
 - [建置可轉譯訊息的發佈和訂閱應用程式](#)
 - [使用 AWS SDK 建立 Amazon SNS 推播通知的平台端點](#)
 - [建立相片資產管理應用程式，讓使用者以標籤管理相片](#)

- [建立 Amazon Textract Explorer 應用程式](#)
- [使用 AWS SDK 建立和發佈至 FIFO Amazon SNS 主題](#)
- [使用 AWS SDK 透過 Amazon Rekognition 偵測影片中的人物和物件](#)
- [使用 AWS SDK 將 SMS 訊息發佈至 Amazon SNS 主題](#)
- [使用 AWS SDK 透過 Amazon S3 將大型訊息發佈至 Amazon SNS Amazon S3](#)
- [使用 SDK 發佈 Amazon SNS SMS 文字訊息 AWS](#)
- [使用 SDK 將 Amazon SNS 訊息發佈至 Amazon SQS 佇列 AWS](#)
- [使用 API Gateway 來調用 Lambda 函數](#)
- [使用排程事件來調用 Lambda 函數](#)
- [Amazon SNS 的無伺服器範例](#)
 - [使用 Amazon SNS 觸發條件調用 Lambda 函數](#)

使用 AWS SDKs Amazon SNS 基本範例

下列程式碼範例示範如何搭配 AWS SDKs 使用 Amazon Simple Notification Service 的基本概念。

範例

- [您好 Amazon SNS](#)
- [使用 AWS SDKs 的 Amazon SNS 動作](#)
 - [CheckIfPhoneNumberIsOptedOut 搭配 AWS SDK 或 CLI 使用](#)
 - [ConfirmSubscription 搭配 AWS SDK 或 CLI 使用](#)
 - [CreateTopic 搭配 AWS SDK 或 CLI 使用](#)
 - [DeleteTopic 搭配 AWS SDK 或 CLI 使用](#)
 - [GetSMSAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [GetTopicAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [ListPhoneNumbersOptedOut 搭配 AWS SDK 或 CLI 使用](#)
 - [ListSubscriptions 搭配 AWS SDK 或 CLI 使用](#)
 - [ListTopics 搭配 AWS SDK 或 CLI 使用](#)
 - [Publish 搭配 AWS SDK 或 CLI 使用](#)
 - [SetSMSAttributes 搭配 AWS SDK 或 CLI 使用](#)
 - [SetSubscriptionAttributes 搭配 AWS SDK 或 CLI 使用](#)

- [SetSubscriptionAttributesRedrivePolicy 搭配 AWS SDK 使用](#)
- [SetTopicAttributes 搭配 AWS SDK 或 CLI 使用](#)
- [Subscribe 搭配 AWS SDK 或 CLI 使用](#)
- [TagResource 搭配 AWS SDK 或 CLI 使用](#)
- [Unsubscribe 搭配 AWS SDK 或 CLI 使用](#)

您好 Amazon SNS

下列程式碼範例示範如何開始使用 Amazon SNS。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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($"{topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [ListTopics](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

CMakeLists.txt CMake 檔案的程式碼。

```
# 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})

```

hello_sns.cpp 來源檔案的程式碼。

```

#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;
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [ListTopics](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [ListTopics](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
        }
    }
}
```

```
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ListTopics](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

初始化 SNS 用戶端並列出您帳戶中的主題。

```
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"));  
};
```

- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [ListTopics](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
import aws.sdk.kotlin.services.sns.SnsClient  
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest  
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated  
import kotlinx.coroutines.flow.transform  
  
/**  
Before running this Kotlin code example, set up your development environment,  
including your credentials.  
  
For more information, see the following documentation topic:  
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html  
*/  
suspend fun main() {  
    listTopicsPag()  
}  
  
suspend fun listTopicsPag() {  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient  
            .listTopicsPaginated(ListTopicsRequest { })  
            .transform { it.topics?.forEach { topic -> emit(topic) } }  
            .collect { topic ->  
                println("The topic ARN is ${topic.topicArn}")  
            }  
    }  
}
```

```
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [ListTopics](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

Package.swift 檔案。

```
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")
    ]
)

```

主要 Swift 程式。

```

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)
        }
    }
}

```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [ListTopics](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDKs 的 Amazon SNS 動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Amazon SNS 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

這些摘錄會呼叫 Amazon SNS API，是必須在內容中執行之大型程式的程式碼摘錄。您可以在 [使用 AWS SDKs Amazon SNS 案例](#) 中查看內容中的動作。

下列範例僅包含最常使用的動作。如需完整的列表，請參閱 [Amazon Simple Notification Service API 參考資料](#)。

範例

- [CheckIfPhoneNumberIsOptedOut 搭配 AWS SDK 或 CLI 使用](#)
- [ConfirmSubscription 搭配 AWS SDK 或 CLI 使用](#)
- [CreateTopic 搭配 AWS SDK 或 CLI 使用](#)
- [DeleteTopic 搭配 AWS SDK 或 CLI 使用](#)
- [GetSMSAttributes 搭配 AWS SDK 或 CLI 使用](#)
- [GetTopicAttributes 搭配 AWS SDK 或 CLI 使用](#)
- [ListPhoneNumbersOptedOut 搭配 AWS SDK 或 CLI 使用](#)
- [ListSubscriptions 搭配 AWS SDK 或 CLI 使用](#)
- [ListTopics 搭配 AWS SDK 或 CLI 使用](#)
- [Publish 搭配 AWS SDK 或 CLI 使用](#)
- [SetSMSAttributes 搭配 AWS SDK 或 CLI 使用](#)
- [SetSubscriptionAttributes 搭配 AWS SDK 或 CLI 使用](#)
- [SetSubscriptionAttributesRedrivePolicy 搭配 AWS SDK 使用](#)
- [SetTopicAttributes 搭配 AWS SDK 或 CLI 使用](#)
- [Subscribe 搭配 AWS SDK 或 CLI 使用](#)
- [TagResource 搭配 AWS SDK 或 CLI 使用](#)
- [Unsubscribe 搭配 AWS SDK 或 CLI 使用](#)

CheckIfPhoneNumberIsOptedOut 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CheckIfPhoneNumberIsOptedOut。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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}");
    }
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

CLI

AWS CLI

檢查電話號碼是否停止接收簡訊

下列check-if-phone-number-is-opted-out範例會檢查指定的電話號碼是否選擇不接收來自目前 AWS 帳戶的簡訊。

```
aws sns check-if-phone-number-is-opted-out \  
--phone-number +1555550100
```

輸出：

```
{  
  "isOptedOut": false  
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CheckIfPhoneNumberIsOptedOut](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
```

```
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#check-if-a-phone-number-has-opted-out>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [CheckIfPhoneNumberIsOptedOut](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ConfirmSubscription 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ConfirmSubscription。

CLI

AWS CLI

確認訂閱

下列 `confirm-subscription` 命令會完成您訂閱名為 `my-topic` 的 SNS 主題時啟動的確認程序。`-token` 參數來自傳送到訂閱呼叫中所指定通知端點的確認訊息。

```
aws sns confirm-subscription \
  --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
  --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

輸出：

```
{
  "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ConfirmSubscription](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ConfirmSubscription](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```

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({});

```

匯入 SDK 和用戶端模組，然後呼叫 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-
  xxx-xxxx-xxxx-xxxxxxxxxxxxx'
  // }
  return response;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [ConfirmSubscription](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [ConfirmSubscription](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

CreateTopic 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 CreateTopic。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [建立並發布到 FIFO 主題](#)
- [將訊息發佈至佇列](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

建立具有特定名稱的主題。

```
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;
        }
    }
}

```

建立具有名稱、特定 FIFO 和重複資料刪除屬性的新主題。

```

    /// <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;
    }

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [CreateTopic](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
//! 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();  
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [CreateTopic](#)。

CLI

AWS CLI

建立 SNS 主題

下列 create-topic 範例會建立名為 my-topic 的 SNS 主題。

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

輸出：

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

如需詳細資訊，請參閱《[AWS 命令列界面使用者指南](#)》中的[搭配 Amazon SQS 和 Amazon SNS 使用 AWS 命令列界面](#)。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [CreateTopic](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [CreateTopic](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 "";
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [CreateTopic](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;  
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [CreateTopic](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun createSNSTopic(topicName: String): String {  
    val request =  
        CreateTopicRequest {  
            name = topicName  
        }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val result = snsClient.createTopic(request)  
        return result.topicArn.toString()  
    }  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 [CreateTopic](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-managing-topics.html#create-a-topic>。
- 如需 API 詳細資訊，請參閱《適用於 PHP 的 AWS SDK API 參考》中的 [CreateTopic](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [CreateTopic](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
# 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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-create-topic.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 [CreateTopic](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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(())
}
```

- 如需 API 詳細資訊，請參閱適用於 Rust API 的 AWS SDK 參考中的 [CreateTopic](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP API 的 AWS SDK 參考》中的 [CreateTopic](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [CreateTopic](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

DeleteTopic 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 DeleteTopic。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [將訊息發佈至佇列](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

藉由主題 ARN 刪除該主題。

```
/// <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;
    }

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [DeleteTopic](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! 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();
}
```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

CLI

AWS CLI

刪除 SNS 主題

下列 delete-topic 範例會刪除指定的 SNS 主題。

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

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [DeleteTopic](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}

```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [DeleteTopic](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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);
        } catch (SnsException e) {
            System.out.println("Error deleting topic: " + e.getMessage());
        }
    }
}
```

```

        System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

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

```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的 [DeleteTopic](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```

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({});

```

匯入 SDK 和用戶端模組，然後呼叫 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
  //   }
  // }
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun deleteSNSTopic(topicArnVal: String) {
    val request =
        DeleteTopicRequest {
            topicArn = topicArnVal
        }

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

```
}  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 DeleteTopic。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}

```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [DeleteTopic](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [DeleteTopic](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteTopic](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

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

```
let snsClient = SNSClient(config: config)

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

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [DeleteTopic](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

GetSMSAttributes 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 GetSMSAttributes。

C++

SDK for C++

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
//! 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();
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [GetSMSAttributes](#)。

CLI

AWS CLI

列出預設簡訊屬性

下列 `get-sms-attributes` 範例會列出傳送簡訊的預設屬性。

```
aws sns get-sms-attributes
```

輸出：

```
{
  "attributes": {
    "DefaultSenderId": "MyName"
  }
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [GetSMSAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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");
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [GetSMSAttributes](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [GetSMSAttributes](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#get-sms-attributes>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [GetSMSAttributes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

GetTopicAttributes 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 GetTopicAttributes。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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");
        }
    }
}

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [GetTopicAttributes](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

//! 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();
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [GetTopicAttributes](#)。

CLI

AWS CLI

擷取主題的屬性

下列 `get-topic-attributes` 範例會顯示指定主題的屬性。

```
aws sns get-topic-attributes \
```

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

輸出：

```
{
  "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"
  }
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [GetTopicAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [GetTopicAttributes](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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,"numNoDelayRetriesBetweenDelays":0}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};

```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。

- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [GetTopicAttributes](#)。

SDK for JavaScript (v2)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

匯入 SDK 和用戶端模組，然後呼叫 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);
  });
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [GetTopicAttributes](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {
    val request =
        GetTopicAttributesRequest {
            topicArn = topicArnVal
        }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [GetTopicAttributes](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
$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());
    }

```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [GetTopicAttributes](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [GetTopicAttributes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ListPhoneNumbersOptedOut 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListPhoneNumbersOptedOut。

CLI

AWS CLI

列出停止接收簡訊

下列 list-phone-numbers-opted-out 範例會列出選擇停止接收簡訊的電話號碼。

```
aws sns list-phone-numbers-opted-out
```

輸出：

```
{
  "phoneNumbers": [
    "+15555550100"
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ListPhoneNumbersOptedOut](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ListPhoneNumbersOptedOut](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#list-opted-out-phone-numbers>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [ListPhoneNumbersOptedOut](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ListSubscriptions 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListSubscriptions。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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();
        }
    }
}

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [ListSubscriptions](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
//! 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;
}
```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [ListSubscriptions](#)。

CLI

AWS CLI

列出您的 SNS 訂閱

下列 `list-subscriptions` 範例顯示您 AWS 帳戶中 SNS 訂閱的清單。

```
aws sns list-subscriptions
```

輸出：

```
{
  "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"
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ListSubscriptions](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ListSubscriptions](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。

- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [ListSubscriptions](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun listSNSSubscriptions() {  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})  
        response.subscriptions?.forEach { sub ->  
            println("Sub ARN is ${sub.subscriptionArn}")  
            println("Sub protocol is ${sub.protocol}")  
        }  
    }  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [ListSubscriptions](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [ListSubscriptions](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [ListSubscriptions](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
# 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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-show-subscriptions.html>。

- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 [ListSubscriptions](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [ListSubscriptions](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

ListTopics 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 ListTopics。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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}");
        }
    }
}

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [ListTopics](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/*! 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;
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [ListTopics](#)。

CLI

AWS CLI

列出您的 SNS 主題

下列 `list-topics` 範例列出您 AWS 帳戶中的所有 SNS 主題。

```
aws sns list-topics
```

輸出：

```
{
  "Topics": [
    {
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [ListTopics](#)。

Go

SDK for Go V2

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的 [ListTopics](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [ListTopics](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [ListTopics](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun listSNSTopics() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 [《適用於 Kotlin 的 AWS SDK API 參考》](#) 中的 [ListTopics](#)。

PHP

SDK for PHP

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [ListTopics](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [ListTopics](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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__
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-show-topics.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 [ListTopics](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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(())
}
```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的 AWS SDK API 參考》中的 [ListTopics](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [ListTopics](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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)  
    }  
}
```



```
        }
        topics.append(arn)
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的 [ListTopics](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Publish 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 Publish。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [建立並發布到 FIFO 主題](#)
- [發布簡訊](#)
- [將訊息發佈至佇列](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

發布訊息至主題。

```
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}");
    }
}
```

將訊息發佈至具有群組、複寫和屬性選項的主題。

```
/// <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);
}
}

```

將使用者的選擇套用至發佈動作。

```

    /// <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;
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [發佈](#)。

C++

適用於 C++ 的 SDK

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
//! 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();
}
```

發佈具有 屬性的訊息。

```

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

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

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

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

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

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

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

    cleanUp(topicARN,
            queueURLS,

```

```
        subscriptionARNS,  
        snsClient,  
        sqsClient);  
  
    return false;  
}
```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的[發佈](#)。

CLI

AWS CLI

範例 1：將訊息發布至主題

下列 `publish` 範例會將指定的訊息發佈到指定的 SNS 主題。訊息來自文字檔案，可讓您包含換行符號。

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

`message.txt` 的內容：

```
Hello World  
Second Line
```

輸出：

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

範例 2：將簡訊發布至電話號碼

下列 `publish` 範例會將訊息 `Hello world!` 發佈至電話號碼 `+1-555-555-0100`。

```
aws sns publish \  
  --message "Hello world!" \  
  --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```



```
--phone-number +1-555-555-0100
```

輸出：

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

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [Publish](#)。

Go

SDK for Go V2

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的[發佈](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[發佈](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;
};
```

將訊息發佈至具有群組、複寫和屬性選項的主題。

```
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();
}
}

```

- 如需詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/sns-examples-publishing-messages.html>。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的[發佈](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

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

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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[發佈](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}

```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-subscribing-unsubscribing-topics.html#publish-a-message-to-an-sns-topic>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[發佈](#)。

PowerShell

PowerShell V4 的工具

範例 1：此範例顯示使用單一 MessageAttribute 內嵌宣告來發佈訊息。

```

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

```

範例 2：此範例顯示預先宣告多個 MessageAttributes 的訊息發佈。

```

$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

```

- 如需 API 詳細資訊，請參閱在 AWS Tools for PowerShell Cmdlet 參考 (V4) 中 [發佈](#)。

PowerShell V5 的工具

範例 1：此範例顯示使用單一 MessageAttribute 內嵌宣告來發佈訊息。

```

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

```

範例 2：此範例顯示預先宣告多個 MessageAttributes 的訊息發佈。

```

$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
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 在 Cmdlet 參考 (V5) 中[發佈](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

發佈具有屬性的郵件，以便訂閱可以根據屬性進行篩選。

```
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

```

發佈根據訂閱者的通訊協定採用不同形式的訊息。

```

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

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[發佈](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
# 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
```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-send-message.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的[發佈](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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(())
}

```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的 AWS SDK API 參考》中的[發佈](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
    oo_result = lo_sns->publish(
testing purposes. " oo_result is returned for
    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.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的[發佈](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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)")
```

- 如需 API 詳細資訊，請參閱在適用於 AWS Swift 的 SDK API 參考中 [發佈](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

SetSMSAttributes 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 SetSMSAttributes。

C++

SDK for C++

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

如何使用 Amazon SNS 設定 DefaultSMSType 屬性。

```
//! 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();
}
```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的 [SetSMSAttributes](#)。

CLI

AWS CLI

若要設定簡訊屬性

下列 `set-sms-attributes` 範例會將簡訊的預設寄件者 ID 設定為 `MyName`。

```
aws sns set-sms-attributes \  
  --attributes DefaultSenderId=MyName
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetSMSAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [SetSMSAttributes](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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;  
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [SetSMSAttributes](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
$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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#set-sms-attributes>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [SetSMSAttributes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

SetSubscriptionAttributes 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 SetSubscriptionAttributes。

CLI

AWS CLI

設定訂閱屬性

下列 set-subscription-attributes 範例會將 RawMessageDelivery 屬性設定為 SQS 訂閱。

```
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
```

此命令不會產生輸出。

下列 set-subscription-attributes 範例會將 FilterPolicy 屬性設定為 SQS 訂閱。

```
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\"] }"
```

此命令不會產生輸出。

下列 `set-subscription-attributes` 範例會從 SQS 訂閱移除 `FilterPolicy` 屬性。

```
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 "{}"
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetSubscriptionAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [SetSubscriptionAttributes](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [SetSubscriptionAttributes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

SetSubscriptionAttributesRedrivePolicy 搭配 AWS SDK 使用

以下程式碼範例顯示如何使用 SetSubscriptionAttributesRedrivePolicy。

Java

適用於 Java 1.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
// 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);
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

SetTopicAttributes 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 SetTopicAttributes。

CLI

AWS CLI

設定主題的屬性

下列 `set-topic-attributes` 範例會設定指定主題的 `DisplayName` 屬性。

```
aws sns set-topic-attributes \  
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \  
  --attribute-name DisplayName \  
  --attribute-value MyTopicDisplayName
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [SetTopicAttributes](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [SetTopicAttributes](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```

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({});

```

匯入 SDK 和用戶端模組，然後呼叫 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;
};

```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [SetTopicAttributes](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun setTopAttr(
```



```
        attribute: String?,
        topicArnVal: String?,
        value: String?,
    ) {
        val request =
            SetTopicAttributesRequest {
                attributeName = attribute
                attributeValue = value
                topicArn = topicArnVal
            }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            snsClient.setTopicAttributes(request)
            println("Topic ${request.topicArn} was updated.")
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》<https://sdk.amazonaws.com/kotlin/api/latest/index.html> 中的 SetTopicAttributes。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}

```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的 [SetTopicAttributes](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

# 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

```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-enable-resource.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的 [SetTopicAttributes](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [SetTopicAttributes](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Subscribe 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 Subscribe。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [建立並發布到 FIFO 主題](#)
- [將訊息發佈至佇列](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
/// <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;
}
```

使用篩選條件訂閱主題的佇列。

```
/// <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;
    }

```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的[訂閱](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```

//! 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();
}

```

訂閱行動應用程式至主題。

```

//! 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();
}

```

訂閱主題的 Lambda 函數。

```

//! 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();
}

```

訂閱主題的 SQS 佇列。

```

    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;
    }

```

使用篩選條件訂閱主題。

```

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;
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的[訂閱](#)。

CLI

AWS CLI

訂閱主題

下列 `subscribe` 命令會將電子郵件地址訂閱至指定的主題。

```
aws sns subscribe \
```

```
--topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \  
--protocol email \  
--notification-endpoint my-email@example.com
```

輸出：

```
{  
  "SubscriptionArn": "pending confirmation"  
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [Subscribe](#)。

Go

SDK for Go V2

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

使用篩選條件訂閱主題的佇列。

```
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
}
```

- 如需 API 詳細資訊，請參閱 適用於 Go 的 AWS SDK API 參考中的[訂閱](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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);
        }
    }
}

```

訂閱 HTTP 端點至主題。

```

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);
    }
}
}
}

```

訂閱主題的 Lambda 函數。

```

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 "";
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的[訂閱](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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'
// }
};

```

訂閱行動應用程式至主題。

```

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;
};
```

訂閱主題的 Lambda 函數。

```
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;
};
```

訂閱主題的 SQS 佇列。

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

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
  test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

使用篩選條件訂閱主題。

```
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;
};
```

- 如需詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-javascript/v3/developer-guide/sns-examples-managing-topics.html#sns-examples-subscribing-email>。
- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的[訂閱](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
suspend fun subEmail(
    topicArnVal: String,
    email: String,
): String {
    val request =
        SubscribeRequest {
            protocol = "email"
            endpoint = email
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

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

訂閱主題的 Lambda 函數。

```
suspend fun subLambda(
    topicArnVal: String?,
    lambdaArn: String?,
) {
    val request =
        SubscribeRequest {
            protocol = "lambda"
            endpoint = lambdaArn
            returnSubscriptionArn = true
        }
```

```
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[訂閱](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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());
    }

```

訂閱 HTTP 端點至主題。

```

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());
}
```

- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[訂閱](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[訂閱](#)。

Ruby

SDK for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
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
```



```

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

```

- 如需詳細資訊，請參閱《適用於 Ruby 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-ruby/v3/developer-guide/sns-example-create-subscription.html>。
- 如需 API 詳細資訊，請參閱 適用於 Ruby 的 AWS SDK API 參考中的[訂閱](#)。

Rust

SDK for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```

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(())
}
```

- 如需 API 詳細資訊，請參閱適用於 Rust API 的 AWS SDK 參考中的[訂閱](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```
TRY.
    oo_result = lo_sns->subscribe(
        returned for testing purposes."
        "oo_result is
```

```

        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.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的[訂閱](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

訂閱主題的電子郵件地址。

```

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.")
```

訂閱主題的電話號碼，以透過簡訊接收通知。

```
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.")
```

- 如需 API 詳細資訊，請參閱 [《適用於 Swift 的 SDK API 參考》](#) 中的訂閱。AWS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

TagResource 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 TagResource。

CLI

AWS CLI

將標籤新增至主題

下列 `tag-resource` 範例會將中繼資料標籤新增到指定的 Amazon SNS 主題。

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

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [TagResource](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 [TagResource](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun addTopicTags(topicArn: String) {
    val tag =
        Tag {
            key = "Team"
            value = "Development"
        }

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

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

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

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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [TagResource](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Unsubscribe 搭配 AWS SDK 或 CLI 使用

下列程式碼範例示範如何使用 Unsubscribe。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- [將訊息發佈至佇列](#)

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

透過訂閱 ARN 取消訂閱主題。

```
/// <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;
    }

```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的 [取消訂閱](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

    //! 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();
}
```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的[取消訂閱](#)。

CLI

AWS CLI

取消訂閱主題

下列 unsubscribe 範例會從主題中刪除指定的訂閱。

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

此命令不會產生輸出。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 [Unsubscribe](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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);
    }
}
}

```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[取消訂閱](#)。

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在單獨的模組中建立用戶端並將其匯出。

```
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({});
```

匯入 SDK 和用戶端模組，然後呼叫 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-xxxxxxxxxxxxxx",
) => {
  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;
};
```

- 如需詳細資訊，請參閱 [《適用於 JavaScript 的 AWS SDK 開發人員指南》](#)。
- 如需 API 詳細資訊，請參閱 適用於 JavaScript 的 AWS SDK API 參考中的 [取消訂閱](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun unSub(subscriptionArnVal: String) {  
    val request =  
        UnsubscribeRequest {  
            subscriptionArn = subscriptionArnVal  
        }  
  
    SnsClient { region = "us-east-1" }.use { snsClient ->  
        snsClient.unsubscribe(request)  
        println("Subscription was removed for ${request.subscriptionArn}")  
    }  
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [取消訂閱](#)。

PHP

SDK for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-subscribing-unsubscribing-topics.html#unsubscribe-from-a-topic>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[取消訂閱](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的 [取消訂閱](#)。

SAP ABAP

適用於 SAP ABAP 的開發套件

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [取消訂閱](#)。

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.")
```

- 如需 API 詳細資訊，請參閱《適用於 AWS Swift 的 SDK API 參考》中的[取消訂閱](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDKs Amazon SNS 案例

下列程式碼範例示範如何在 Amazon SNS AWS SDKs 中實作常見案例。這些案例說明如何透過在 Amazon SNS 中呼叫多個函數或與其他函數結合，來完成特定任務 AWS 服務。每個案例均包含完整原始碼的連結，您可在連結中找到如何設定和執行程式碼的相關指示。

案例的目標是獲得中等水平的經驗，協助您了解內容中的服務動作。

範例

- [建置應用程式以將資料提交至 DynamoDB 資料表](#)
- [建置可轉譯訊息的發佈和訂閱應用程式](#)
- [使用 AWS SDK 建立 Amazon SNS 推播通知的平台端點](#)
- [建立相片資產管理應用程式，讓使用者以標籤管理相片](#)
- [建立 Amazon Textract Explorer 應用程式](#)
- [使用 AWS SDK 建立和發佈至 FIFO Amazon SNS 主題](#)
- [使用 AWS SDK 透過 Amazon Rekognition 偵測影片中的人物和物件](#)
- [使用 AWS SDK 將 SMS 訊息發佈至 Amazon SNS 主題](#)
- [使用 AWS SDK 透過 Amazon S3 將大型訊息發佈至 Amazon SNS Amazon S3](#)
- [使用 SDK 發佈 Amazon SNS SMS 文字訊息 AWS](#)
- [使用 SDK 將 Amazon SNS 訊息發佈至 Amazon SQS 佇列 AWS](#)
- [使用 API Gateway 來調用 Lambda 函數](#)
- [使用排程事件來調用 Lambda 函數](#)

建置應用程式以將資料提交至 DynamoDB 資料表

下列程式碼範例顯示如何建置應用程式，以將資料提交至 Amazon DynamoDB 資料表，並在使用者更新資料表時通知您。

Java

適用於 Java 2.x 的 SDK

示範如何建立動態 Web 應用程式，以使用 Amazon DynamoDB Java API 提交資料，以及使用 Amazon Simple Notification Service Java API 傳送文字訊息。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- DynamoDB
- Amazon SNS

JavaScript

SDK for JavaScript (v3)

此範例說明如何建置應用程式，讓使用者將資料提交至 Amazon DynamoDB 資料表，以及使用 Amazon Simple Notification Service (Amazon SNS) 傳送文字訊息給管理員。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例也可在 [適用於 JavaScript 的 AWS SDK v3 開發人員指南](#) 中取得。

此範例中使用的服務

- DynamoDB
- Amazon SNS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建置可轉譯訊息的發佈和訂閱應用程式

以下程式碼範例示範如何建立具有訂閱和發布功能並且可轉譯訊息的應用程式。

.NET

適用於 .NET 的 SDK

示範如何使用 Amazon Simple Notification Service .NET API 來建立具有訂閱和發布功能的 Web 應用程式。此外，此範例應用程式也會轉譯訊息。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon SNS
- Amazon Translate

Java

SDK for Java 2.x

示範如何使用 Amazon Simple Notification Service Java API 來建立具有訂閱和發布功能的 Web 應用程式。此外，此範例應用程式也會轉譯訊息。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

如需完整的原始碼和如何設定及執行使用 Java Async API 之範例的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon SNS
- Amazon Translate

Kotlin

SDK for Kotlin

示範如何使用 Amazon SNS Kotlin API 來建立具有訂閱和發布功能的應用程式。此外，此範例應用程式也會轉譯訊息。

如需完整的原始碼和如何建立 Web 應用程式的指示，請參閱 [GitHub](#) 上的完整範例。

如需完整的原始碼和如何建立原生 Android 應用程式的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon SNS
- Amazon Translate

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 建立 Amazon SNS 推播通知的平台端點

下列程式碼範例示範如何為 Amazon SNS 推播通知建立平台端點。

CLI

AWS CLI

建立平台應用程式端點

下列 `create-platform-endpoint` 範例會使用指定的字符，為指定的平台應用程式建立端點。

```
aws sns create-platform-endpoint \  
  --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/MyApplication \  
  --token EXAMPLE12345...
```

輸出：

```
{  
  "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/MyApplication/12345678-abcd-9012-efgh-345678901234"  
}
```

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
    }
}
}

```

```
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立相片資產管理應用程式，讓使用者以標籤管理相片

下列程式碼範例示範如何建立無伺服器應用程式，讓使用者以標籤管理相片。

.NET

適用於 .NET 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

若要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

適用於 C++ 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

若要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

適用於 Java 2.x 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

若要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

如要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

適用於 Kotlin 的 SDK

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

如要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

如要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK for Rust

顯示如何開發照片資產管理應用程式，以便使用 Amazon Rekognition 偵測圖片中的標籤，並將其儲存以供日後擷取。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

如要深入探索此範例的來源，請參閱 [AWS 社群](#) 上的文章。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立 Amazon Textract Explorer 應用程式

下列程式碼範例示範如何透過互動式應用程式探索 Amazon Textract 輸出。

JavaScript

適用於 JavaScript (v3) 的 SDK

說明如何使用 適用於 JavaScript 的 AWS SDK 建置 React 應用程式，該應用程式使用 Amazon Textract 從文件映像擷取資料，並將其顯示在互動式網頁中。此範例會在 Web 瀏覽器中執行，且登入資料需要經過驗證的 Amazon Cognito 身分。它使用 Amazon Simple Storage Service (Amazon S3 進行儲存，對於通知，它會輪詢訂閱 Amazon Simple Notification Service (Amazon SNS)) 主題的 Amazon Simple Queue Service (Amazon SQS) 佇列。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python (Boto3)

顯示如何使用 適用於 Python (Boto3) 的 AWS SDK 搭配 Amazon Textract 來偵測文件映像中的文字、表單和資料表元素。輸入影像和 Amazon Textract 輸出會顯示在 Tkinter 應用程式中，可讓您探索偵測到的元素。

- 將文件影像提交到 Amazon Textract，並探索偵測到元素的輸出。
- 將影像直接傳送至 Amazon Textract 或透過 Amazon Simple Storage Service (Amazon S3) 儲存貯體。
- 使用非同步 API 可以在任務完成時啟動將通知發布到 Amazon Simple Notification Service (Amazon SNS) 主題的任務。
- 輪詢 Amazon Simple Queue Service (Amazon SQS) 佇列以取得任務完成訊息並顯示結果。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon Cognito Identity

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 建立和發佈至 FIFO Amazon SNS 主題

下列程式碼範例示範如何建立並發布到 FIFO Amazon SNS 主題。

Java

適用於 Java 2.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

此範例

- 建立一個 Amazon SNS FIFO 主題、兩個 Amazon SQS FIFO 佇列和一個標準佇列。
- 訂閱佇列到主題並向該主題發布訊息。

[測試](#)可驗證每個佇列的訊息接收狀況。[完整範例](#)也會顯示存取政策的新增，並在最後刪除資源。

```
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);
    }
}
```


- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
 - [CreateTopic](#)
 - [發布](#)
 - [Subscribe](#)

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

建立 Amazon SNS FIFO 主題，訂閱 Amazon SQS FIFO 和標準佇列到該主題，並向該主題發布訊息。

```
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

```

- 如需 API 的詳細資訊，請參閱 AWS SDK for Python (Boto3) API Reference 中的下列主題。
 - [CreateTopic](#)
 - [發布](#)
 - [Subscribe](#)

SAP ABAP

適用於 SAP ABAP 的開發套件

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

建立 FIFO 主題、將 Amazon SQS FIFO 佇列訂閱至主題，然後將訊息發佈至 Amazon SNS 主題。

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

```


- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的下列主題。
 - [CreateTopic](#)
 - [發布](#)
 - [Subscribe](#)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 透過 Amazon Rekognition 偵測影片中的人物和物件

下列程式碼範例示範如何使用 Amazon Rekognition 偵測映像中的人物和物件。

Java

適用於 Java 2.x 的 SDK

示範如何使用 Amazon Rekognition Java API 來建立應用程式，以偵測位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體的映像中的人臉和物件。此應用程式可使用 Amazon Simple Email Service (Amazon SES) 向管理員傳送包含結果的電子郵件通知。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

適用於 Python (Boto3) 的 SDK

使用 Amazon Rekognition 透過啟動非同步偵測任務來偵測映像中的人臉、物件和人物。此範例也會設定 Amazon Rekognition 以在任務完成時通知 Amazon Simple Notification Service

(Amazon SNS) 主題，並訂閱 Amazon Simple Queue Service (Amazon SQS) 佇列到該主題。當佇列收到有關任務的訊息時，會擷取任務並輸出結果。

這個範例在 GitHub 上的檢視效果最佳。如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 將 SMS 訊息發佈至 Amazon SNS 主題

以下代碼範例顯示做法：

- 建立 Amazon SNS 主題。
- 使用手機號碼訂閱主題。
- 將簡訊發布至主題，讓所有訂閱的電話號碼一次接收訊息。

Java

適用於 Java 2.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

創造一個主題並回傳其 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 "";
}
}

```

讓端點訂閱主題

```

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);
    }
}
}
}

```

設定訊息的屬性，例如寄件者的 ID、最高價格及其類型。訊息屬性為選用。

```
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);
        }
    }
}
```

```

    }
  }
}

```

發布訊息至主題。訊息會傳送至每位訂閱者。

```

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);
        }
    }
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 透過 Amazon S3 將大型訊息發佈至 Amazon SNS Amazon S3

下列程式碼範例顯示如何使用 Amazon S3 將大型訊息發布到 Amazon SNS，以便存放訊息承載。

Java

適用於 Java 1.x 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

若要發布大型訊息，請使用適用於 Java 的 Amazon SNS 擴充用戶端程式庫。您傳送的訊息會參考包含實際訊息內容的 Amazon S3 物件。

```
import com.amazonaws.jmes.messaging.AmazonSQSExtendedClient;
import com.amazonaws.jmes.messaging.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());
    }
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 發佈 Amazon SNS SMS 文字訊息 AWS

下列程式碼範例示範如何使用 Amazon SNS 發佈簡訊。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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}");
    }
}
}
```

- 如需 API 詳細資訊，請參閱 適用於 .NET 的 AWS SDK API 參考中的[發佈](#)。

C++

適用於 C++ 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/**
 * 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();
}

```

- 如需 API 詳細資訊，請參閱 適用於 C++ 的 AWS SDK API 參考中的[發佈](#)。

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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);
    }
}
}
```


- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的[發佈](#)。

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun pubTextSMS(
    messageVal: String?,
    phoneNumberVal: String?,
) {
    val request =
        PublishRequest {
            message = messageVal
            phoneNumber = phoneNumberVal
        }

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

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的[發佈](#)。

PHP

SDK for PHP

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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());
}
```

- 如需詳細資訊，請參閱《適用於 PHP 的 AWS SDK 開發人員指南》<https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/sns-examples-sending-sms.html#publish-to-a-text-message-sms-message>。
- 如需 API 詳細資訊，請參閱 適用於 PHP 的 AWS SDK API 參考中的[發佈](#)。

Python

SDK for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Python (Boto3) API 參考中的[發佈](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 將 Amazon SNS 訊息發佈至 Amazon SQS 佇列 AWS

下列程式碼範例示範如何：

- 建立主題 (FIFO 或非 FIFO)。
- 為主題訂閱多個佇列，並提供套用篩選條件的選擇。
- 發佈訊息至主題。
- 輪詢佇列以獲取收到的訊息。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
/// <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\nand queue URL {queueUrl}" +
                $"\r\nhas been created.\r\n");

            if (i == 0)
            {
                Console.WriteLine(
                    $"The queue URL is used to retrieve the queue ARN,\r\n" +
                    $"which is used to create a subscription.");
                Console.WriteLine(new string('-', 80));
            }

            var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

            if (i == 0)
            {
                Console.WriteLine(
```

```

        $"An AWS Identity and Access Management (IAM) policy must
        be attached to an SQS queue, enabling it to receive\r\n" +
        $"messages from an SNS topic");
    }

    await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
queueUrl);

    await SetupFilters(i, queueArn, queueName);
}
}

Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Set up filters with user options for a queue.
/// </summary>
/// <param name="queueCount">The number of this queue.</param>
/// <param name="queueArn">The ARN of the queue.</param>
/// <param name="queueName">The name of the queue.</param>
/// <returns>Async Task.</returns>
public static async Task SetupFilters(int queueCount, string queueArn, string
queueName)
{
    if (_useFifoTopic)
    {
        Console.WriteLine(new string('-', 80));
        // Only explain this once.
        if (queueCount == 0)
        {
            Console.WriteLine(
                "Subscriptions to a FIFO topic can have filters." +
                "If you add a filter to this subscription, then only the
filtered messages " +
                "will be received in the queue.");

            Console.WriteLine(
                "For information about message filtering, " +
                "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

            Console.WriteLine(
                "For this example, you can filter messages by a " +

```

```

        "TONE attribute.");
    }

    var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
subscription to the topic?");

    string? filterPolicy = null;
    if (useFilter)
    {
        filterPolicy = CreateFilterPolicy();
    }
    var subscriptionArn = await
SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
        queueArn);
    _subscriptionArns[queueCount] = subscriptionArn;

    Console.WriteLine(
        $"The queue {queueName} has been subscribed to the topic
{_topicName} " +
        $"with the subscription ARN {subscriptionArn}");
    Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Use user input to create a filter policy for a subscription.
/// </summary>
/// <returns>The serialized filter policy.</returns>
public static string CreateFilterPolicy()
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine(
        $"You can filter messages by one or more of the following" +
        $"TONE attributes.");

    List<string> filterSelections = new List<string>();

    var selectionNumber = 0;
    do
    {
        Console.WriteLine(
            $"Enter a number to add a TONE filter, or enter 0 to stop adding
filters.");
        for (int i = 0; i < _tones.Length; i++)

```

```

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

        var selection = GetUserResponse("", filterSelections.Any() ? "0" :
"1");
        int.TryParse(selection, out selectionNumber);
        if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
        {
            filterSelections.Add(_tones[selectionNumber - 1]);
        }
    } while (selectionNumber != 0);

    var filters = new Dictionary<string, List<string>>
    {
        { "tone", filterSelections }
    };
    string filterPolicy = JsonSerializer.Serialize(filters);
    return filterPolicy;
}

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

    var keepSendingMessages = true;
    string? deduplicationId = null;
    string? toneAttribute = null;
    while (keepSendingMessages)
    {
        Console.WriteLine();
        var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

        if (_useFifoTopic)
        {
            Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +

```

```

        "\r\nAll messages within the same group will be
received in the order " +
        "they were published.");

        Console.WriteLine();
        var 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,
            messageId);

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

```

```

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

/// <summary>
/// Poll for the published messages to see the results of the user's choices.
/// </summary>
/// <returns>Async task.</returns>
public static async Task<List<Message>> PollForMessages(string queueUrl)
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
        "\r\nPress any key to continue.");
    if (UseConsole)
    {
        Console.ReadLine();
    }

    var moreMessages = true;
    var messages = new List<Message>();
    while (moreMessages)
    {
        var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

        moreMessages = newMessages.Any();
        if (moreMessages)
        {
            messages.AddRange(newMessages);
        }
    }

    Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

    foreach (var message in messages)
    {
        Console.WriteLine("\tMessage:" +
            $"{"\n\t{message.Body}");
    }

    Console.WriteLine(new string('-', 80));

```

```

        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))

```

```

        {
            await SnsWrapper.UnsubscribeByArn(subscriptionArn);
        }
    }

    var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
    if (deleteTopic)
    {
        await SnsWrapper.DeleteTopicByArn(_topicArn);
    }
}
catch (Exception ex)
{
    Console.WriteLine($"Unable to clean up resources. Here's why:
{ex.Message}.");
}

Console.WriteLine(new string('-', 80));
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="defaultAnswer">Optional default answer to use.</param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question, bool defaultAnswer =
true)
{
    if (UseConsole)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
            ynResponse.Equals("y",
                StringComparison.InvariantCultureIgnoreCase);

        return response;
    }
    // If not using the console, use the default.
    return defaultAnswer;
}

/// <summary>

```



```

    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}

```

建立包裝 Amazon SQS 操作的類別。

```

    /// <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;
    }
}

```

建立包裝 Amazon SNS 操作的類別。

```
/// <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;
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的下列主題。
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)

- [發布](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

C++

SDK for C++

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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();
}

```

- 如需 API 詳細資訊，請參閱《適用於 C++ 的 AWS SDK API 參考》中的下列主題。

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [發布](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)

- [Subscribe](#)
- [Unsubscribe](#)

Go

SDK for Go V2

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

在命令提示中執行互動式案例。

```
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
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))
}

```

定義包裝此範例中所用 Amazon SNS 動作的結構。

```

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
}

```

定義包裝此範例中所用 Amazon SQS 動作的結構。

```

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,
    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
}

```

清除資源。

```

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)
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Go 的 AWS SDK API 參考》中的下列主題。
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [發布](#)
 - [ReceiveMessage](#)

- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

Java

SDK for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 "";
}
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [發布](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

JavaScript

SDK for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

這是此案例的進入點。

```
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();
};
```

上述程式碼提供必要的相依性並啟動案例。下一節包含範例的大量內容。

```
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();
    }
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 JavaScript 的 AWS SDK API 參考》中的下列主題。

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [發布](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

Kotlin

SDK for Kotlin

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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> = mutableListOfOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: String?) {
```

```

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

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

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

```



```

    }
} else {
    val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
        dataType = "String"
        stringValue = "true"
    }

    val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
        mapOf(msgAttValue to messAttr)
    if (duplication.compareTo("y") == 0) {
        val request = PublishRequest {
            message = messageVal
            messageGroupId = groupIdVal
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    } else {
        // Create a publish request with the message and attributes.
        val request = PublishRequest {
            topicArn = topicArnVal
            message = messageVal
            messageDeduplicationId = deduplicationID
            messageGroupId = groupIdVal
            messageAttributes = mapAtt
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.publish(request)
            println(result.messageId.toString() + " Message sent.")
        }
    }
}

}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {

```

```

// No filter subscription is added.
request = SubscribeRequest {
    protocol = "sqs"
    endpoint = queueArnVal
    returnSubscriptionArn = true
    topicArn = topicArnVal
}

SnsClient { region = "us-east-1" }.use { snsClient ->
    val result = snsClient.subscribe(request)
    println(
        "The queue " + queueArnVal + " has been subscribed to the topic "
+ topicArnVal + "\n" +
        "with the subscription ARN " + result.subscriptionArn,
    )
    return result.subscriptionArn
}
} else {
    request = SubscribeRequest {
        protocol = "sqs"
        endpoint = queueArnVal
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println("The queue $queueArnVal has been subscribed to the topic
$topicArnVal with the subscription ARN ${result.subscriptionArn}")

        val attributeNameVal = "FilterPolicy"
        val gson = Gson()
        val jsonString = "{\"tone\": []}"
        val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
        val toneArray = jsonObject.getAsJsonArray("tone")
        for (value: String? in filterList) {
            toneArray.add(JsonPrimitive(value))
        }

        val updatedJsonString: String = gson.toJson(jsonObject)
        println(updatedJsonString)
        val attRequest = SetSubscriptionAttributesRequest {
            subscriptionArn = result.subscriptionArn
            attributeName = attributeNameVal

```

```

        attributeValue = updatedJsonString
    }

    snsClient.setSubscriptionAttributes(attRequest)
    return result.subscriptionArn
}
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

```

```
suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

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

```
        name = topicName
    }

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

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }

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

- 如需 API 詳細資訊，請參閱 AWS SDK for Kotlin API reference 中的下列主題。
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [發布](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)

- [Subscribe](#)
- [Unsubscribe](#)

Swift

SDK for Swift

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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)
    }
}
}
```

- 如需 API 詳細資訊，請參閱《適用於 Swift 的 AWS SDK API 參考》中的下列主題。

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [發布](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 API Gateway 來調用 Lambda 函數

下列程式碼範例示範如何建立 Amazon API Gateway 調用的 AWS Lambda 函數。

Java

適用於 Java 2.x 的 SDK

顯示如何使用 Lambda Java 執行時間 API 建立 AWS Lambda 函數。此範例會叫用不同的 AWS 服務來執行特定的使用案例。此範例示範如何建立 Amazon API Gateway 調用的 Lambda 函數，該函數會掃描 Amazon DynamoDB 資料表中的工作週年紀念日，並使用 Amazon Simple Notification Service (Amazon SNS) 傳送文字訊息給您的員工，在他們的週年紀念日向他們道賀。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的 SDK

示範如何使用 Lambda JavaScript 執行時間 API 建立 AWS Lambda 函數。此範例會叫用不同的 AWS 服務來執行特定的使用案例。此範例示範如何建立 Amazon API Gateway 調用的 Lambda 函數，該函數會掃描 Amazon DynamoDB 資料表中的工作週年紀念日，並使用 Amazon Simple Notification Service (Amazon SNS) 傳送文字訊息給您的員工，在他們的週年紀念日向他們道賀。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例也可在 [適用於 JavaScript 的 AWS SDK v3 開發人員指南](#) 中取得。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Python

適用於 Python (Boto3) 的開發套件

此範例顯示如何建立和使用目標為 AWS Lambda 函數的 Amazon API Gateway REST API。Lambda 處理常式會展示如何根據 HTTP 方法來路由；如何從查詢字串、標頭和本文中取得資料；以及如何傳回 JSON 回應。

- 部署 Lambda 函數。
- 建立 API Gateway REST API。

- 建立目標為 Lambda 函數的 REST 資源。
- 授與許可讓 API Gateway 調用 Lambda 函數。
- 使用 Request 套件來將請求傳送到 REST API。
- 清理示範期間建立的所有資源。

這個範例在 GitHub 上的檢視效果最佳。如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用排程事件來調用 Lambda 函數

下列程式碼範例示範如何建立由 Amazon EventBridge 排程事件呼叫的 AWS Lambda 函數。

Java

適用於 Java 2.x 的 SDK

顯示如何建立叫用 AWS Lambda 函數的 Amazon EventBridge 排程事件。將 EventBridge 設定為在調用 Lambda 函數時使用 cron 運算式來進行排程。在此範例中，您會使用 Lambda Java 執行期 API 建立 Lambda 函數。此範例會叫用不同的 AWS 服務來執行特定的使用案例。此範例示範如何建立應用程式，將行動裝置文字訊息傳送給員工，在他們的週年紀念日向他們道賀。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- CloudWatch Logs
- DynamoDB

- EventBridge
- Lambda
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的 SDK

顯示如何建立叫用 AWS Lambda 函數的 Amazon EventBridge 排程事件。將 EventBridge 設定為在調用 Lambda 函數時使用 cron 運算式來進行排程。在此範例中，您會使用 Lambda JavaScript 執行期 API 建立 Lambda 函數。此範例會叫用不同的 AWS 服務來執行特定的使用案例。此範例示範如何建立應用程式，將行動裝置文字訊息傳送給員工，在他們的週年紀念日向他們道賀。

如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例也可在 [適用於 JavaScript 的 AWS SDK v3 開發人員指南](#) 中取得。

此範例中使用的服務

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Python

SDK for Python (Boto3)

此範例說明如何將 AWS Lambda 函數註冊為排程 Amazon EventBridge 事件的目標。Lambda 處理常式會將合適的訊息和完整的事件資料寫入 Amazon CloudWatch Logs 中以供日後擷取。

- 部署 Lambda 函式。
- 建立一個 EventBridge 排程事件，並將 Lambda 函式做為目標。
- 授予許可讓 EventBridge 調用 Lambda 函式。
- 列印 CloudWatch Logs 中的最新資料，以顯示排程調用的結果。

- 清理示範期間建立的所有資源。

這個範例在 GitHub 上的檢視效果最佳。如需完整的原始碼和如何設定及執行的指示，請參閱 [GitHub](#) 上的完整範例。

此範例中使用的服務

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Amazon SNS 的無伺服器範例

下列程式碼範例示範如何使用 Amazon SNS 搭配 AWS SDKs。

範例

- [使用 Amazon SNS 觸發條件調用 Lambda 函數](#)

使用 Amazon SNS 觸發條件調用 Lambda 函數

下列程式碼範例顯示如何實作 Lambda 函數，以便接收在收到來自 SNS 主題的訊息時觸發的事件。函數會從事件參數擷取訊息，並記錄每一則訊息的內容。

.NET

適用於 .NET 的 SDK

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 .NET 搭配 Lambda 來使用 SNS 事件。

```
// 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 for Go V2

 Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 Go 搭配 Lambda 來使用 SNS 事件。

```
// 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 for Java 2.x

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 Java 搭配 Lambda 來使用 SNS 事件。

```
// 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 for JavaScript (v3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 JavaScript 搭配 Lambda 來使用 SNS 事件。

```
// 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;
  }
}
```

使用 TypeScript 搭配 Lambda 來使用 SNS 事件。

```
// 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 for PHP

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 PHP 搭配 Lambda 來使用 SNS 事件。

```
// 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 for Python (Boto3)

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 Python 搭配 Lambda 來使用 SNS 事件。

```
# 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 for Ruby

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 Ruby 搭配 Lambda 來使用 SNS 事件。

```
# 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 for Rust

Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在[無伺服器範例](#)儲存庫中設定和執行。

使用 Rust 搭配 Lambda 來使用 SNS 事件。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
// aws_lambda_events = { version = "0.10.0", default-features = false, features
//   = ["sns"] }
// lambda_runtime = "0.8.1"
// tokio = { version = "1", features = ["macros"] }
// tracing = { version = "0.1", features = ["log"] }
// tracing-subscriber = { version = "0.3", default-features = false, features =
//   ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);

    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用 Amazon SNS](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

Amazon SNS 安全性

AWS [共同責任模型](#)適用於 Amazon Simple Notification Service 中的資料保護。如此模型所述，AWS 負責保護執行所有的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。此內容包含您使用 AWS 之服務的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，請參閱[資料隱私權常見問答集](#)。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 [AWS 共同的責任模型和 GDPR](#) 部落格文章。

基於資料保護目的，我們建議您保護 AWS 帳戶登入資料，並使用 AWS Identity and Access Management (IAM) 設定個別使用者帳戶。如此一來，每個使用者都只會獲得授予完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。建議使用 TLS 1.2 或更新版本。
- 使用設定 API 和使用使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及服務中的所有 AWS 預設安全控制。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。
- 如果您在 AWS 透過命令列界面或 API 存取時需要 FIPS 140-2 驗證的密碼編譯模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱[聯邦資訊處理標準 \(FIPS\) 140-2 概觀](#)。
- 訊息資料保護
 - 訊息資料保護是 Amazon SNS 的新主要功能
 - 使用 MDP 掃描郵件中的機密或敏感資訊
 - 為流經主題的所有內容提供訊息稽核
 - 針對發佈至主題的訊息和主題傳送的訊息提供內容存取控制

Important

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的欄位中，例如名稱欄位。這包括當您使用主控台 AWS CLI、AWS API 或 SDKs 來使用 Amazon SNS 或其他 Amazon Web Services 時。您在標籤或自由格式欄位中輸入的任何資料都可能用於計費或診斷記錄。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

Amazon SNS 資料加密

資料保護是指保護往返 Amazon SNS 的傳輸中資料，以及存放在 Amazon SNS 資料中心內磁碟的靜態資料。您可以透過 Secure Sockets Layer (SSL) 或用戶端加密來保護傳輸中的資料。根據預設，Amazon SNS 會使用磁碟加密來存放訊息和檔案。您可以要求 Amazon SNS 先加密訊息，再將訊息儲存至其資料中心的加密檔案系統，以保護靜態資料。Amazon SNS 建議使用 SSE 來最佳化資料加密。

使用伺服器端加密保護 Amazon SNS 資料

伺服器端加密 (SSE) 可讓您使用 AWS Key Management Service () 中管理的金鑰來保護 Amazon SNS 主題中的訊息內容，以將敏感資料存放在加密主題中 AWS KMS。

Amazon SNS 一收到訊息，SSE 就會將其加密。訊息會以加密形式儲存，且只在傳送時才會解密。

- 如需使用 AWS Management Console 或管理 SSE 的資訊適用於 Java 的 AWS SDK（透過使用 [CreateTopic](#) 和 [SetTopicAttributes](#) API 動作設定 KmsMasterKeyId 屬性），請參閱 [使用伺服器端加密設定 Amazon SNS 主題加密](#)。
- 如需使用 AWS CloudFormation（透過使用 [AWS::SNS::Topic](#) 資源設定 KmsMasterKeyId 屬性）建立加密主題的詳細資訊，請參閱 AWS CloudFormation 使用者指南。

Important

所有對啟用 SSE 之主題的請求都必須使用 HTTPS 和 [簽章版本 4](#)。

如需有關其他服務與加密主題之間相容性的資訊，請參閱您的服務說明文件。

Amazon SNS 只支援對稱加密 KMS 金鑰。您無法使用任何其他類型的 KMS 金鑰來加密服務資源。如需判斷 KMS 金鑰是否為對稱加密金鑰的說明，請參閱 [識別非對稱 KMS 金鑰](#)。

AWS KMS 結合安全、高可用性的硬體和軟體，提供針對雲端擴展的金鑰管理系統。當您搭配使用 Amazon SNS 時 AWS KMS，加密訊息資料的 [資料金鑰](#) 也會加密，並使用其保護的資料存放。

以下為使用 AWS KMS 的優點：

- 您可以自行建立和管理 [AWS KMS key](#) 金鑰。
- 您也可以使用 AWS Amazon SNS 的受管 KMS 金鑰，這些金鑰對於每個帳戶和區域都是唯一的。
- AWS KMS 安全標準可協助您符合加密相關的合規要求。

如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[什麼是 AWS Key Management Service ?](#)。

加密範圍

SSE 會加密 Amazon SNS 主題中的訊息內文。

SSE 不會加密下列項目：

- 主題中繼資料 (主題名稱和屬性)
- 訊息中繼資料 (主旨、訊息 ID、時間戳記和屬性)
- 資料保護政策
- 每個主題指標

Note

- 只有在啟用加密主題後傳送的訊息，才會對訊息進行加密。Amazon SNS 不會加密待處理訊息。
- 即使已停用其主題的加密，已加密訊息仍會維持加密。

重要用語

以下重要術語有助於您更加了解 SSE 的功能。如需描述的詳細資訊，請參閱 [Amazon Simple Notification Service API 參考](#)。

資料金鑰

資料加密金鑰 (DEK) 負責加密 Amazon SNS 訊息的內容。

如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的[資料金鑰](#)，以及 AWS Encryption SDK 開發人員指南中的[信封加密](#)。

AWS KMS key ID

別名、別名 ARN、金鑰 ID 或的金鑰 ARN AWS KMS key，或自訂 AWS KMS- 在您的 帳戶或其他帳戶中。雖然 AWS KMS Amazon SNS AWS 受管 的別名一律為 `alias/aws/sns`，但自訂的別名 AWS KMS 可以是 `alias/MyAlias`。您可以使用這些 AWS KMS 金鑰來保護 Amazon SNS 主題中的訊息。

 Note

請謹記以下幾點：

- 第一次使用 AWS Management Console 為主題指定 Amazon SNS 的 AWS 受管 KMS 時，會 AWS KMS 建立 Amazon SNS 的 AWS 受管 KMS。
- 或者，當您第一次在已啟用 SSE 的主題上使用 Publish 動作時，會 AWS KMS 建立 Amazon SNS 的 AWS 受管 KMS。

您可以建立 AWS KMS 金鑰、定義控制如何使用 AWS KMS 金鑰的政策，以及使用 AWS KMS 主控台的 AWS KMS keys 區段或 [CreateKey](#) AWS KMS 動作來稽核 AWS KMS 用量。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [AWS KMS keys](#) 和 [建立金鑰](#)。如需識別 AWS KMS 符的更多範例，請參閱 AWS Key Management Service API 參考中的 [KeyId](#)。如需尋找 AWS KMS 識別符的資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [尋找金鑰 ID 和 ARN](#)。

 Important

使用需支付額外費用 AWS KMS。如需詳細資訊，請參閱 [估算 AWS KMS 成本](#) 和 [AWS Key Management Service 定價](#)。

管理 Amazon SNS 加密金鑰和成本

以下各節提供有關使用 AWS Key Management Service (AWS KMS) 中所管理金鑰的資訊。

 Note

Amazon SNS 只支援對稱加密 KMS 金鑰。您無法使用任何其他類型的 KMS 金鑰來加密服務資源。如需判斷 KMS 金鑰是否為對稱加密金鑰的說明，請參閱 [識別非對稱 KMS 金鑰](#)。

估算 AWS KMS 成本

為了預測成本並更好地了解您的 AWS 帳單，您可能想知道 Amazon SNS 使用您的 的頻率 AWS KMS key。

Note

下列公式可以提供預期成本的良好概念，但由於 Amazon SNS 的分佈特性，實際成本可能會比較高。

若要計算每個主題的 API 請求數量 (R)，請使用下列公式：

$$R = B / D * (2 * P)$$

B 是帳單週期 (以秒為單位)。

D 是資料金鑰重複使用期間 (以秒為單位，Amazon SNS 重複使用資料金鑰最多 5 分鐘)。

P 是發布的委託人數量，傳送至 Amazon SNS 主題。

以下為計算範例。如需確切的定價資訊，請參閱 [AWS Key Management Service 定價](#)。

範例 1：計算 1 個發佈者和 1 個主題的 AWS KMS API 呼叫數量

此範例假設如下：

- 帳單週期是 1 月 1-31 日 (2,678,400 秒)。
- 資料金鑰重複使用期間為 5 分鐘 (300 秒)。
- 有 1 個主題。
- 有 1 個發布委託人。

$$2,678,400 / 300 * (2 * 1) = 17,856$$

範例 2：計算多個發布者和 2 個主題的 AWS KMS API 呼叫數量

此範例假設如下：

- 帳單週期是 2 月 1-28 日 (2,419,200 秒)。
- 資料金鑰重複使用期間為 5 分鐘 (300 秒)。
- 有 2 個主題。
- 第一個主題有 3 個發布委託人。
- 第二個主題有 5 個發布委託人。

$$(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024$$

設定 AWS KMS 許可

在使用 SSE 之前，您必須設定 AWS KMS key 政策以允許主題加密和訊息加密和解密。如需範例和有關 AWS KMS 權限的詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的 [AWS KMS API 許可：動作和資源參考](#)。如需如何設定使用伺服器端加密的 Amazon SNS 主題的詳細資訊，請參閱 [其他資訊](#)。

Note

您也可以使用 IAM 政策管理對稱加密 KMS 金鑰的許可。如需詳細資訊，請參閱[搭配使用 IAM 政策 AWS KMS](#)。

雖然您可以設定全域許可來向 Amazon SNS 進行傳送和接收，但 AWS KMS 需要您替 IAM 政策 Resource 區段中特定區域裡的 KMS 完整 ARN 明確命名。

您也必須確保的金鑰政策 AWS KMS key 允許必要的許可。若要這樣做，請將在 Amazon SNS 中產生和消費加密訊息的委託人命名為 KMS 金鑰政策中的使用者。

或者，您可以在指派給發佈和訂閱以接收 Amazon SNS 中加密訊息之主體的 IAM 政策中指定必要的 AWS KMS 動作和 KMS ARN。如需詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的[管理 AWS KMS 的存取權](#)。

如果您為 Amazon SNS 主題選取客戶管理金鑰，且您使用別名來透過 IAM 政策或具有條件金鑰 `kms:ResourceAliases` 的 KMS 金鑰政策來控制 KMS 金鑰的存取權，請確保所選客戶管理的金鑰也具有關聯的別名。如需有關使用別名來控制 KMS 金鑰存取權的詳細資訊，請參閱 AWS Key Management Service 開發人員指南中的[使用別名控制 KMS 金鑰的存取權](#)。

允許使用者透過 SSE 傳送訊息至主題

發布者對於 AWS KMS key 必須擁有 `kms:GenerateDataKey*` 和 `kms:Decrypt` 許可。

```
{
  "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"
  ]
}
```

啟用來自 AWS 服務和加密主題的事件來源之間的相容性

數個 AWS 服務會將事件發佈至 Amazon SNS 主題。若要允許這些事件來源可用於加密主題，您必須執行以下步驟。

1. 使用客戶受管金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[建立索引鍵](#)。
2. 若要允許 AWS 服務擁有 kms:GenerateDataKey* 和 kms:Decrypt 許可，請將下列陳述式新增至 KMS 政策。

```

{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  ]
}
```

事件來源	服務主體
Amazon CloudWatch	cloudwatch.amazonaws.com
Amazon CloudWatch Events	events.amazonaws.com

事件來源	服務主體
AWS CodeCommit	codecommit.amazonaws.com
AWS Database Migration Service	dms.amazonaws.com
AWS Directory Service	ds.amazonaws.com
Amazon DynamoDB	dynamodb.amazonaws.com
Amazon Inspector	inspector.amazonaws.com
Amazon Redshift	redshift.amazonaws.com
Amazon RDS	events.rds.amazonaws.com
Amazon S3 Glacier	glacier.amazonaws.com
Amazon Simple Email Service	ses.amazonaws.com
Amazon Simple Storage Service	s3.amazonaws.com
AWS Snowball Edge	importexport.amazonaws.com
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager 包含兩個服務原則： ssm-incidents.amazonaws.com ； ssm-contacts.amazonaws.com

Note

有些 Amazon SNS 事件來源會要求您在 AWS KMS key 政策中提供 IAM 角色（而不是服務主體）：

- [Amazon EC2 Auto Scaling](#)
- [Amazon Elastic Transcoder](#)
- [AWS CodePipeline](#)
- [AWS Config](#)
- [AWS Elastic Beanstalk](#)

- [AWS IoT](#)
- [EC2 Image Builder](#)

3. 將 `aws:SourceAccount` 和 `aws:SourceArn` 條件金鑰新增至 KMS 資源政策，以進一步保護 KMS 金鑰免遭[混淆代理人](#)攻擊。有關每種情況的確切詳細資訊，請參閱服務專屬文件清單（上方）。

 Important

EventBridge-to-encrypted主題不支援將 `aws:SourceAccount`、`aws:SourceArn`、和 `aws:SourceOrgID` 新增至 AWS KMS 政策。

```
{
  "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-customer-resource-id"
    }
  }
}
```

4. [啟用主題的 SSE](#) 使用您的 KMS。
5. 提供加密主題的 ARN 至事件來源。

AWS KMS 錯誤

當您使用 Amazon SNS 和 時 AWS KMS，您可能會遇到錯誤。以下清單說明錯誤和可能的故障診斷解決方案。

KMSAccessDeniedException

加密文字參考不存在或您無法存取的金鑰。

HTTP 狀態碼：400

KMSDisabledException

由於指定的 KMS 未啟用，因此請求遭到拒絕。

HTTP 狀態碼：400

KMSInvalidStateException

由於所指定資源的狀態對於此請求無效，因此請求遭到拒絕。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [AWS KMS keys 的金鑰狀態](#)。

HTTP 狀態碼：400

KMSNotFoundException

由於找不到指定的實體或資源，因此請求遭到拒絕。

HTTP 狀態碼：400

KMSOptInRequired

AWS 存取金鑰 ID 需要 服務的訂閱。

HTTP 狀態碼：403

KMSThrottlingException

由於請求調節，因此請求遭到拒絕。如需限流的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 [配額](#)。

HTTP 狀態碼：400

使用伺服器端加密設定 Amazon SNS 主題加密

Amazon SNS 支援伺服器端加密 (SSE)，以使用 AWS Key Management Service () 保護訊息的內容 AWS KMS。請依照下列指示，使用 Amazon SNS 主控台或 CDK 啟用 SSE。

選項 1：使用 啟用加密 AWS Management Console

1. 登入 [Amazon SNS 主控台](#)。
2. 導覽至主題頁面，選取您的主題，然後選擇編輯。
3. 展開 Encryption (加密) 區段並執行下列動作：
 - 將加密切換為啟用。
 - 選取 AWS 受管SNS金鑰 (alias/aws/sns) 做為加密金鑰。預設會選取此項。
4. 選擇 Save changes (儲存變更)。

Note

- 如果 不存在，AWS 受管金鑰 則會自動建立。
- 如果您沒有看到金鑰或許可不足，請向管理員詢問 `kms:ListAliases`和 `kms:DescribeKey`。

選項 2：使用 啟用加密 AWS CDK

若要在 CDK 應用程式中使用 AWS 受管SNS 金鑰，請新增下列程式碼片段：

```
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();
    }
}
```


其他資訊

- 自訂 KMS 金鑰 – 您可以視需要指定自訂金鑰。在 Amazon SNS 主控台中，從清單中選取您的自訂 KMS 金鑰或輸入 ARN。
- 自訂 KMS 金鑰的許可 – 如果使用自訂 KMS 金鑰，請在金鑰政策中包含下列項目，以允許 Amazon SNS 加密和解密訊息：

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

對消費者的影響

啟用 SSE 不會變更訂閱者以透明方式使用 message. AWS manages 加密和解密的方式。訊息會保持靜態加密，並在交付給訂閱者之前自動解密。為了獲得最佳安全性，AWS 建議為所有端點啟用 HTTPS，以確保訊息的安全傳輸。

使用加密的 Amazon SQS 佇列訂閱設定 Amazon SNS 主題加密 Amazon SQS

您可以對主題啟用伺服器端加密 (SSE) 來保護其資料。若要允許 Amazon SNS 將訊息傳送到加密的 Amazon SQS 佇列，與 Amazon SQS 佇列相關聯的客戶受管金鑰必須具有政策陳述式，以授予

Amazon SNS 服務主體對 AWS KMS API 動作 `GenerateDataKey` 和 `Decrypt` 的存取權。如需使用 SSE 的詳細資訊，請參閱 [使用伺服器端加密保護 Amazon SNS 資料](#)。

本主題說明如何使用 為具有加密 Amazon SQS 佇列訂閱的 Amazon SNS Amazon SQS AWS Management Console。

步驟 1：建立自訂 KMS 金鑰

1. 用至少具有 `AWSKeyManagementServicePowerUser` 政策的使用者登入 [AWS KMS 主控台](#)。
2. 選擇 `Create a key` (建立金鑰)。
3. 若要建立對稱加密 KMS 金鑰，在 `Key type` (金鑰類型) 欄位中，選擇 `Symmetric` (對稱)。

如需如何在 AWS KMS 主控台中建立非對稱 KMS 金鑰的相關資訊，請參閱[建立非對稱 KMS 金鑰 \(主控台\)](#)。

4. 在 `Key usage` (金鑰用途) 欄位中，系統會自動選取 `Encrypt and decrypt` (加密和解密) 選項。

如需如何建立可以產生和驗證 MAC 代碼的 KMS 金鑰的相關資訊，請參閱[建立 HMAC KMS 金鑰](#)。

如需進階選項的相關資訊，請參閱[特殊用途金鑰](#)。

5. 選擇 `Next` (下一步)。
6. 輸入 KMS 金鑰的別名。別名名稱的開頭不可以是 `aws/`。`aws/` 字首由 Amazon Web Services 保留，以在您的帳戶 AWS 受管金鑰 中代表。

Note

新增、刪除或更新別名可允許或拒絕 KMS 金鑰的許可。如需詳細資訊，請參閱 [ABAC for AWS KMS](#) 和 [使用別名來控制對 KMS 金鑰的存取](#)。

別名是您可用來識別 KMS 金鑰的顯示名稱。我們建議您選擇別名來表示您計劃保護的資料類型，或您計劃搭配 KMS 金鑰一起使用的應用程式。

在 AWS Management Console 中建立 KMS 金鑰時需要別名。但在使用 [CreateKey](#) 操作時是選用的。

7. (選用) 輸入 KMS 金鑰的描述。

您可以立即新增描述或在任意時間更新，除非**金鑰狀態**為 Pending Deletion 或 Pending Replica Deletion。若要新增、變更或刪除現有客戶受管金鑰的描述，[請在中編輯描述](#) AWS Management Console 或使用 [UpdateKeyDescription](#) 操作。

8. (選用) 輸入標籤索引鍵和選用標籤值。若要將其他標籤新增至 KMS 金鑰，請選擇 Add tag (新增標籤)。

Note

標記或取消標記 KMS 金鑰可以允許或拒絕 KMS 金鑰的許可。如需詳細資訊，請參閱 [ABAC for AWS KMS](#) 和 [使用標籤來控制對 KMS 金鑰的存取](#)。

當您將標籤新增至 AWS 資源時，AWS 會產生成本分配報告，其中包含依標籤彙總的用量和成本。標籤也可以用來控制 KMS 金鑰的存取。如需標記 KMS 金鑰的詳細資訊，請參閱 [標記金鑰](#) 和 [ABAC for AWS KMS](#)。

9. 選擇 Next (下一步)。
10. 選取可管理 KMS 金鑰的 IAM 使用者和角色。

Note

此金鑰政策提供此 KMS 金鑰的完整 AWS 帳戶 控制權。它允許帳戶管理員使用 IAM 政策授予其他主體管理 KMS 金鑰的許可。如需詳細資訊，請參閱 [預設金鑰政策](#)。

IAM 最佳實務不建議使用具有長期憑證的 IAM 使用者。盡可能使用提供臨時憑證的 IAM 角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 [IAM 中的安全最佳實務](#)。

11. (選用) 為了防止選取的 IAM 使用者和角色刪除此 KMS 金鑰，請在頁面底部的 Key deletion (金鑰刪除) 區段中，清除 Allow key administrators to delete this key (允許金鑰管理員刪除此金鑰) 核取方塊。
12. 選擇 Next (下一步)。
13. 選取可將金鑰用於 [密碼編譯操作](#) 的 IAM 使用者和角色 選擇 Next (下一步)。
14. 在 Review and edit key policy (檢閱和編輯金鑰政策) 頁面，請將以下陳述式新增至金鑰政策，然後選擇 Finish (完成)。

```
{  
  "Sid": "Allow Amazon SNS to use this key",
```

```
"Effect": "Allow",
"Principal": {
  "Service": "sns.amazonaws.com"
},
"Action": [
  "kms:Decrypt",
  "kms:GenerateDataKey*"
],
"Resource": "*"
}
```

新的客戶受管金鑰會顯示在金鑰清單中。

步驟 2：建立加密的 Amazon SNS 主題

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。
3. 請選擇建立主題。
4. 在 Create new topic (建立新主題) 頁面的 Name (名稱) 中，輸入主題名稱 (例如 MyEncryptedTopic)，然後選擇 Create topic (建立主題)。
5. 展開 Encryption (加密) 區段並執行下列動作：
 - a. 選擇 Enable server-side encryption (啟用伺服器端加密)。
 - b. 指定客戶受管金鑰。如需詳細資訊，請參閱[重要用語](#)。

對於每個客戶受管金鑰類型，都會顯示描述、帳戶和客戶受管金鑰 ARN。

Important

如果您並非客戶受管金鑰的擁有者，或者您登入時所用的帳戶並無 `kms:ListAliases` 和 `kms:DescribeKey` 許可，則無法在 Amazon SNS 主控台上檢視客戶受管金鑰的相關資訊。

請要求客戶受管金鑰的擁有者授予您這些許可。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[AWS KMS API 許可：動作和資源參考](#)。

- c. 對於客戶受管金鑰，選擇[您之前建立的](#) MyCustomKey，然後選擇啟用伺服器端加密。
6. 選擇 Save changes (儲存變更)。

將會對您的主題啟用 SSE，並顯示 MyTopic (我的主題) 頁面。

主題的加密狀態、AWS 帳戶、客戶受管金鑰、客戶受管金鑰 ARN，以及描述會顯示在加密索引標籤上。

您的新加密主題會顯示在主題清單。

步驟 3：建立和訂閱加密的 Amazon SQS 佇列

1. 請登入 [Amazon SQS 主控台](#)。
2. 請選擇 Create New Queue (建立新佇列)。
3. 在 Create New Queue (建立新佇列) 頁面上，執行下列操作：
 - a. 輸入 Queue Name (佇列名稱) (例如，MyEncryptedQueue1)。
 - b. 選擇 Standard Queue (標準佇列)，然後選擇 Configure Queue (設定佇列)。
 - c. 選擇 Use SSE (使用 SSE)。
 - d. 對於 AWS KMS key，選擇[您之前建立的](#) MyCustomKey，然後選擇建立佇列。
4. 重複進行以上程序，以建立第二個佇列 (例如，名為 MyEncryptedQueue2)。

您的新加密佇列會顯示在佇列清單。

5. 在 Amazon SQS 主控台，選擇 MyEncryptedQueue1 和 MyEncryptedQueue2 然後選擇 Queue Actions (佇列動作)、Subscribe Queues to SNS Topic (訂閱佇列至 SNS 主題)。
6. 在 Subscribe to a Topic (訂閱主題) 對話方塊中，於 Choose a Topic (選擇主題) 中選取 MyEncryptedTopic，然後選擇 Subscribe (訂閱)。

您對加密主題的加密佇列訂閱會顯示在 Topic Subscription Result (主題訂閱結果) 對話方塊中。

7. 選擇 OK (確定)。

步驟 4：將訊息發佈至您的加密主題

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽面板上，選擇 Topics (主題)。
3. 從主題清單中，選擇 MyEncryptedTopic，然後選擇 Publish message (發佈訊息)。
4. 在 Publish a message (發佈訊息) 頁面上，執行下列步驟：

- a. (選用) 在 Message details (訊息詳細資訊) 區段中，輸入 Subject (主旨) (例如，Testing message publishing)。
- b. 在 Message body (訊息內文) 區段中，輸入訊息內文 (例如，My message body is encrypted at rest.)。
- c. 選擇 Publish message (發佈訊息)。

您的訊息會發佈到您的訂閱加密佇列。

步驟 5：驗證訊息交付

1. 請登入 [Amazon SQS 主控台](#)。
2. 從佇列清單中，選擇 MyEncryptedQueue1，然後選擇 Send and receive messages (傳送及接收訊息)。
3. 在 Send and receive messages in MyEncryptedQueue1 (在 MyEncryptedQueue1 中傳送和接收訊息) 頁面上，選擇 Poll for messages (輪詢訊息)。

此時將會顯示 [您稍早傳送的](#) 訊息。

4. 選擇 More Details (更多詳細資訊) 可檢視您的訊息。
5. 完成後，請選擇 Close (關閉)。
6. 針對 MyEncryptedQueue2 重複此程序。

使用 VPC 端點保護 Amazon SNS 流量

適用於 Amazon SNS 的 Amazon Virtual Private Cloud (Amazon VPC) 端點是 VPC 中的邏輯實體，僅允許連線到 Amazon SNS。VPC 會將請求路由至 Amazon SNS，並將回應路由回 VPC。以下各節提供使用 VPC 端點以及建立 VPC 端點政策的相關資訊。

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 託管 AWS 資源，您可以在 VPC 和 Amazon SNS 之間建立私有連線。使用此連線，您可以將訊息發佈到 Amazon SNS 主題，而不需透過公有網際網路。

Amazon VPC 是一項 AWS 服務，可用來在您定義的虛擬網路中啟動 AWS 資源。您可利用 VPC 來控制您的網路設定，例如 IP 地址範圍、子網路、路由表和網路閘道。若要將您的 VPC 連接到 Amazon SNS，請定義介面 VPC 端點。這種類型的端點可讓您將 VPC 連線至 AWS 服務。端點能為 Amazon SNS 提供可靠、可擴展性的連線，無須使用網際網路閘道、網路位址轉譯 (NAT) 執行個體或 VPN 連接。如需詳細資訊，請參閱 [《Amazon VPC 使用者指南》中的 AWS 服務 使用介面 VPC 端點存取](#)。

本節資訊適用於 Amazon VPC 使用者。如需詳細資訊並開始使用建立 VPC，請參閱《[Amazon VPC 使用者指南](#)》中的[規劃 VPC](#)。

 Note

VPC 端點不允許您訂閱私有 IP 地址的 Amazon SNS 主題。

為 Amazon SNS 建立 Amazon VPC 端點

若要從 Amazon VPC 將訊息發佈至 Amazon SNS 主題，請建立一個界面 VPC 端點。然後，您可以將訊息發佈到主題，同時讓您的流量維持在 VPC 管理的網路中。

使用以下資訊來建立端點，並測試 VPC 和 Amazon SNS 之間的連線。或者，請參閱 [從 Amazon VPC 發佈 Amazon SNS 訊息](#)，以取得協助您從零開始的逐步解說。

建立端點

您可以使用 AWS Management Console、AWS CLI、AWS SDK、Amazon SNS API 或在 VPC 中建立 Amazon SNS 端點 AWS CloudFormation。

如需使用 Amazon VPC 主控台或 AWS CLI，請參閱 Amazon VPC 使用者指南中的[建立界面端點](#)。

 Important

Amazon 虛擬私有雲端只能搭配 HTTPS Amazon SNS 端點使用。

建立端點時，請將 Amazon SNS 指定為您要 VPC 連接的服務。在 Amazon VPC 主控台中，服務名稱取決於所選區域。例如，如果您選擇美國東部 (維吉尼亞北部)，服務名稱會是 `com.amazonaws.us-east-1.sns`。

若您設定 Amazon SNS 從 Amazon VPC 傳送訊息，則必須啟用私有 DNS 並使用 `sns.us-east-2.amazonaws.com` 格式指定端點。

私有 DNS 不支援 `queue.amazonaws.com` 或 `us-east-2.queue.amazonaws.com` 之類的延遲端點。

如需有關使用 建立和設定端點的資訊 AWS CloudFormation，請參閱 AWS CloudFormation 《使用者指南》中的 [AWS::EC2::VPCEndpoint](#) 資源。

測試 VPC 和 Amazon SNS 之間的連線

為 Amazon SNS 建立端點後，您可以從 VPC 發佈訊息到 Amazon SNS 主題。若要測試此連線，請執行以下動作：

1. 連線至位於 VPC 中的 Amazon EC2 執行個體。如需連線的詳細資訊，請參閱 Amazon EC2 文件中的[連線至 Linux 執行個體](#)或[連線至 Windows 執行個體](#)。

例如，若要使用 SSH 用戶端連線到 Linux 執行個體，請在終端機中執行以下命令：

```
$ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
```

其中：

- *ec2-key-pair.pem* 是包含當您建立執行個體時 Amazon EC2 所提供金鑰對的檔案。
 - *instance-hostname* 是執行個體的公開主機名稱。若要在 [Amazon EC2 主控台](#) 中取得主機名稱：選擇執行個體、選擇執行個體，然後尋找公有 DNS 的值。
2. 從您的執行個體，使用 Amazon SNS [publish](#) 命令搭配 AWS CLI。您可以使用下列命令，傳送簡單的訊息到主題：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

其中：

- *aws-region* 是主題所在的 AWS 區域。
- *sns-topic-arn* 是主題的 Amazon 資源名稱 (ARN)。若要從 [Amazon SNS 主控台](#) 取得 ARN：選擇 Topics (主題)、找到主題，然後找到 ARN 欄中的值。

如果 Amazon SNS 成功接收訊息，終端會列印訊息 ID，看起來如下：

```
{
  "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

為 Amazon SNS 建立 VPC 端點政策

您可以為 Amazon SNS 的 Amazon VPC 端點建立政策，在其中您可以指定以下內容：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的[使用 VPC 端點控制服務的存取](#)。

以下範例 VPC 端點政策指定允許 IAM 使用者 MyUser 發佈到 Amazon SNS 主題 MyTopic。

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

拒絕以下各項：

- 其他 Amazon SNS API 動作，例如 sns:Subscribe 和 sns:Unsubscribe。
- 其他嘗試使用此 VPC 端點的 IAM 使用者和規則。
- MyUser 發佈訊息到不同的 Amazon SNS 主題。

Note

IAM 使用者仍然可以從外部 VPC 使用其他 Amazon SNS API 動作。

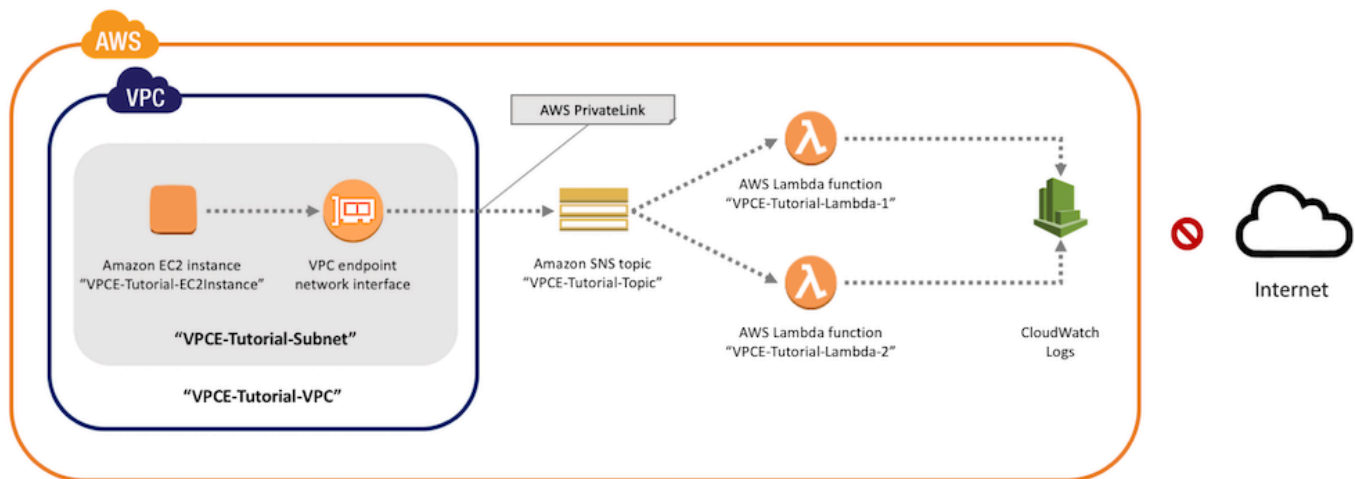
從 Amazon VPC 發佈 Amazon SNS 訊息

本節說明如何發佈訊息到 Amazon SNS 主題，同時在私有網路中保持安全。您從託管在 Amazon Virtual Private Cloud (Amazon VPC) 中的 Amazon EC2 執行個體發佈訊息。訊息會保留在 AWS 網路中，而不會傳輸公有網際網路。透過從 VPC 私下發佈訊息，您可以改善應用程式和 Amazon SNS 之間的流量安全性。當您發佈客戶的個人身分識別資訊 (PII)，或是當您的應用程式受限於市場法規時，此安全性是很重要的。例如，如果您的醫療系統必須遵守「健康保險流通與責任法案」(HIPAA)，或金融系統必須遵守「支付卡產業資料安全標準」(PCI DSS)，則私下發佈會很有幫助。

一般步驟如下：

- 使用 AWS CloudFormation 範本在您的 中自動建立暫時私有網路 AWS 帳戶。
- 以 Amazon SNS 建立與 VPC 連線的 VPC 端點。
- 登入到 Amazon EC2 執行個體並私下和發佈訊息到 Amazon SNS 主題。
- 確認該訊息已成功傳遞。
- 刪除您在此程序期間建立的資源，使其不會保留在您的 中 AWS 帳戶。

下圖說明您在完成這些步驟時在 AWS 帳戶中建立的私有網路：



這個網路包含一個 VPC，其中包含 Amazon EC2 執行個體。該執行個體透過介面 VPC 端點連接至 Amazon SNS。這種端點類型會連線至採用 AWS PrivateLink 技術的服務。建立此連接後，即使網路與公有網際網路中斷，您也可以登入 Amazon EC2 執行個體，並將訊息發佈到 Amazon SNS 主題。主題會向兩個訂閱 AWS Lambda 函數發送收到的訊息。這些函數會記錄它們在 Amazon CloudWatch Logs 中所收到的訊息。

完成這些步驟需約 20 分鐘。

主題

- [開始之前](#)
- [步驟 1：建立 Amazon EC2 金鑰對](#)
- [步驟 2：建立 AWS 資源](#)
- [步驟 3：確認您的 Amazon EC2 執行個體無法存取網際網路](#)
- [步驟 4：建立 Amazon SNS 的 Amazon VPC 端點](#)

- [步驟 5：發佈訊息到您的 Amazon SNS 主題](#)
- [步驟 6：驗證您的訊息交付](#)
- [步驟 7：清除](#)
- [相關資源](#)

開始之前

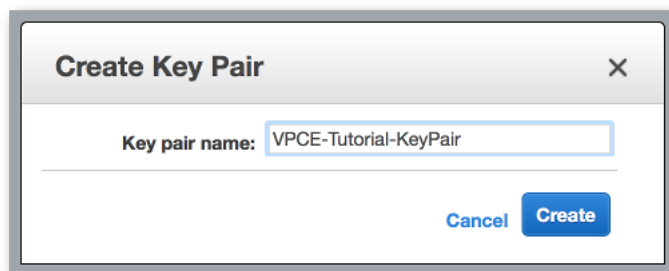
開始之前，您需要 Amazon Web Services (AWS) 帳戶。當您註冊時，您的帳戶會自動註冊 中的所有服務 AWS，包括 Amazon SNS 和 Amazon VPC。如果您尚未建立帳戶，請前往 <https://aws.amazon.com/>，然後選擇 Create a Free Account (建立免費帳戶)。

步驟 1：建立 Amazon EC2 金鑰對

金鑰對是用來登入 Amazon EC2 執行個體的。它包含用於加密您登入資訊的公有金鑰，以及用來解密的私有金鑰。當您建立金鑰對時，您會下載一份私有金鑰複本。稍後，您可以使用金鑰對登入 Amazon EC2 執行個體。若要登入，您需指定金鑰對名稱，並提供私有金鑰。

建立一組金鑰對

1. 登入 AWS Management Console 並開啟位於 <https://Amazon EC2 主控台>。 <https://console.aws.amazon.com/ec2/>
2. 在左側導覽功能表，找到 Network & Security (網路和安全) 部分。然後，選擇 Key Pairs (金鑰對)。
3. 選擇 Create Key Pair (建立金鑰對)。
4. 在 Create Key Pair (建立金鑰對) 視窗中，對於 Key pair name (金鑰對名稱)，輸入 **VPCE-Tutorial-KeyPair**。然後，選擇 Create (建立)。



5. 您的瀏覽器會自動下載私有金鑰檔案。將它存放在安全的地方。Amazon EC2 給予該檔案的副檔名為 .pem。

6. (選擇性) 如果您是在 Mac 或 Linux 電腦上使用 SSH 用戶端來連結到執行個體，請使用 `chmod` 下列命令來設定私有金鑰檔案的許可，即可確保只有您能夠讀取該檔案：

a. 開啟終端機並導覽至包含私有金鑰的目錄：

```
$ cd /filepath_to_private_key/
```

b. 使用以下命令來設定許可：

```
$ chmod 400 VPCE-Tutorial-KeyPair.pem
```

步驟 2：建立 AWS 資源

若要設定基礎設施，請使用 an AWS CloudFormation template。範本是一種檔案，是一種用來建置 Amazon EC2 執行個體與 Amazon SNS 主題之類等 AWS 資源的藍圖。在 GitHub 上有提供此流程的範本，供您下載。

您提供範本給 AWS CloudFormation，並 AWS CloudFormation 佈建在 中做為堆疊所需的資源 AWS 帳戶。堆疊是一組視為單一單位進行管理的資源。完成這些步驟後，您可以使用一次 AWS CloudFormation 刪除堆疊中的所有資源。除非您想要，AWS 帳戶否則這些資源不會保留在您的 中。

此過程的堆疊包含下列資源：

- 一個 VPC 和關聯的網路資源，包括子網路、安全群組、網際網路閘道和路由表。
- 在 VPC 中子網路啟動的 Amazon EC2 執行個體。
- Amazon SNS 主題。
- 兩個 AWS Lambda 函數。這些函數接收發佈到 Amazon SNS 主題的訊息，並在 CloudWatch Logs 中記錄事件。
- Amazon CloudWatch 指標和日誌。
- 允許 Amazon EC2 執行個體使用 Amazon SNS 的 IAM 角色，以及允許 Lambda 函數寫入至 CloudWatch Logs 的 IAM 角色。

建立 AWS 資源

1. 從 GitHub 網站下載[範本檔案](#)。
2. 登入 [AWS CloudFormation 主控台](#)。
3. 選擇 Create Stack (建立堆疊)。

4. 在 Select Template (選擇範本) 頁面中，選擇 Upload a template to Amazon S3 (上傳範本到 Amazon S3)、選擇檔案，並選擇 Next (下一步)。
5. 在 Specify Details (指定詳細資訊) 頁面，指定堆疊和金鑰名稱：
 - a. 在 Stack Name (堆疊名稱) 中輸入 **VPCE-Tutorial-Stack**。
 - b. 對於 KeyName (金鑰名稱)，選擇 VPCE-Tutorial-KeyPair (VPCE-教學-金鑰對)。
 - c. 對於 SSHLocation (SSH 位置)，保留預設值 **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. 選擇下一步。
6. 在 Options (選項) 頁面，保留所有預設值，然後選擇 Next (下一步)。
7. 請在 Review (檢閱) 頁面上確認堆疊詳細資訊。
8. 在功能下，確認 AWS CloudFormation 可能會建立具有自訂名稱的 IAM 資源。
9. 選擇建立。

AWS CloudFormation 主控台會開啟 Stacks 頁面。VPCE-Tutorial-Stack (VPCE-教學-堆疊) 的狀態為 CREATE_IN_PROGRESS (正在建立中)。在幾分鐘的時間內，狀態會在建立程序完成後變更為 CREATE_COMPLETE (建立完成)。

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

選擇 Refresh (重新整理) 按鈕以查看最新的堆疊狀態。

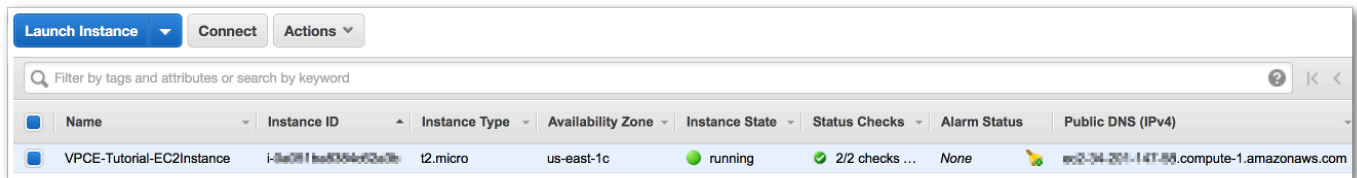
步驟 3：確認您的 Amazon EC2 執行個體無法存取網際網路

之前步驟中，在您的 VPC 啟動的 Amazon EC2 執行個體無法存取網際網路。它不允許傳出流量，而且無法發佈訊息到 Amazon SNS。請登入執行個體以驗證。然後，嘗試連接到公有端點，並嘗試傳送訊息至 Amazon SNS。

此時您的發佈嘗試失敗。後續步驟中，在您為 Amazon SNS 建立 VPC 端點後，您的發佈嘗試成功。

連線到您的 Amazon EC2 執行個體。

1. 前往 <https://console.aws.amazon.com/ec2/> 開啟 Amazon EC2 主控台。
2. 在左側導覽功能表，找到 Instances (執行個體) 部分。然後，選擇 Instances (執行個體)。
3. 在執行個體清單中，選取 VPCE-Tutorial-EC2Instance (VPCE-教學-EC2 執行個體)。
4. 複製公有 DNS 欄中提供的主機名稱。



5. 開啟終端機。從包含金鑰對的目錄中，使用以下命令連線到執行個體，其中 *instance-hostname* (執行個體主機名稱) 是您從 Amazon EC2 主控台複製的主機名稱：

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

驗證該執行個體沒有網際網路連線能力

- 在您的終端機中，嘗試連接到任何公有端點，例如 amazon.com：

```
$ ping amazon.com
```

由於連線嘗試失敗，您可以隨時取消 (在 Windows 上鍵入 Ctrl+C，或在 macOS 上鍵入 Command+C)。

確認該執行個體沒有連線至 Amazon SNS 的能力

1. 登入 [Amazon SNS 主控台](#)。
2. 在左側的導覽功能表中，選擇 Topics (主題)。

3. 在 Topics (主題) 頁面，為主題 VPCE-Tutorial-Topic (VPCE-教學-主題) 複製 Amazon 資源名稱 (ARN)。
4. 在您的終端機，嘗試向該主題發佈一則訊息：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

由於發佈嘗試失敗，您可以隨時取消。

步驟 4：建立 Amazon SNS 的 Amazon VPC 端點

將 VPC 連接到 Amazon SNS，請定義介面 VPC 端點。在您新增端點後，您便可在 VPC 中登入 Amazon EC2 執行個體，且從這裡您可使用 Amazon SNS API。您可以將訊息發佈到主題，該訊息為私下發佈。它們會留在 AWS 網路中，而且不會前往公有網際網路。

Note

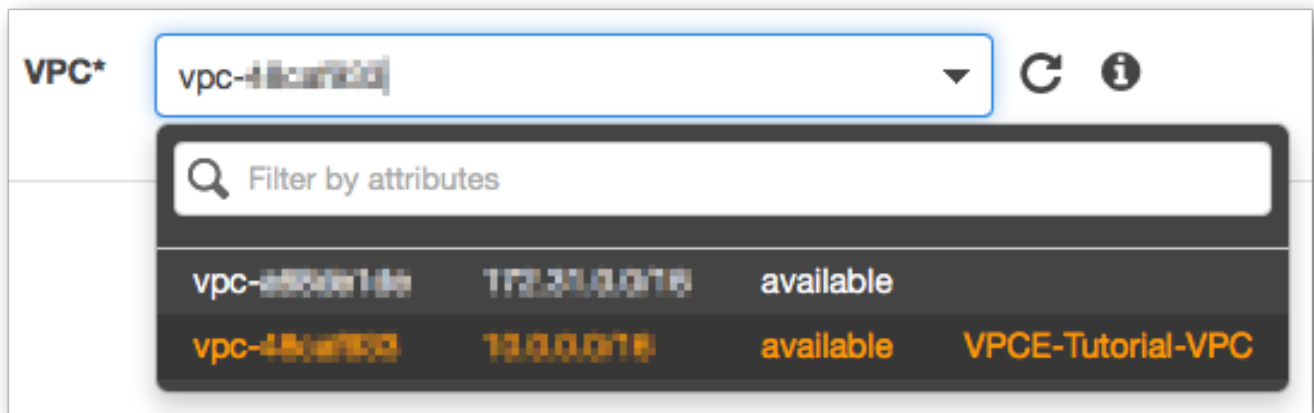
執行個體仍然無法存取網際網路上的 AWS 其他服務和端點。

建立端點

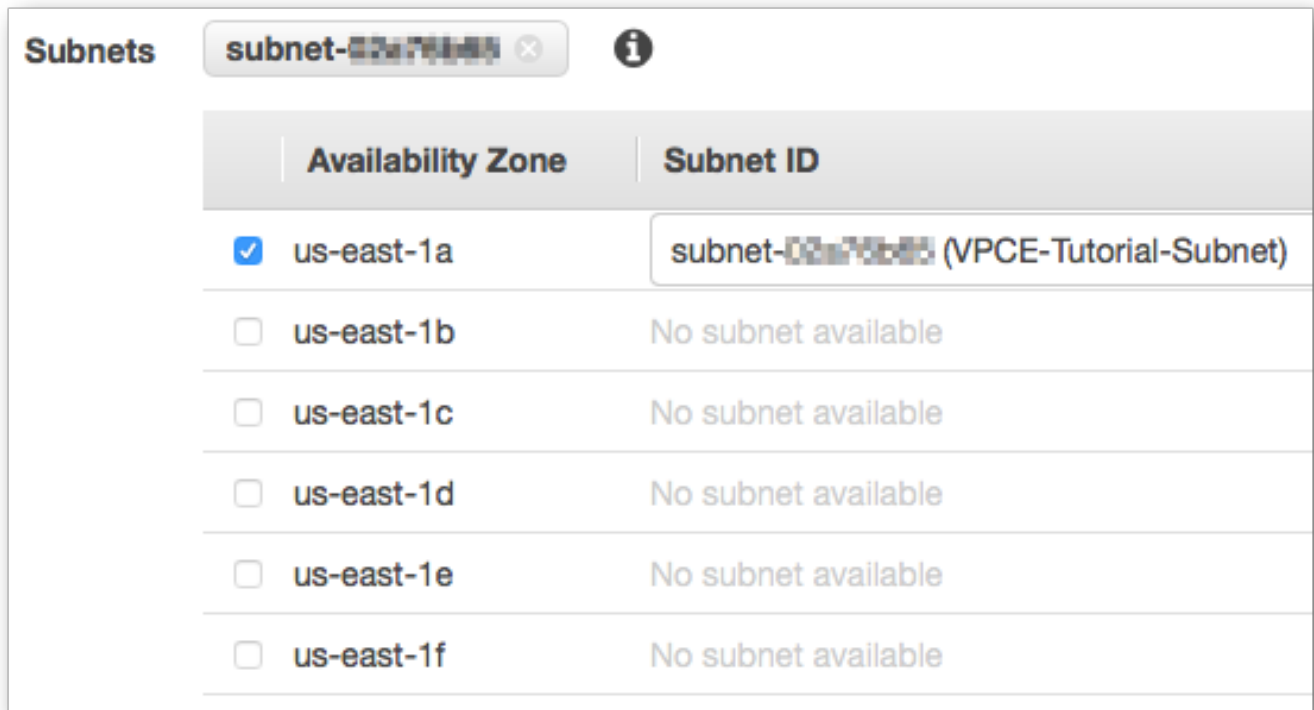
1. 在 <https://console.aws.amazon.com/vpc/> 開啟 Amazon VPC 主控台。
2. 在左側的導覽功能表中，選擇 Endpoints (端點)。
3. 選擇 Create Endpoint (建立端點)。
4. 在 Create Endpoint (建立端點) 頁面上，對於 Service category (服務類別)，保留 AWS services (服務) 的預設選擇。
5. 對於 Service Name (服務名稱)，選擇 Amazon SNS 服務名稱。

該服務名稱取決於所選區域。例如，如果您選擇美國東部 (維吉尼亞北部)，服務名稱會是 `com.amazonaws.us-east-1.sns`。

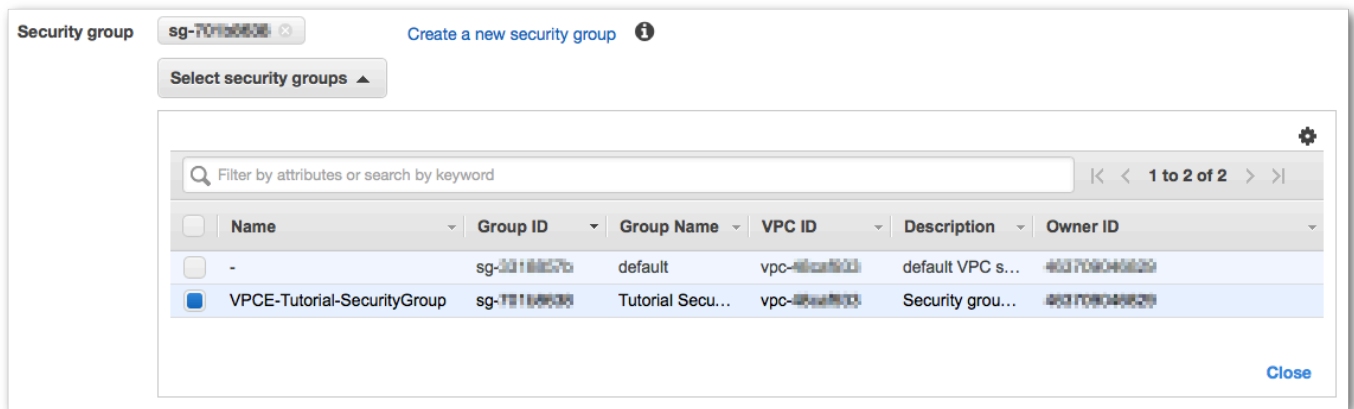
6. 對於 VPC，選擇名為 VPCE-Tutorial-VPC (VPCE-教學-VPC) 的 VPC。



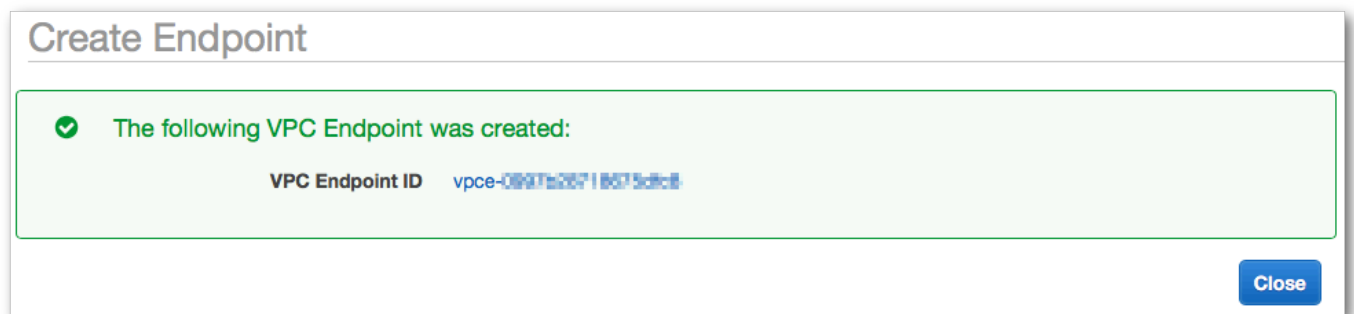
7. 對於 Subnets (子網路)，在子網路 ID 中選擇有 VPCE-Tutorial-Subnet 的子網路。



8. 對於 Enable Private DNS Name (啟用私有 DNS 名稱)，選擇 Enable for this endpoint (為此端點啟用)。
9. 對於 Security group (安全群組)，選擇 Select security group (選取安全群組)，並選擇 VPCE-Tutorial-SecurityGroup (VPCE-教學-安全群組)。

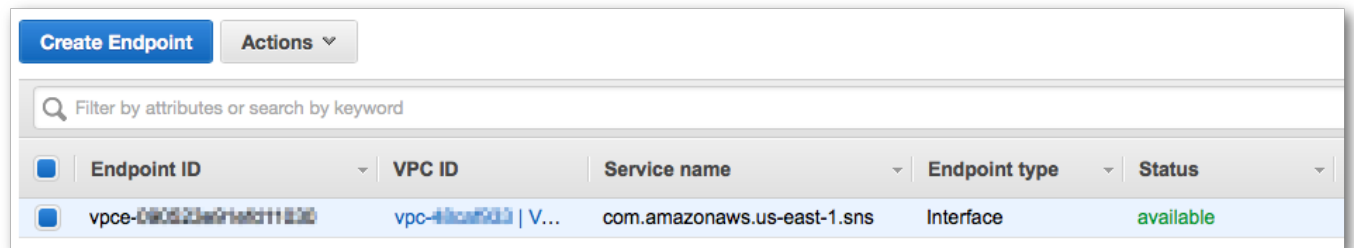


10. 選擇 Create endpoint (建立端點)。Amazon VPC 主控台確認 VPC 端點已建立。



11. 選擇關閉。

Amazon VPC 主控台開啟 Endpoints (端點) 頁面。新的端點狀態為 pending (等待中)。在幾分鐘的時間內，狀態會在建立程序完成後變更為 available (可用)。



步驟 5：發佈訊息到您的 Amazon SNS 主題

現在，您的 VPC 包含一個 Amazon SNS 端點，您可以登入 Amazon EC2 執行個體和將訊息發佈到主題。

發佈訊息

1. 如果您的終端機不再連接到您的 Amazon EC2 執行個體，再次連線：

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. 執行您之前的相同命令，以發佈訊息到 Amazon SNS 主題。此時發佈操作嘗試成功，並由 Amazon SNS 傳回訊息 ID：

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

{
  "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

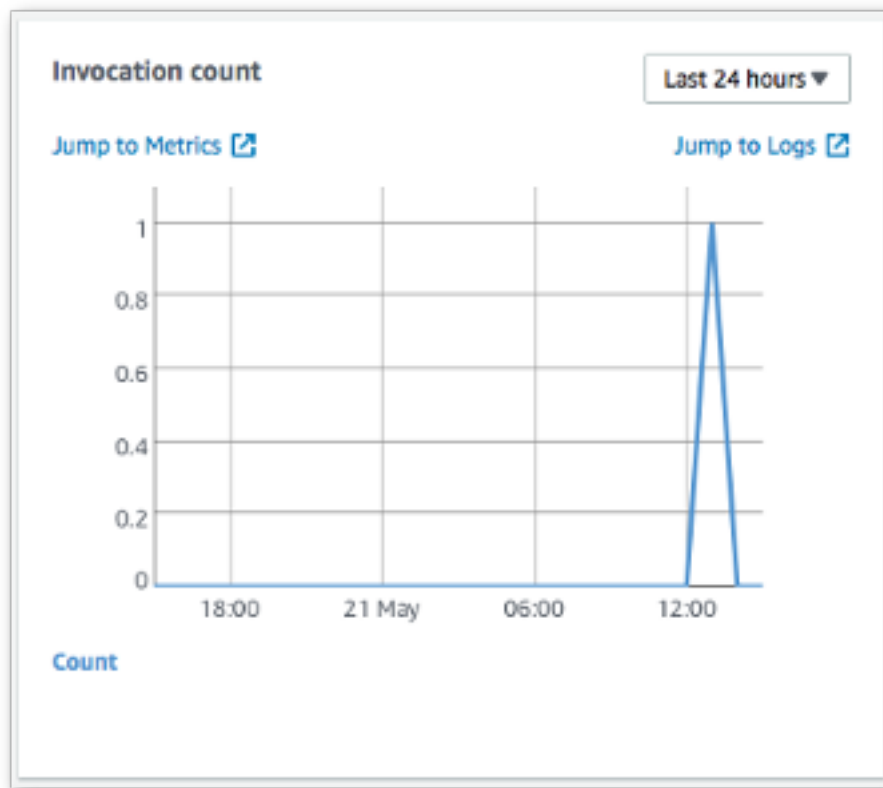
步驟 6：驗證您的訊息交付

當 Amazon SNS 主題接收到訊息時，它透過將訊息傳送給兩個訂閱 Lambda 函式來散發訊息。當這些函數接收訊息時，他們會記錄該事件至 CloudWatch Logs。為了驗證您的訊息交付成功，可檢查已呼叫的函數，並檢查 CloudWatch Logs 已更新。

確認 Lambda 函數已呼叫

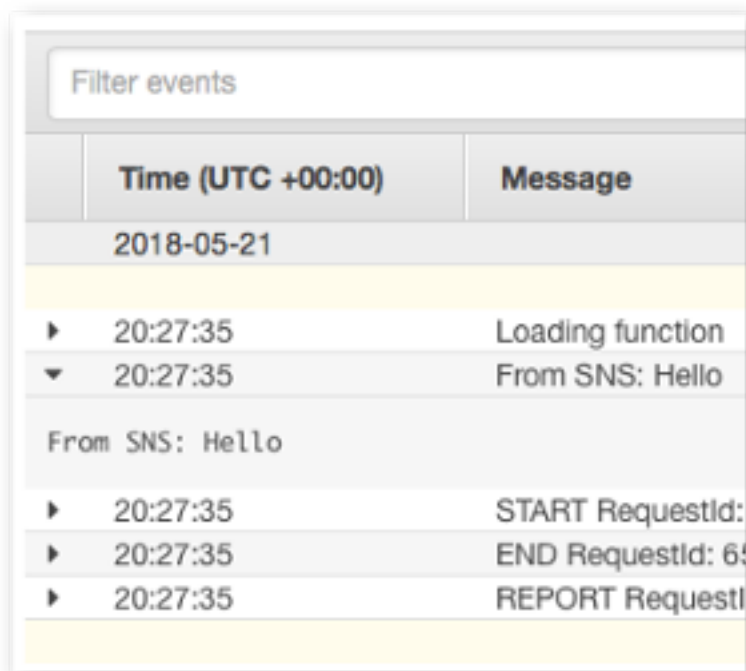
1. 開啟 AWS Lambda 主控台，網址為 <https://console.aws.amazon.com/lambda/>。
2. 在 Functions (函數) 頁面，選擇 VPCE-Tutorial-Lambda-1 (VPCE-教學-Lambda-1)。
3. 選擇 Monitoring (監控)。
4. 檢查 Invocation count (呼叫次數) 圖表。此圖表顯示 Lambda 函數已執行的次數。

叫用次數符合您發佈訊息到主題的發佈次數。



確認 CloudWatch Logs 已更新

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在左側的導覽功能表中，選擇 Logs (日誌)。
3. 檢查由 Lambda 函數所寫入的日誌：
 - a. 選擇 /aws/lambda/VPCE-Tutorial-Lambda-1/ 日誌群組。
 - b. 選擇日誌串流。
 - c. 檢查該日誌包含項目 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. 在主控台頂端選擇 Log Groups(日誌群組)，以返回 Log Groups (日誌群組) 頁面。然後，請為 /aws/lambda/VPCE-Tutorial-Lambda-2/ 日誌群組重複上述步驟。

恭喜您！透過為 Amazon SNS 新增端點至 VPC，您可以在由 VPC 管理的網路中發佈訊息到主題。私下發佈的訊息不會公開到公有網際網路。

步驟 7：清除

除非您想要保留建立的資源，否則您現在便可將它們刪除。透過刪除不再使用 AWS 的資源，您可以避免不必要的 費用 AWS 帳戶。

首先，請使用 Amazon VPC 主控台刪除您的 VPC 端點。然後，刪除您在 AWS CloudFormation 主控台中刪除堆疊所建立的其他資源。當您刪除堆疊時，AWS CloudFormation 會從 中移除堆疊的資源 AWS 帳戶。

刪除您的 VPC 端點

1. 在 <https://console.aws.amazon.com/vpc/> 開啟 Amazon VPC 主控台。
2. 在左側的導覽功能表中，選擇 Endpoints (端點)。
3. 選擇您建立的端點。
4. 選擇 Actions (動作)，然後選擇 Delete Endpoint (刪除端點)。

5. 在 Delete Endpoint (刪除端點) 視窗中，選擇 Yes, Delete (是的，刪除)。

該端點狀態變更為 deleting (刪除)。當刪除完成後，該端點便從頁面刪除。

刪除 AWS CloudFormation 堆疊

1. 開啟 AWS CloudFormation 主控台，網址為 <https://console.aws.amazon.com/cloudformation>。
2. 選取堆疊 VPCE-Tutorial-Stack (VPCE-教學-堆疊)。
3. 選擇 Actions (動作)，然後選擇 Delete Stack (刪除堆疊)。
4. 在 Delete Stack (刪除堆疊) 視窗中，選擇 Yes, Delete (是的，刪除)。

該堆疊狀態變更為 DELETE_IN_PROGRESS (正在刪除中)。當刪除完成後，該堆疊便從頁面刪除。

相關資源

如需詳細資訊，請參閱下列資源。

- [AWS 安全部落格：使用 AWS PrivateLink 保護發佈至 Amazon SNS 的訊息](#)
- [什麼是 Amazon VPC？](#)
- [VPC 端點](#)
- [什麼是 Amazon EC2？](#)
- [AWS CloudFormation 概念](#)

使用訊息資料保護增強 Amazon SNS 安全性

- [訊息資料保護](#)是 Amazon SNS 中的一項功能，用於定義您自己的規則和政策，以稽核和控制動態資料 (相對於靜態資料) 的內容。
- 訊息資料保護為以訊息為中心的企業應用程式提供控管、合規和稽核服務，因此 Amazon SNS 主題擁有者可以控制資料輸入和輸出，而且可以追蹤和記錄內容流程。
- 您可以撰寫以承載為基礎的控管規則，以阻止未經授權的承載內容進入訊息串流。
- 您可以將不同的內容存取許可授與個別訂閱者，並稽核整個內容流程程序。

Amazon SNS 中的 Identity and Access Management

存取 Amazon SNS 需要 AWS 登入資料，可用來驗證您的請求。這些登入資料必須具有存取 AWS 資源的許可，例如 Amazon SNS 主題和訊息。以下章節提供如何使用 [AWS Identity and Access Management \(IAM\)](#) 與 Amazon SNS，透過控制其存取權來協助確保您資源安全的詳細資訊。

AWS Identity and Access Management (IAM) 是一種 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制哪些人員可進行身分驗證 (登入) 並獲得授權 (具有許可) 以使用 Amazon SNS 資源。IAM 是您可以免費使用 AWS 服務的。

目標對象

AWS Identity and Access Management (IAM) 的使用方式會有所不同，取決於您在 Amazon SNS 中執行的工作。

服務使用者 – 如果您使用 Amazon SNS 執行任務，您的管理員會為您提供您需要的憑證和許可。隨著您為了執行作業而使用的 Amazon SNS 功能數量變多，您可能會需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon SNS 中的功能，請參閱 [對 Amazon Simple Notification Service 身分和存取進行故障診斷](#)。

服務管理員：若您在公司負責管理 Amazon SNS 資源，您應該擁有 Amazon SNS 的完整存取權。您的任務是判斷服務使用者應存取的 Amazon SNS 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司可搭配 Amazon SNS 使用 IAM 的方式，請參閱 [Amazon SNS 如何與 IAM 搭配使用](#)。

IAM 管理員：如果您是 IAM 管理員，建議您掌握如何撰寫政策以管理 Amazon SNS 存取權的詳細資訊。若要檢視您可以在 IAM 中使用的基於 Amazon SNS 身分的政策範例，請參閱 [Amazon Simple Notification Service 身分型政策範例](#)。

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證 (登入 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分身分身分身分登入。AWS IAM Identity Center (IAM Identity Center) 使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料，都是聯合身分的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使用聯合 AWS 身分存取時，您會間接擔任角色。

根據您身分的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱 AWS 登入 《使用者指南》中的[如何登入您的 AWS 帳戶](#)。

如果您以 AWS 程式設計方式存取，AWS 會提供軟體開發套件 (SDK) 和命令列界面 (CLI)，以使用您的憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱《IAM 使用者指南》中的[適用於 API 請求的 AWS Signature 第 4 版](#)。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如，AWS 建議您使用多重驗證 (MFA) 來提高帳戶的安全性。如需更多資訊，請參閱《AWS IAM Identity Center 使用者指南》中的[多重要素驗證](#)和《IAM 使用者指南》中的[IAM 中的 AWS 多重要素驗證](#)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取帳戶中的所有 AWS 服務和資源。此身分稱為 AWS 帳戶 Theroot 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的[需要根使用者憑證的任務](#)。

聯合身分

最佳實務是，要求人類使用者，包括需要管理員存取權的使用者，使用臨時 AWS 服務憑證與身分提供者聯合來存取。

聯合身分是來自您的企業使用者目錄、Web 身分提供者、AWS Directory Service、身分中心目錄或任何使用透過身分來源提供的憑證 AWS 服務存取的使用者。當聯合身分存取時 AWS 帳戶，它們會擔任角色，而角色會提供臨時登入資料。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連接並同步到您自己的身分來源中的一組使用者 AWS 帳戶和群組，以便在所有和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的[什麼是 IAM Identity Center ?](#)。

IAM 使用者和群組

[IAM 使用者](#)是中的身分 AWS 帳戶，具有單一人員或應用程式的特定許可。建議您盡可能依賴臨時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 [IAM 使用者指南](#)中的為需要長期憑證的使用案例定期輪換存取金鑰。

[IAM 群組](#)是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供臨時憑證。如需更多資訊，請參閱《IAM 使用者指南》中的 [IAM 使用者的使用案例](#)。

IAM 角色

[IAM 角色](#)是具有特定許可 AWS 帳戶的身分。它類似 IAM 使用者，但不與特定的人員相關聯。若要暫時在中擔任 IAM 角色 AWS Management Console，您可以從[使用者切換至 IAM 角色（主控台）](#)。您可以透過呼叫 AWS CLI 或 AWS API 操作或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

使用臨時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱《[IAM 使用者指南](#)》中的為第三方身分提供者 (聯合) 建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的[許可集](#)。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。不過，對於某些 AWS 服務，您可以直接將政策連接到資源 (而不是使用角色做為代理)。如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的[IAM 中的跨帳戶資源存取](#)。
- 跨服務存取 – 有些 AWS 服務使用其他中的功能 AWS 服務。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
- 轉送存取工作階段 (FAS) – 當您使用 IAM 使用者或角色在其中執行動作時 AWS，您被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫的委託人許可 AWS 服務，結合 AWS 服務請求向下游服務提出請求。只有在服務收到需要與其他 AWS 服務或資源互動才能完成的請求時，才會提出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱[《轉發存取工作階段》](#)。

- 服務角色 – 服務角色是服務擔任的 [IAM 角色](#)，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可權給 AWS 服務](#)。
- 服務連結角色 – 服務連結角色是一種連結至的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 – 您可以使用 IAM 角色來管理在 EC2 執行個體上執行之應用程式的臨時登入資料，以及提出 AWS CLI 或 AWS API 請求。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體描述檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得臨時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM 角色來授予許可權給 Amazon EC2 執行個體上執行的應用程式](#)。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策是 中的物件，當與身分或資源建立關聯時，AWS 會定義其許可。當委託人（使用者、根使用者或角色工作階段）發出請求時，會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策會以 JSON 文件 AWS 的形式存放在 中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的[JSON 政策概觀](#)。

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 iam:GetRole 動作的政策。具有該政策的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。如需了解如何在受管政策及內嵌政策之間選擇，請參閱《IAM 使用者指南》中的[在受管政策和內嵌政策間選擇](#)。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中[指定主體](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策中使用來自 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的[存取控制清單 \(ACL\) 概觀](#)。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 Principal 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的[IAM 實體許可界限](#)。
- 服務控制政策 (SCPs) – SCPs 是 JSON 政策，可指定中組織或組織單位 (OU) 的最大許可 AWS Organizations。AWS Organizations 是一種服務，用於分組和集中管理您企業擁有 AWS 帳戶的多個。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的[服務控制政策](#)。
- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許

可，並可能影響身分的有效許可，包括 AWS 帳戶 根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs 的詳細資訊，包括 AWS 服務 支援 RCPs 清單，請參閱 AWS Organizations 《使用者指南》中的[資源控制政策 \(RCPs\)](#)。

- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的[工作階段政策](#)。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的[政策評估邏輯](#)。

存取控制

Amazon SNS 有自己的資源型許可系統，使用與 AWS Identity and Access Management (IAM) 政策相同的語言撰寫的政策。這表示您可以實現類似 Amazon SNS 政策和 IAM 政策的用途。

Note

請務必了解，所有 AWS 帳戶 都可以將其許可委派給其帳戶下的使用者。跨帳戶存取權可讓您共用 AWS 資源的存取權，而無需管理其他使用者。如需使用跨帳戶存取的相關資訊，請參閱 IAM 使用者指南中的[啟用跨帳戶存取權](#)。

Amazon SNS 存取控制使用案例

對於如何授予或拒絕資源的存取，您有很大的彈性。不過，一般使用案例相當簡單：

- 您想要授予另一個 AWS 帳戶 特定類型的主題動作（例如 Publish）。如需詳細資訊，請參閱[授予對主題的 AWS 帳戶 存取權](#)。
- 您想要限制您的主題訂閱為僅限 HTTPS 協定。如需詳細資訊，請參閱[限制訂閱到 HTTPS](#)。
- 您想要允許 Amazon SNS 發佈訊息至您的 Amazon SQS 佇列。如需詳細資訊，請參閱[發佈訊息至 Amazon SQS 佇列](#)。

重要 Amazon SNS 存取政策概念

以下章節說明使用存取原則語言所需要了解的概念。它們是以邏輯順序列出，清單的前面項目是您首先需要知道的項目。

權限

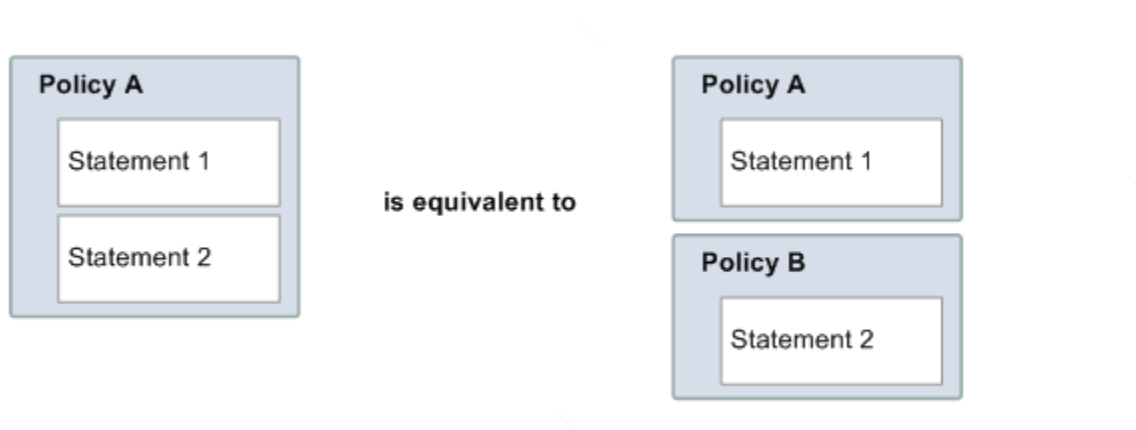
許可的概念是有關允許或不允許特定資源的一些存取類型。許可尤其遵照此形式：「A 有/無許可對適用 D 的 C 執行 B」。例如，Jane (A) 擁有發佈 (B) 到 TopicA (C) 的許可，只要她使用 HTTP 協定 (D)。每當 Jane 發佈到 TopicA 時，服務會檢查她是否擁有許可，以及要求是否符合許可中所述的條件。

陳述式

陳述式是以存取原則語言撰寫之正式的單一許可的描述。您一律要撰寫陳述式做為較廣容器文件 (已知為政策) 的一部分 (請參見下一個概念)。

政策

政策是做為一個或多個陳述式之容器的文件 (以存取原則語言撰寫)。例如，政策可以包含兩個陳述式：一個陳述 Jane 可以使用電子郵件通訊協定來訂閱，另一個陳述 Bob 不可發佈至主題 A。如下圖所示，相等的情況是有兩個政策，一個陳述 Jane 可以使用電子郵件通訊協定，另一個陳述 Bob 不能發佈到主題 A。



政策文件中只允許 ASCII 字元。您可以使用 `aws:SourceAccount` 和 `aws:SourceOwner` 來解決您需要插入其他 AWS 服務包含非 ASCII 字元 ARNs 的情況。請參閱[aws:SourceAccount 與 aws:SourceOwner 的比較](#)之間的差異。

發行者

發行者是撰寫政策以授予資源許可的人員。發行者（依定義）一律是資源擁有者。AWS 不允許 AWS 服務使用者為他們未擁有的資源建立政策。如果 John 是資源擁有者，則當 John 提交他撰寫的政策以授予該資源的許可時，會 AWS 驗證 John 的身分。

Principal

委託人是接收政策中許可的一位或多位人員。委託人是陳述式「A 有許可對適用 D 的 C 執行 B」中的 A。在政策中，您可將委託人設定為「任何人」(例如，您可以指定萬用字元代表所有人員)。您可以如此做，例如，您不想要根據申請者的實際身分限制存取，而是根據其他識別特徵，如申請者的 IP 位址。

動作

動作是委託人擁有執行許可的活動。動作是陳述式「A 有許可對適用 D 的 C 執行 B」中的 B。一般而言，動作只是請求中的操作 AWS。例如，Jane 使用 Action=Subscribe 將請求傳送到 Amazon SNS。您可以在政策中指定一個或多個動作。

資源

資源是委託人請求存取的物件。資源是陳述式「A 有許可對適用 D 的 C 執行 B」中的 C。

條件和金鑰

條件是有關許可的任何限制或詳細資訊。條件是陳述式「A 有許可對適用 D 的 C 執行 B」中的 D。指定條件的政策部分可能是所有部分中最詳細和複雜的。一般條件與下列相關：

- 日期和時間 (例如，請求必須在特定日期之前抵達)
- IP 位址 (例如，申請者的 IP 位址必須是特定 CIDR 範圍的一部分)

金鑰是特定特性，即存取限制的基礎。例如，請求的日期和時間。

您同時使用條件和金鑰來表達限制。了解您實際如何實作限制的最簡單方式是使用範例：如果您想要限制存取期限為 2010 年 5 月 30 日之前，您使用稱為 DateLessThan 的條件。您可以使用名為 aws:CurrentTime 並將其值設定為 2010-05-30T00:00:00Z 的金鑰。AWS 定義您可以使用的條件和金鑰。AWS 服務本身（例如 Amazon SQS 或 Amazon SNS）也可能定義服務特定的金鑰。如需詳細資訊，請參閱[Amazon SNS API 許可：動作和資源參考](#)。

要求者

請求者是將請求傳送至 AWS 服務並要求存取特定資源的人員。請求者傳送請求到 AWS，基本上是說：「您允許我執行 B 到 C，其中 D 適用？」

評估

評估是 AWS 服務用來根據適用政策判斷是否應拒絕或允許傳入請求的程序。如需評估邏輯的資訊，請參閱 [評估邏輯](#)。

Effect

效用是您想要政策陳述式在評估時間傳回的結果。您在政策中撰寫陳述式時指定此值，並且可能的值是拒絕和允許。

例如，您可以撰寫一個政策，含有拒絕所有來自南極之請求的陳述式 (效用=拒絕使用配置至南極之 IP 位址的請求)。或者，您可以撰寫一個政策，含有允許所有不是來自南極之請求的陳述式 (效用=允許不是來自南極的請求)。雖然兩個陳述式看似執行一樣的事情，在存取原則語言邏輯中，它們是不同的。如需詳細資訊，請參閱[評估邏輯](#)。

雖然您可為效用 (允許或拒絕) 指定的只有兩個可能值，卻可以在政策評估時間上有三種不同結果：預設拒絕、允許或明確拒絕。如需詳細資訊，請參閱下列概念和 [評估邏輯](#)。

預設拒絕

預設拒絕是沒有允許或明確拒絕之政策的預設結果。

允許

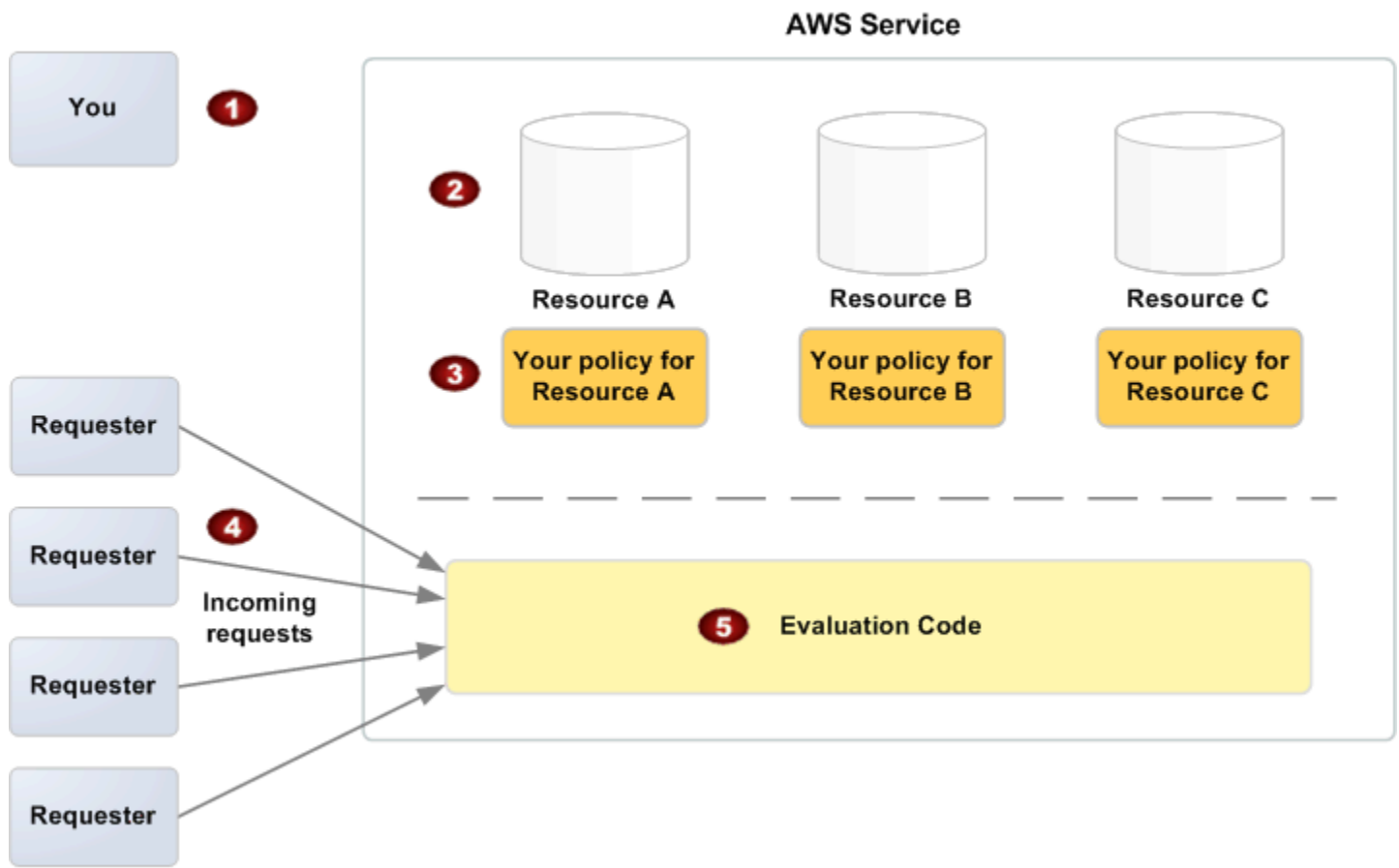
允許源自於具有效用=允許的陳述式，假設所有陳述的條件都符合。範例：如果請求在 2010 年 4 月 30 日下午 1:00 前收到即允許。允許覆寫所有預設拒絕，但是明確拒絕則無法覆寫。

明確拒絕

明確拒絕源自於具有效用=拒絕的陳述式，假設所有陳述的條件都符合。範例：拒絕所有來自南極的請求。任何來自南極的請求一律都予以拒絕，不論任何其他政策是否可能允許。

Amazon SNS 存取控制架構概觀

下圖和表格說明與之互動的主要元件，以提供您的資源的存取控制。



1 您是資源擁有者。

2 您的 資源（包含在 AWS 服務內，例如 Amazon SQS 佇列）。

3 您的政策。

一般每個資源都有一個政策，雖然您可以擁有多個。AWS 服務本身提供用於上傳和管理政策的 API。

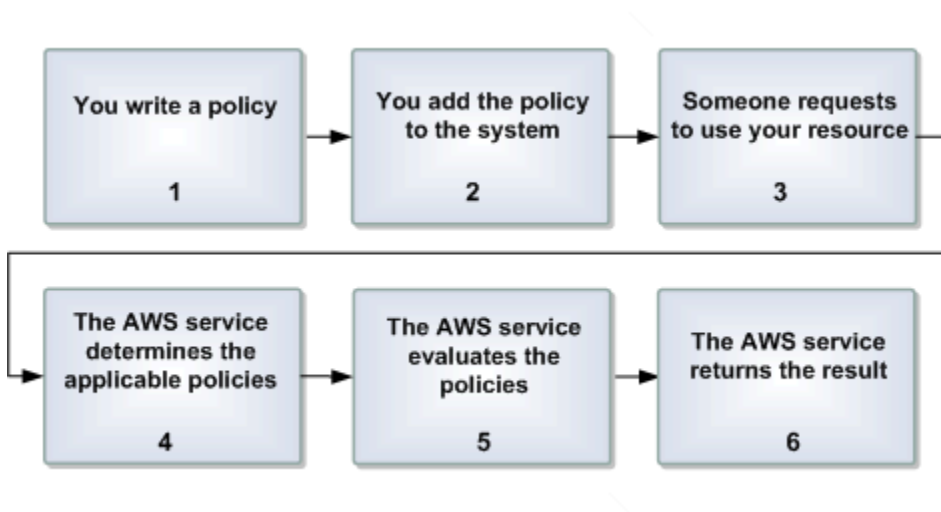
4 請求者及其傳入 AWS 服務請求。

5 存取原則語言評估代碼。

這是 AWS 服務內的一組程式碼，可針對適用的政策評估傳入的請求，並判斷是否允許請求者存取資源。如需服務如何做出決定的資訊，請參閱 [評估邏輯](#)。

在 Amazon SNS 中使用存取政策語言

下圖和表格說明存取控制如何搭配存取原則語言的一般程序。



使用存取控制搭配存取原則語言的程序

1	您為您的資源撰寫政策。 例如，您撰寫政策來為您的 Amazon SNS 主題指定許可。
2	您可以將政策上傳至 AWS。 AWS 服務本身提供用來上傳政策的 API。例如，您使用 Amazon SNS <code>SetTopicAttributes</code> 動作為特定 Amazon SNS 主題上傳政策。
3	某人傳送要求使用您的資源。 例如，使用者傳送使用您的主題之一的請求到 Amazon SNS。
4	AWS 服務會決定哪些政策適用於請求。 例如，Amazon SNS 查看所有可用的 Amazon SNS 政策，然後決定適用的政策 (根據資源內容、誰是申請者等)。
5	AWS 服務會評估政策。 例如，Amazon SNS 評估政策並判定是否允許申請者使用您的主題。如需決定邏輯的資訊，請參閱 評估邏輯。
6	AWS 服務會拒絕請求或繼續處理請求。

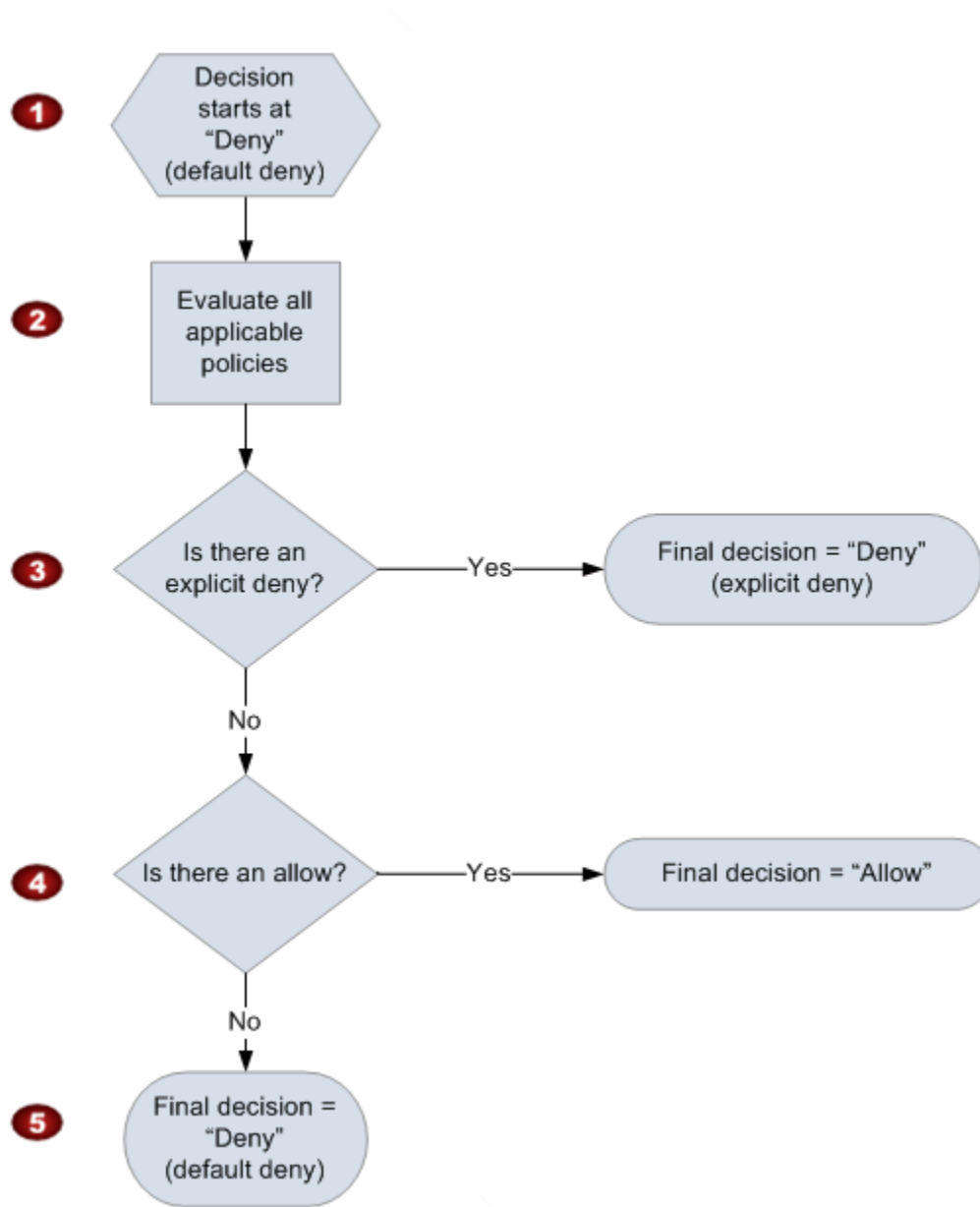
例如，根據政策評估結果，服務傳回「存取遭拒」錯誤給申請者或繼續處理請求。

評估邏輯

評估時間的目標是決定是否應該允許或拒絕授予的請求。評估邏輯遵循幾個基本規則：

- 根據預設，所有使用您的資源的請求來自任何人，但是您被拒絕
- 所有允許覆寫任何預設拒絕
- 明確拒絕覆寫任何允許
- 評估政策的順序並不重要

以下流程圖和討論更詳細說明如何做出決定。



1 決定以預設拒絕開始。

2 然後強制執程式碼會評估所有適用於請求的政策 (根據資源、委託人、動作和條件)。
強制執程式碼評估政策的順序並不重要。

3 在所有那些政策中，強制執程式碼會尋找適用於請求的明確拒絕。

如果找到即使一個，強制執程式碼也會傳回「拒絕」的決定，然後完成程序 (這是明確拒絕；如需詳細資訊，請參閱 [明確拒絕](#))。

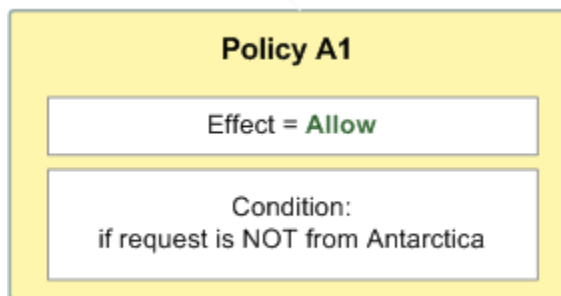
- 4 如果找不到明確拒絕，強制執行程式碼會尋找適用於請求的任何「允許」指示。
如果找到即使一個，強制執行程式碼也會傳回「允許」的決定，然後完成程序 (服務繼續處理請求)。
- 5 如果找不到允許，則最後決定是「拒絕」(因為沒有明確拒絕或允許，這會被視為預設拒絕 (如需詳細資訊，請參閱 [預設拒絕](#)))。

明確拒絕和預設拒絕的相互作用

如果政策不直接套用於請求，其會導致預設拒絕。例如，如果使用者請求使用 Amazon SNS，但主題上的政策 AWS 帳戶 完全不參考使用者的，則該政策會導致預設拒絕。

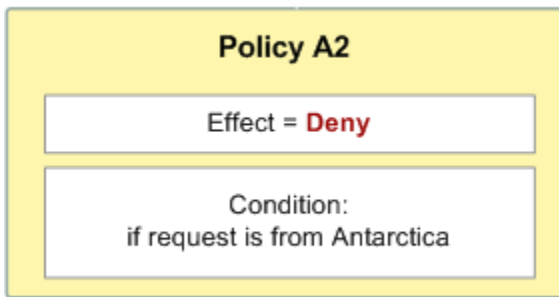
如果未符合陳述式中的條件，政策也會導致預設拒絕。如果符合陳述式中的所有條件，則政策會根據政策中的效用元素的值導致允許或明確拒絕。如果未符合條件，政策不會指定做什麼，所以該狀況下的預設結果是預設拒絕。

例如，假設您想要防止來自南極的請求。您撰寫一個僅允許非來自南極之請求的政策 (稱為政策 A1)。下圖說明該政策。



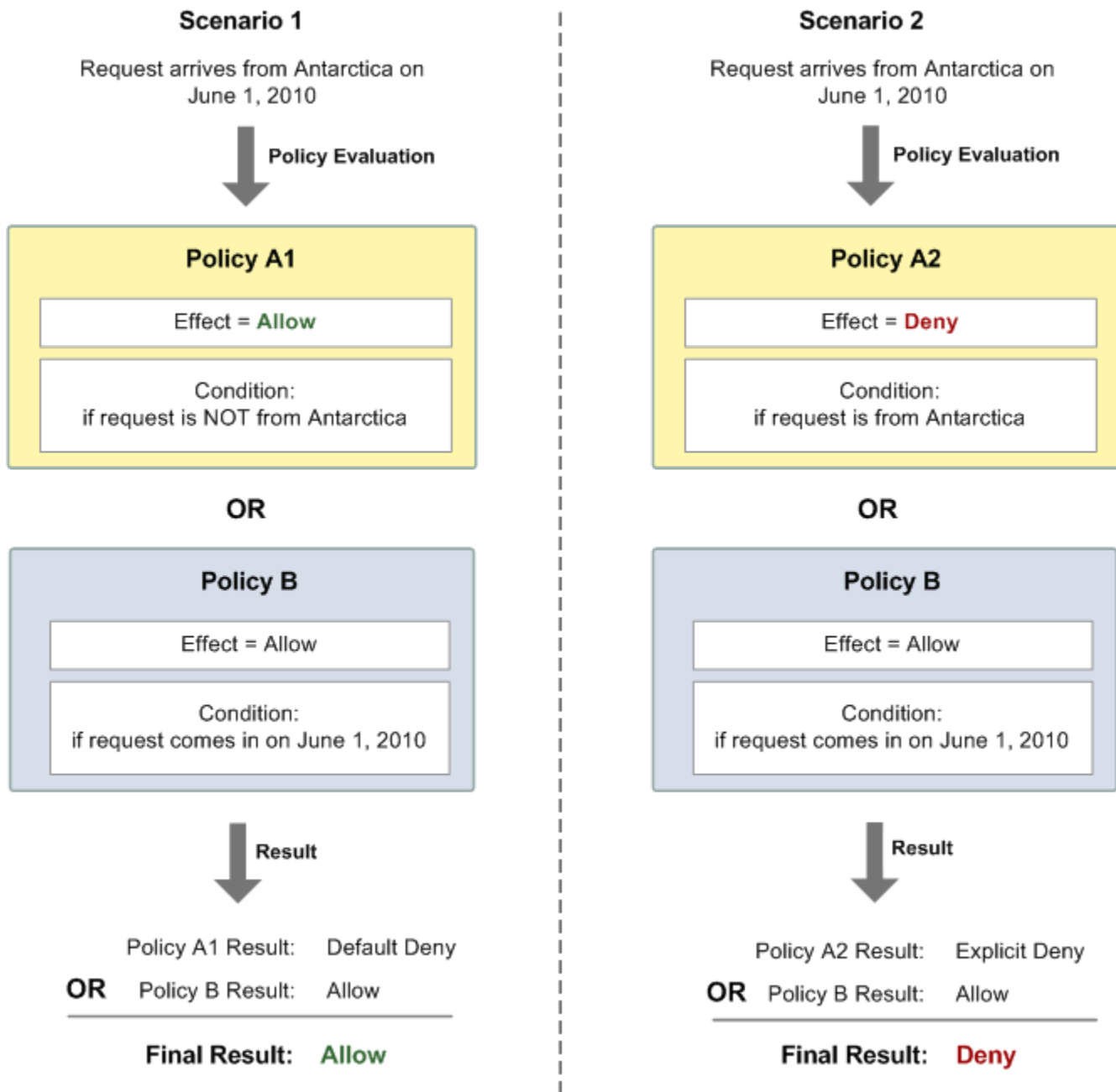
如果某人從美國傳送請求，便符合條件 (請求不是來自南極)。因此，允許該請求。但是，若某人從南極傳送請求，便不符合條件，於是請求的結果是預設拒絕。

您可以透過重寫政策 (名為政策 A2)(如下圖)，將結果轉變為明確拒絕。這時，若請求來自南極，政策明確拒絕該請求。



若某人從南極傳送請求，便符合條件，於是請求的結果是明確拒絕。

預設拒絕和明確拒絕的區別非常重要，因為允許可以覆寫前者，但不能覆寫後者。例如，假設有另一個政策允許請求在 2010 年 6 月 1 日到達。當此政策結合限制從南極來的存取的政策時，此政策會如何影響整體結果？我們來比較結合以日期為基礎的政策 (我將其稱為政策 B) 和前述政策 A1 和 A2 時的整體結果。情況 1 結合政策 A1 和政策 B，情況 2 結合政策 A2 和政策 B。下圖和討論顯示當請求在 2010 年 6 月 1 日來自南極時的結果。



在情況 1 中，政策 A1 如本節之前所述傳回預設拒絕。政策 B 因為政策 (按照定義) 允許 2010 年 6 月 1 日進來的請求而傳回允許。政策 B 的允許會覆寫政策 A1 的預設拒絕，請求因此而被允許。

在情況 2 中，政策 A2 如本節之前所述傳回明確拒絕。一樣，政策 B 傳回允許。政策 A2 的明確拒絕會覆寫政策 B 的允許，請求因此而被拒絕。

Amazon SNS 存取控制的範例案例

本節提供幾個存取控制的一般使用案例的範例。

授予對主題的 AWS 帳戶 存取權

假設您在 Amazon SNS 中有一個主題，而且您想要允許一或多個 對該主題 AWS 帳戶 執行特定動作，例如發佈訊息。您可以使用 Amazon SNS API 動作 `AddPermission` 來完成此操作。

`AddPermission` 動作可讓您指定主題、AWS 帳戶 IDs 清單、動作清單和標籤。然後，Amazon SNS 會自動產生新的政策陳述式，並將其新增至主題的存取控制政策。您不需要自行撰寫政策陳述式，Amazon SNS 會為您處理此問題。如果您稍後需要移除政策，您可以呼叫 `RemovePermission` 並提供您在新增許可時使用的標籤。

例如，如果您在 `arn:aws:sns:us-east-2:444455556666:MyTopic` 主題上呼叫 `AddPermission`，請指定 AWS 帳戶 ID `1111-2222-3333`、`Publish` 動作和標籤 `grant-1234-publish`，Amazon SNS 將產生下列政策陳述式，並將其插入主題的存取控制政策：

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

新增此陳述式後，AWS 帳戶 `1111-2222-3333` 將具有將訊息發佈至主題的許可。

其他資訊：

- **自訂政策管理：**雖然授予許可很 `AddPermission` 方便，但針對更複雜的案例手動管理主題的存取控制政策通常很有用，例如新增條件或將許可授予特定 IAM 角色或服務。您可以使用 `SetTopicAttributes` API 直接更新政策屬性來執行此操作。
- **安全最佳實務：**授予許可時請小心，以確保只有信任的 AWS 帳戶 或 實體可以存取您的 Amazon SNS 主題。定期檢閱和稽核連接到主題的政策，以維護安全性。
- **政策限制：**請記住，Amazon SNS 政策的大小和複雜性有限制。如果您需要新增許多許可或複雜條件，請確保您的政策保持在這些限制內。

限制訂閱到 HTTPS

若要將 Amazon SNS 主題的通知交付通訊協定限制為 HTTPS，您必須建立自訂政策。Amazon SNS 中的 `AddPermission` 動作不允許您在授予主題存取權時指定通訊協定限制。因此，您需要手動撰寫強制執行此限制的政策，然後使用 `SetTopicAttributes` 動作將政策套用至您的主題。

以下是您可以建立限制訂閱 HTTPS 的政策的方式：

1. **撰寫 政策。**政策必須指定您要授予存取權的 AWS 帳戶 ID，並強制執行僅允許 HTTPS 訂閱的條件。以下是範例政策，授予 AWS 帳戶 ID 1111-2222-3333 訂閱主題的許可，但前提是使用的通訊協定是 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. **套用 政策。**使用 Amazon SNS API 中的 `SetTopicAttributes` 動作，將此政策套用至您的主題。將主題的 `Policy` 屬性設定為您建立的 JSON 政策。

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
    .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
    .attributeName("Policy")
    .attributeValue(jsonPolicyString) // The JSON policy as a string
    .build());
```

其他資訊：

- 自訂存取控制。此方法可讓您強制執行更精細的存取控制，例如限制訂閱通訊協定，這無法單獨透過 `AddPermission` 動作執行。自訂政策為需要特定條件的案例提供彈性，例如通訊協定強制執行或 IP 地址限制。
- 安全最佳實務。限制 HTTPS 的訂閱可確保傳輸中的資料已加密，以增強通知的安全性。定期檢閱您的主題政策，以確保其符合您的安全與合規要求。
- 政策測試。在生產環境中套用政策之前，請在開發環境中進行測試，以確保其如預期般運作。這有助於防止意外存取問題或意外限制。

發佈訊息至 Amazon SQS 佇列

若要從 Amazon SNS 主題發佈訊息至 Amazon SQS 佇列，您需要在 Amazon SQS 佇列上設定正確的許可。雖然 Amazon SNS 和 Amazon SQS 都使用 AWS 存取控制政策語言，但您必須在 Amazon SQS 佇列上明確設定政策，以允許從 Amazon SNS 主題傳送訊息。

您可以使用 `SetQueueAttributes` 動作將自訂政策套用至 Amazon SQS 佇列，來達成此目標。與 Amazon SNS 不同，Amazon SQS 不支援使用條件建立政策陳述式 `AddPermission` 的動作。因此，您必須手動撰寫政策。

以下是 Amazon SQS 政策的範例，該政策授予 Amazon SNS 將訊息傳送至佇列的許可。請注意，此政策與 Amazon SQS 佇列相關聯，而不是與 Amazon SNS 主題相關聯。指定的動作是 Amazon SQS 動作，而資源是佇列的 Amazon Resource Name (ARN)。您可以使用 `GetQueueAttributes` 動作來擷取佇列的 ARN。

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


此政策使用 `aws:SourceArn` 條件，根據所傳送訊息的來源限制對 SQS 佇列的存取。這可確保只允許來自指定 SNS 主題（在此情況下，`arn:aws:sns:us-east-2:444455556666:MyTopic`）的訊息交付至佇列。

其他資訊：

- 佇列 ARN。請確定您使用 `GetQueueAttributes` 動作擷取 Amazon SQS 佇列的正確 ARN。此 ARN 對於設定正確的許可至關重要。
- 安全最佳實務。設定政策時，請務必遵循最低權限原則。僅授予 Amazon SNS 主題與 Amazon SQS 佇列互動的必要許可，並定期檢閱您的政策，以確保其 up-to-date 且安全。
- Amazon SNS 中的預設政策。Amazon SNS 不會自動授予預設政策，允許其他 AWS 服務或帳戶存取新建立的主題。根據預設，Amazon SNS 主題是沒有許可的建立，這表示這些主題是私有的，而且只有建立它們的帳戶才能存取。若要啟用其他 AWS 服務、帳戶或主體的存取權，您必須明確定義存取政策並將其連接至主題。這符合最低權限原則，確保預設不會授予意外存取。
- 測試和驗證。設定政策後，透過將訊息發佈至 Amazon SNS 主題並驗證它們是否已成功交付至 Amazon SQS 佇列來測試整合。這有助於確認政策已正確設定。

允許 Amazon S3 事件通知發佈至主題

若要允許來自另一個的 Amazon S3 儲存貯 AWS 帳戶體將事件通知發佈到您的 Amazon SNS 主題，您需要相應地設定主題的存取政策。這包括撰寫自訂政策，以從特定授予 Amazon S3 服務的許可，AWS 帳戶然後將此政策套用至您的 Amazon SNS 主題。

以下是您可以設定的方式：

1. 撰寫政策。政策應授予 Amazon S3 服務 (`s3.amazonaws.com`) 發佈至 Amazon SNS 主題的必要許可。您將使用 `SourceAccount` 條件，以確保只有 AWS 帳戶擁有 Amazon S3 儲存貯體的指定才能發佈通知到您的主題。

以下為政策的範例：

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

- 主題擁有者 – 111122223333 是擁有 Amazon SNS 主題的 AWS 帳戶 ID。
 - Amazon S3 儲存貯體擁有者 – 444455556666 是擁有 Amazon S3 儲存貯體傳送通知的 AWS 帳戶 ID。
2. 套用 政策。使用 `SetTopicAttributes` 動作在 Amazon SNS 主題上設定此政策。這將更新主題的存取控制，以包含自訂政策中指定的許可。

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
    .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
    .attributeName("Policy")
    .attributeValue(jsonPolicyString) // The JSON policy as a string
    .build());
```

其他資訊：

- 使用 **SourceAccount** 條件。SourceAccount 條件可確保在此情況下，只有來自指定 AWS 帳戶 (444455556666 的事件) 可以觸發 Amazon SNS 主題。這是安全措施，可防止未經授權的帳戶傳送通知到您的主題。
- 其他支援的服務**SourceAccount**。下列 服務支援 SourceAccount 條件。當您想要根據原始帳戶限制對 Amazon SNS 主題的存取時，請務必使用此條件。
 - Amazon API Gateway
 - Amazon CloudWatch
 - Amazon DevOps Guru
 - Amazon EventBridge
 - Amazon GameLift 伺服器
 - Amazon Pinpoint SMS and Voice API
 - Amazon RDS
 - Amazon Redshift

- Amazon S3 Glacier
 - Amazon SES
 - Amazon Simple Storage Service
 - AWS CodeCommit
 - AWS Directory Service
 - AWS Lambda
 - AWS Systems Manager Incident Manager
- 測試和驗證。套用政策後，透過觸發 Amazon S3 儲存貯體中的事件並確認成功發佈至 Amazon SNS 主題，來測試設定。這有助於確保您的政策設定正確。
 - 安全最佳實務。定期檢閱和稽核您的 Amazon SNS 主題政策，以確保它們符合您的安全需求。限制只能存取信任的帳戶和服務對於維護安全操作至關重要。

允許 Amazon SES 發佈到其他帳戶擁有的主題

您可以允許另一個 AWS 服務 發佈到另一個 擁有的主題 AWS 帳戶。假設您登錄到 111122223333 帳戶，打開了 Amazon SES，並建立了一個電子郵件。若要將此電子郵件的通知發佈到 444455556666 帳戶擁有的 Amazon SNS 主題，您必須建立如下所示的政策。若要這麼做，您需要提供有關主體 (其他服務) 和每個資源擁有權的資訊。Resource 陳述式提供主題 ARN，其中包括主題擁有者的帳戶 ID 444455556666。"aws:SourceOwner": "111122223333" 陳述式會指定您的帳戶擁有該電子郵件。

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
  ]
}
```

```
}  
]  
}
```

將事件發佈到 Amazon SNS 時，下列服務支援 `aws:SourceOwner`：

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps Guru
- Amazon GameLift 伺服器
- Amazon Pinpoint SMS and Voice API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` 與 `aws:SourceOwner` 的比較

Important

`aws:SourceOwner` 已棄用，新服務只能透過 `aws:SourceArn` 和 `aws:SourceAccount` 與 Amazon SNS 整合。對於目前支援 `aws:SourceOwner` 的現有服務，Amazon SNS 仍維持回溯相容性。

`aws:SourceAccount` 和 `aws:SourceOwner` 條件金鑰是由一些發布至 Amazon SNS 主題的 AWS 服務所設定的。支援時，該值將是服務代表其發佈資料的 12 位數 AWS 帳戶 ID。有些服務支持一個，有些支持另一個。

- 如需 Amazon S3 通知的使用 [允許 Amazon S3 事件通知發佈至主題](#) 方式，`aws:SourceAccount` 以及支援該條件 AWS 的服務清單，請參閱。

- 如需 [允許 Amazon SES 發佈到其他帳戶擁有的主題](#) Amazon SES 如何使用 `aws:SourceOwner` 和支援該條件 AWS 的服務清單，請參閱。

允許 中組織中的帳戶 AWS Organizations 發佈至不同帳戶中的主題

AWS Organizations 此服務可協助您集中管理帳單、控制存取和安全性，以及跨 共用資源 AWS 帳戶。

您可以在[組織主控台](#)中找到您的組織 ID。如需詳細資訊，請參閱[從管理帳戶檢視組織的詳細資訊](#)。

在此範例中，AWS 帳戶 組織中的任何 `myOrgId` 都可以發佈至帳戶 `MyTopic` 中的 Amazon SNS 主題 `444455556666`。政策會使用 `aws:PrincipalOrgID` 全域條件金鑰來檢查組織 ID 值。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "myOrgId"
        }
      }
    }
  ]
}
```

允許任何 CloudWatch 警示發佈到不同帳戶中的主題

使用下列步驟，透過不同的 CloudWatch 警示叫用 Amazon SNS 主題 AWS 帳戶。此範例使用兩個帳戶：

- 帳戶 A 用於建立 CloudWatch 警示。
- 帳戶 B 用於建立 SNS 主題。

在帳戶 B 中建立 SNS 主題

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中選擇 Topics (主題)，然後選擇 Create topic (建立主題)。
3. 選擇主題類型的標準，然後為主題建立名稱。
4. 選擇建立主題，然後複製主題的 ARN。
5. 在導覽窗格中選擇 Subscriptions (訂閱)，然後選擇 Create subscription (建立訂閱)。
6. 在主題 ARN 區段中新增主題的 ARN，選擇電子郵件做為通訊協定，然後輸入電子郵件地址。
7. 選擇建立訂閱，然後檢查您的電子郵件以確認訂閱。

在帳戶 A 中建立 CloudWatch 警示

1. 透過 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 在導覽窗格中，選擇警示，然後選擇建立警示。
3. 如果您尚未建立警示，請立即建立一個警示。否則，請選取您的指標，然後提供閾值和比較參數的詳細資訊。
4. 在設定動作的通知下，選擇使用主題 ARN 通知其他帳戶，然後從帳戶 B 輸入主題 ARN。
5. 建立警示的名稱，然後選擇建立警示。

更新帳戶 B 中 SNS 主題的存取政策

1. 登入 [Amazon SNS 主控台](#)。
2. 在導覽窗格中，選擇主題，然後選取主題。
3. 選擇編輯，然後將以下內容新增至政策：

Note

將以下政策中的範例值取代為您自己的值。

```
{
  "Version": "2008-10-17",
  "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": "example-topic-arn-account-b",
"Condition": {
  "ArnLike": {
    "aws:SourceArn": "arn:aws:cloudwatch:example-region:111122223333:alarm:"
  }
}
}
```

測試警示

若要測試警示，請根據指標資料點變更警示閾值，或手動變更警示狀態。當您變更警示閾值或警示狀態時，您會收到電子郵件通知。

使用本機 Amazon SNS 主題和轉送訊息的解決方法

使用下列步驟為 CloudWatch 警示啟用跨帳戶 Amazon SNS 通知：

1. 在與 CloudWatch 警示 (111122223333) 相同的帳戶中建立 [Amazon SNS 主題](#)。
2. 訂閱 [Lambda 函數](#) 或 [Amazon EventBridge 規則](#) 至該 Amazon SNS 主題。
3. 然後，Lambda 函數或 EventBridge 規則可以將訊息發佈到目標帳戶 (444455556666) 中的 Amazon SNS 主題。

限制僅從特定 VPC 端點發佈至 Amazon SNS 主題

在這種情況下，帳戶 444455556666 中的主題僅允許從具有 ID vpce-1ab2c34d 的 VPC 端點發佈。

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

Amazon SNS 如何與 IAM 搭配使用

在您使用 IAM 管理對 Amazon SNS 的存取權之前，請瞭解哪些 IAM 功能可以與 Amazon SNS 搭配使用。

您可以搭配 Amazon Simple Notification Service 使用的 IAM 功能

IAM 功能	支援 Amazon SNS
身分型政策	是
資源型政策	是
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
主體許可	是

IAM 功能	支援 Amazon SNS
服務角色	是
服務連結角色	否

若要全面了解 Amazon SNS 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《[AWS IAM 使用者指南](#)》中的與 IAM 搭配使用的服務。

AWS 的 受管政策 Amazon Simple Notification Service

AWS 受管政策是由 AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您可以開始將許可指派給使用者、群組和角色。

請記住，AWS 受管政策可能不會授予特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的[客戶管理政策](#)，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受管政策中定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。AWS 服務當新的啟動或新的 API 操作可用於現有服務時，AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》中的[AWS 受管政策](#)。

AWS 受管政策：AmazonSNSFullAccess

AmazonSNSFullAccess 使用提供 Amazon SNS 的完整存取權 AWS Management Console。此政策也包含下列在使用 Amazon SNS 呼叫 AWS End User Messaging SMS 時的讀取和寫入動作。您可以將此政策連接至您的使用者、群組或角色。

許可詳細資訊

下列許可僅適用於使用 Amazon SNS APIs 時：

- `sns:*` – 允許執行與 Amazon SNS 相關的任何動作的完整許可。此萬用字元 (*) 表示使用者可以執行所有可能的 Amazon SNS 動作。
- `sms-voice:DescribeVerifiedDestinationNumbers` – 可讓您擷取已驗證在內傳送簡訊的電話號碼清單 AWS 帳戶。

- `sms-voice:CreateVerifiedDestinationNumber` – 可讓您驗證新的電話號碼，以搭配簡訊服務使用 AWS。
- `sms-voice:SendDestinationNumberVerificationCode` – 可讓您將驗證碼傳送至正在進行簡訊驗證的電話號碼 AWS。
- `sms-voice:SendTextMessage` – 可讓您建立新的文字訊息，並將其傳送至收件人的電話號碼。`SendTextMessage` 只會在每次叫用時傳送簡訊給一個收件人。
- `sms-voice>DeleteVerifiedDestinationNumber` – 可讓您從 中的已驗證號碼清單中移除電話號碼 AWS 帳戶
- `sms-voice:VerifyDestinationNumber` – 可讓您啟動和完成驗證程序，以便用於簡訊服務的電話號碼 AWS。
- `sms-voice:DescribeAccountAttributes` – 可讓您擷取與 內簡訊服務相關的帳戶層級屬性詳細資訊 AWS。
- `sms-voice:DescribeSpendLimits` – 可讓您擷取 內與簡訊服務相關的花費限制資訊 AWS 帳戶
- `sms-voice:DescribePhoneNumbers` – 可讓您擷取 內與簡訊服務相關聯的電話號碼詳細資訊 AWS 帳戶
- `sms-voice:SetTextMessageSpendLimitOverride` – 可讓您設定或覆寫 內 SMS 文字訊息的花費限制 AWS 帳戶
- `sms-voice:DescribeOptedOutNumbers` – 可讓您擷取已選擇不接收來自您 AWS 帳戶簡訊的電話號碼清單。
- `sms-voice>DeleteOptedOutNumber` – 可讓您從 中的選擇退出號碼清單中移除電話號碼 AWS 帳戶

AmazonSNSFullAccess 範例政策

```
{
  "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"
        }
    }
}
]
}

```

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的 [AmazonSNSFullAccess](#)。

AWS 受管政策：AmazonSNSReadOnlyAccess

AmazonSNSReadOnlyAccess 使用 提供對 Amazon SNS 的唯讀存取 AWS Management Console。此政策也包含下列在使用 Amazon SNS 呼叫 AWS End User Messaging SMS 時的唯讀動作。您可以將此政策連接至您的使用者、群組和角色。

許可詳細資訊

下列許可僅適用於使用 Amazon SNS APIs時：

- `sns:GetTopicAttributes` – 可讓您擷取 Amazon SNS 主題的屬性。這包括主題的 ARN (Amazon Resource Name)、訂閱者清單、交付政策、存取控制政策，以及與主題相關聯的任何其他中繼資料等資訊。
- `sns:List*` – 可讓您針對 Amazon SNS 資源執行開頭 `List` 為的任何操作。這包括列出與 Amazon SNS 相關各種元素的許可，例如：

- `sns:ListTopics` – 可讓您擷取 中所有 Amazon SNS 主題的清單 AWS 帳戶。
- `sns:ListSubscriptions` – 可讓您擷取 Amazon SNS 主題的所有訂閱清單。
- `sns:ListSubscriptionsByTopic` – 可讓您列出特定 Amazon SNS 主題的所有訂閱。
- `sns:ListPlatformApplications` – 可讓您列出為行動推播通知建立的所有平台應用程式。
- `sns:ListEndpointsByPlatformApplication` – 可讓您列出與平台應用程式相關聯的所有端點。
- `sns:CheckIfPhoneNumberIsOptedOut` – 可讓您檢查特定電話號碼是否已選擇不透過 Amazon SNS 接收簡訊。
- `sns:GetEndpointAttributes` – 可讓您擷取與 Amazon SNS 平台應用程式相關聯的端點屬性。這可能包括屬性，例如端點的啟用狀態、自訂使用者資料，以及與端點相關聯的任何其他中繼資料。
- `sns:GetDataProtectionPolicy` – 可讓您擷取與 Amazon SNS 主題相關聯的資料保護政策。
- `sns:GetPlatformApplicationAttributes` – 可讓您擷取 Amazon SNS 平台應用程式的屬性。平台應用程式用於 Amazon SNS，透過 Apple 推送通知服務 (APNS) 或 Firebase Cloud Messaging (FCM) 等服務將推送通知傳送至行動裝置。
- `sns:GetSMSAttributes` – 可讓您擷取 的預設簡訊設定 AWS 帳戶。
- `sns:GetSMSSandboxAccountStatus` – 可讓您擷取 的 SMS 沙盒目前狀態 AWS 帳戶。
- `sns:GetSubscriptionAttributes` – 可讓您擷取特定訂閱 Amazon SNS 主題的屬性。
- `sms-voice:DescribeVerifiedDestinationNumbers` – 可讓您檢視或擷取已驗證在 中傳送簡訊的電話號碼清單 AWS 帳戶
- `sms-voice:DescribeAccountAttributes` – 可讓您檢視或擷取與內簡訊服務相關的帳戶層級屬性資訊 AWS。
- `sms-voice:DescribeSpendLimits` – 可讓您檢視或擷取與 內簡訊服務相關的花費限制資訊 AWS 帳戶
- `sms-voice:DescribePhoneNumbers` – 可讓您檢視或擷取 內用於簡訊服務的電話號碼相關資訊 AWS 帳戶
- `sms-voice:DescribeOptedOutNumbers` – 可讓您檢視或擷取已選擇不接收簡訊的電話號碼清單 AWS 帳戶

AmazonSNSReadOnlyAccess 範例政策

```
{
  "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"
        }
      }
    }
  ]
}

```

若要檢視此政策的許可，請參閱《AWS 受管政策參考》中的 [AmazonSNSFullAccess](#)。

AWS 受管政策的 Amazon SNS 更新

檢視自此服務開始追蹤 Amazon SNS AWS 受管政策更新以來的詳細資訊。如需此頁面變更的自動提醒，請訂閱 Amazon SNS 文件歷史記錄頁面上的 RSS 摘要。

變更	描述	日期
AmazonSNSFullAccess – 更新現有政策	Amazon SNS 新增了新的許可，以允許使用完整存取 Amazon SNS AWS Management Console。	09/24/2024
AmazonSNSReadOnlyAccess – 更新現有政策	Amazon SNS 新增了新的許可，以允許使用對 Amazon SNS 進行唯讀存取 AWS Management Console。	09/24/2024
Amazon SNS 開始追蹤變更	Amazon SNS 開始追蹤其 AWS 受管政策的變更。	08/27/2024

Amazon SNS 的政策動作

支援政策動作：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 Action 元素描述您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API 操作相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon SNS 動作的清單，請參閱《服務授權參考》中的 [Amazon Simple Notification Service 定義的資源](#)。

Amazon SNS 中的政策動作會在動作之前使用下列字首：

```
sns
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
```

```
"sns:action1",
"sns:action2"
]
```

若要檢視 Amazon SNS 身分類型政策的範例，請參閱 [Amazon Simple Notification Service 身分型政策範例](#)。

Amazon SNS 的政策資源

支援政策資源：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"

```

若要查看 Amazon SNS 資源類型及其 ARN 的清單，請參閱《服務授權參考》中的 [Amazon Simple Notification Service 定義的動作](#)。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 [Amazon Simple Notification Service 定義的資源](#)。

若要檢視 Amazon SNS 身分類型政策的範例，請參閱 [Amazon Simple Notification Service 身分型政策範例](#)。

Amazon SNS 的政策條件索引鍵

支援服務特定政策條件金鑰：是

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用 [條件運算子](#) 的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵，AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，會使用邏輯 OR 操作 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 [IAM 政策元素：變數和標籤](#)。

AWS 支援全域條件金鑰和服務特定的條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容索引鍵](#)。

若要查看 Amazon SNS 條件索引鍵的清單，請參閱《服務授權參考》中的 [Amazon Simple Notification Service 的條件索引鍵](#)。若要了解您可以搭配哪些動作和資源使用條件金鑰，請參閱 [Amazon Simple Notification Service 定義的資源](#)。

若要檢視 Amazon SNS 身分類型政策的範例，請參閱 [Amazon Simple Notification Service 身分型政策範例](#)。

Amazon SNS 的 ACL

支援 ACL：否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

搭配使用 ABAC 與 Amazon SNS

支援 ABAC (政策中的標籤)：部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在 AWS 中，這些屬性稱為標籤。您可以將標籤連接至 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的 [條件元素](#) 中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的[使用 ABAC 授權定義許可](#)。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的[使用屬性型存取控制 \(ABAC\)](#)。

將暫時憑證與 Amazon SNS 搭配使用

支援臨時憑證：是

當您使用臨時登入資料登入時，有些 AWS 服務 無法運作。如需詳細資訊，包括使用哪些臨時登入 AWS 服務 資料，請參閱《[AWS 服務 IAM 使用者指南](#)》中的使用 IAM 的 。

如果您 AWS Management Console 使用使用者名稱和密碼以外的任何方法登入，則表示您使用的是暫時登入資料。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時登入資料。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱《IAM 使用者指南》中的[從使用者切換至 IAM 角色 \(主控台\)](#)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱[IAM 中的暫時性安全憑證](#)。

Amazon SNS 的跨服務主體許可

支援轉寄存取工作階段 (FAS)：是

當您使用 IAM 使用者或角色在 中執行動作時 AWS，您會被視為委託人。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，結合 AWS 服務 請求向下游服務提出請求。FAS 請求只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時才會提出。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱[轉發存取工作階段](#)。

Amazon SNS 的服務角色

支援服務角色：是

服務角色是服務擔任的 [IAM 角色](#)，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[建立角色以委派許可權給 AWS 服務](#)。

Warning

變更服務角色的許可可能會中斷 Amazon SNS 功能。只有在 Amazon SNS 提供指引時，才能編輯服務角色。

Amazon SNS 的服務連結角色

支援服務連結角色：否

服務連結角色是連結至的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶，並由服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱[可搭配 IAM 運作的AWS 服務](#)。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon Simple Notification Service 身分型政策範例

根據預設，使用者和角色不具備建立或修改 Amazon SNS 資源的許可。他們也無法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 來執行任務。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

如需了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的[建立 IAM 政策 \(主控台\)](#)。

如需 Amazon SNS 所定義之動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱《服務授權參考》中的[適用 Amazon Simple Notification Service 的動作、資源和條件金鑰](#)。

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、存取或刪除 Amazon SNS 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用 AWS 受管政策來授予許多常見使用案例的許可。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需更多資訊，請參閱 IAM 使用者指南中的 [AWS 受管政策](#) 或 [任務職能的AWS 受管政策](#)。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 [IAM 中的政策和許可](#)。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。如果透過特定 等使用服務動作

AWS 服務，您也可以使用條件來授予存取 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 [IAM JSON 政策元素：條件](#)。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html中的透過 MFA 的安全 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 [IAM 安全最佳實務](#)。

使用 Amazon SNS 主控台

若要存取 Amazon Simple Notification Service 主控台，您必須擁有最基本的一組許可。這些許可必須允許您列出和檢視中 Amazon SNS 資源的詳細資訊 AWS 帳戶。如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為了確保使用者和角色仍可使用 Amazon SNS 主控台，也請將 Amazon SNS *ConsoleAccess* 或 *ReadOnly* AWS 受管政策連接至實體。如需詳細資訊，請參閱《IAM 使用者指南》中的[新增許可到使用者](#)。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 Principal 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 [IAM 實體許可界限](#)。
- 服務控制政策 (SCPs) – SCPs 是 JSON 政策，可指定中組織或組織單位 (OU) 的最大許可 AWS Organizations。AWS Organizations 是一種服務，用於分組和集中管理您企業擁有 AWS 帳戶的多個。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 會

限制成員帳戶中實體的許可，包括每個 AWS 帳戶 根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的[服務控制政策](#)。

- 資源控制政策 (RCP) - RCP 是 JSON 政策，可用來設定您帳戶中資源的可用許可上限，採取這種方式就不需要更新附加至您所擁有的每個資源的 IAM 政策。RCP 會限制成員帳戶中資源的許可，並可能影響身分的有效許可，包括 AWS 帳戶 根使用者，無論它們是否屬於您的組織。如需 Organizations 和 RCPs 的詳細資訊，包括 AWS 服務 支援 RCPs 清單，請參閱 AWS Organizations 《使用者指南》中的[資源控制政策 \(RCPs\)](#)。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過撰寫程式的方式建立角色或聯合使用者的暫時工作階段時，做為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的[工作階段政策](#)。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的[政策評估邏輯](#)。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視連接到他們使用者身分的內嵌及受管政策。此政策包含在主控台上完成此動作的許可，或使用 AWS CLI 或 AWS API 以程式設計方式完成此動作的許可。

```
{
  "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": "*"
}
]
```

Amazon SNS 身分型政策

支援身分型政策：是

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的[IAM JSON 政策元素參考](#)。

Amazon SNS 的身分型政策範例

若要檢視 Amazon SNS 身分類型政策的範例，請參閱 [Amazon Simple Notification Service 身分型政策範例](#)。

Amazon SNS 中的資源型政策

支援以資源基礎的政策

是

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源

的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中[指定主體](#)。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，做為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當委託人和資源位於不同位置時 AWS 帳戶，信任帳戶中的 IAM 管理員也必須授予委託人實體（使用者或角色）存取資源的許可。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的快帳戶資源存取](#)。

使用以身分為基礎的政策搭配 Amazon SNS

Amazon Simple Notification Service 與 AWS Identity and Access Management (IAM) 整合，以便您可以指定中的使用者 AWS 帳戶可以使用 Amazon SNS 資源執行哪些 Amazon SNS 動作。您可以在政策中指定特定主題。例如，您可以在建立 IAM 政策 (在您的組織中授予特定使用者許可) 時使用變數，以便在您的 AWS 帳戶帳戶中對特定主題使用 Publish 動作。如需詳細資訊，請參閱[使用 IAM](#) 指南中的政策變數。

Important

使用 Amazon SNS 搭配 IAM 並不會改變您使用 Amazon SNS 的方式。Amazon SNS 動作沒有變更，也無與使用者和存取控制相關的新 Amazon SNS 動作。

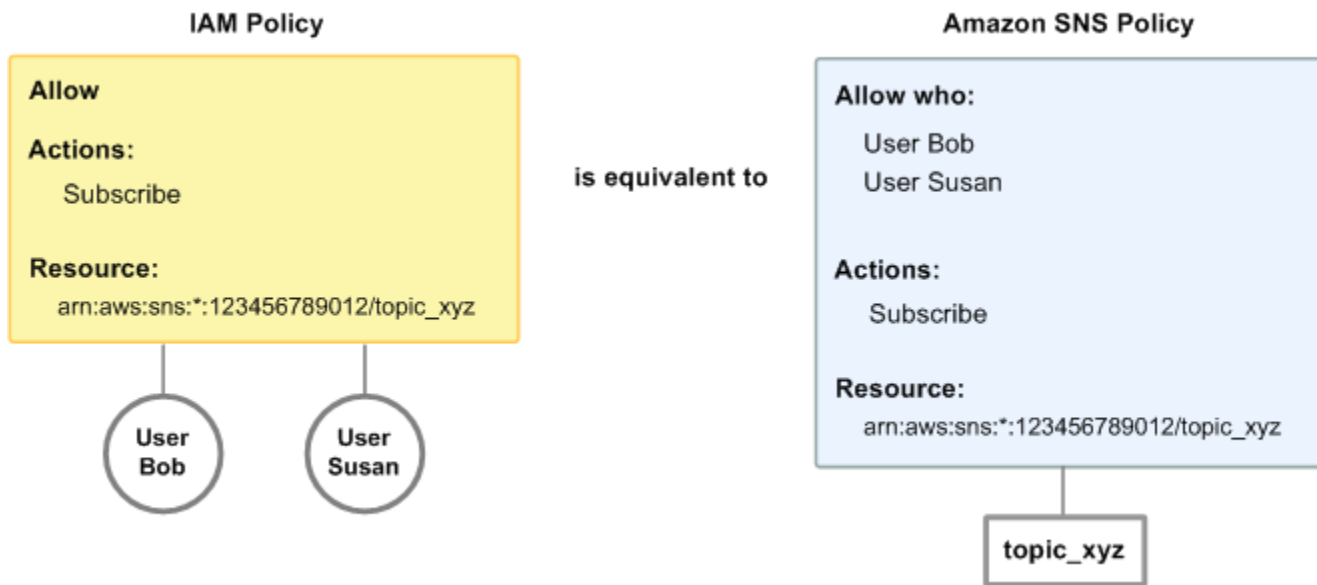
如需涵蓋 Amazon SNS 動作和資源之政策的詳細資訊，請參閱[用於 Amazon SNS 的政策範例](#)。

IAM 和 Amazon SNS 政策一起搭配

您使用 IAM 政策來限制使用者對 Amazon SNS 動作和主題的存取。IAM 政策只能限制您 AWS 帳戶中的使用者存取，不能限制其他的存取 AWS 帳戶。

您對特定主題使用 Amazon SNS 政策來限制誰可使用該主題 (例如誰可發佈訊息到主題，誰可訂閱主題等)。Amazon SNS 政策可以將存取權授予其他 AWS 帳戶，或您自己的使用者 AWS 帳戶。

若要提供您的使用者 Amazon SNS 主題的許可，您可以使用 IAM 政策、Amazon SNS 政策或二者都使用。大部分，您可以使用任一個達成相同結果。例如，下圖顯示 IAM 政策和相當的 Amazon SNS 政策。IAM 政策允許 IAM 政策中名為 topic_xyz AWS 帳戶主題的 Amazon SNS Subscribe 動作連接至使用者 Bob 和 Susan (這表示 Bob 和 Susan 具有政策中所述的許可)。Amazon SNS 同樣授予 Bob 和 Susan 許可，以存取 topic_xyz 的 Subscribe。



Note

前面的範例顯示沒有條件的簡易政策。您可以在任一政策中指定特定條件，並取得相同結果。

IAM AWS 和 Amazon SNS 政策之間有一種差異：Amazon SNS 政策系統可讓您將許可授予其他政策 AWS 帳戶，而 IAM 政策則不會。

您可根據您的需求決定如何一起使用兩個系統來管理您的許可。以下範例顯示兩個政策系統如何搭配運作。

Example 1

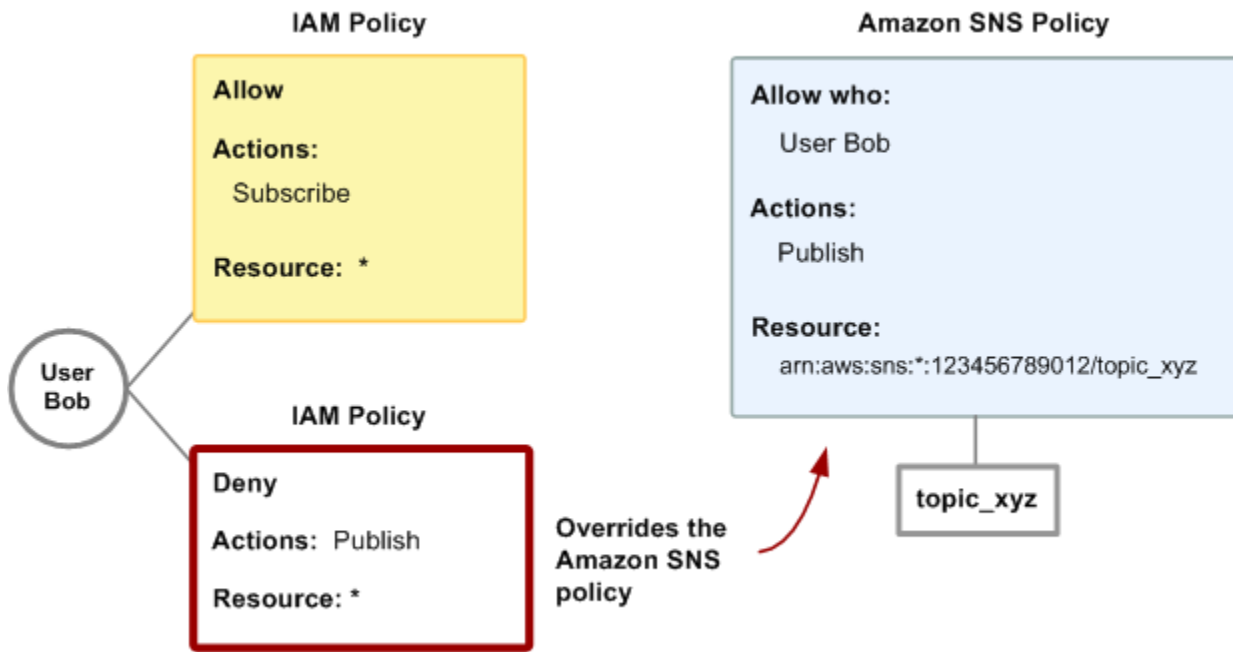
在此範例中，IAM 政策和 Amazon SNS 政策二者均套用到 Bob。IAM 政策授予他任何 AWS 帳戶主題 `Subscribe` 的許可，而 Amazon SNS 政策授予他在特定主題 (`topic_xyz`) `Publish` 上使用的許可。此圖說明了此概念。



如果 Bob 傳送訂閱 AWS 帳戶中任何主題的請求，IAM 政策會允許 動作。如果 Bob 會傳送發佈訊息到 topic_xyz 的請求，Amazon SNS 政策會允許該動作。

Example 2

在此範例中，我們建置在範例 1 (其中 Bob 套用了兩個政策) 上。也就是說，Bob 發佈訊息到他不應該有的 topic_xyz，所以您想要將其發佈到主題的能力整個移除。最簡單的方式是新增一個拒絕他存取所有主題的 Publish 動作的 IAM 政策。第三個政策會覆寫原本讓他發佈到 topic_xyz 的 Amazon SNS 政策的許可，因為明確拒絕會覆寫允許 (如需政策評估的詳細資訊，請參閱 [評估邏輯](#))。此圖說明了此概念。



如需涵蓋 Amazon SNS 動作和資源之政策的詳細資訊，請參閱 [用於 Amazon SNS 的政策範例](#)。

Amazon SNS 資源 ARN 格式

對於 Amazon SNS，主題是您唯一可在政策中指定的資源類型。以下是主題的 Amazon 資源名稱 (ARN) 格式。

```
arn:aws:sns:region:account_ID:topic_name
```

如需關於 ARN 的詳細資訊，請前往 IAM 使用者指南中的 [ARN](#)。

Example

以下是 us-east-2 區域中名為 MyTopic 之主題的 ARN，屬於 AWS 帳戶 123456789012。

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

Example

如果您在 Amazon SNS 支援的各個不同區域擁有名為 MyTopic 的主題，您可使用下列 ARN 指定主題。

```
arn:aws:sns:*:123456789012:MyTopic
```

您可以在主題名稱中使用 * 和 ? 萬用字元。例如，以下可參照 Bob 建立的所有主題，其中首碼為 bob_。

```
arn:aws:sns:*:123456789012:bob_*
```

為方便您使用，當您建立主題時，Amazon SNS 會在回應中傳回主題的 ARN。

Amazon SNS API 動作

在 IAM 政策中，您可以指定任何 Amazon SNS 所提供的動作。不過，ConfirmSubscription 和 Unsubscribe 動作不需要身分驗證，表示即使您在政策中指定那些動作，IAM 也不會限制使用者對那些動作的存取。

您在政策中指定的各個動作必須以小寫字串 sns: 做為首碼。例如若要指定所有 Amazon SNS 動作，您可使用 sns:*。如需各項動作的清單，請前往 [Amazon Simple Notification Service API 參考](#)。

Amazon SNS 政策金鑰

Amazon SNS 會實作下列 AWS 廣泛的政策金鑰，以及一些服務特定的金鑰。

如需每個支援的條件金鑰清單 AWS 服務，請參閱《IAM 使用者指南》中的 [的動作、資源和條件金鑰 AWS 服務](#)。如需可在多個中使用的條件索引鍵清單 AWS 服務，請參閱《IAM 使用者指南》中的 [AWS 全域條件內容索引鍵](#)。

Amazon SNS 使用下列服務特定金鑰。在限制對 Subscribe 請求之存取的政策中使用這些金鑰。

- sns:endpoint - Subscribe 請求或先前已確認之訂閱的 URL、電子郵件地址或 ARN。使用字串條件 (請參閱 [用於 Amazon SNS 的政策範例](#)) 來限制對特定端點 (例如：*@yourcompany.com) 的存取。
- sns:protocol - protocol 請求或先前已確認之訂閱的 Subscribe 值。使用字串條件 (請參閱 [用於 Amazon SNS 的政策範例](#)) 來限制發佈至特定傳送協定 (例如：https)。

用於 Amazon SNS 的政策範例

本節顯示幾個控制使用者存取 Amazon SNS 的簡易政策。

 Note

未來，Amazon SNS 可能根據政策所聲明的目標，新增邏輯上應包含到下列其中一個政策中的動作。

Example 1: 允許群組建立和管理主題

在此範例中，我們建立授予 CreateTopic、ListTopics、SetTopicAttributes 和 DeleteTopic 存取權的政策。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
"sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: 允許 IT 群組發佈訊息到特定的主題

在此範例中，我們為 IT 建立群組，並指定授予對特定主題 Publish 之存取權的政策。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3：讓使用者 AWS 帳戶 能夠訂閱主題

在此範例中，我們使用符合 Subscribe 和 sns:Protocol 政策金鑰之條件的字串，建立授予 sns:Endpoint 動作之存取權的政策。

```
{
  "Statement": [{
    "Effect": "Allow",
```

```

    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "sns:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }
}]
}

```

Example 4: 允許合作夥伴發佈訊息到特定的主題

您可以使用 Amazon SNS 政策或 IAM 政策來允許合作夥伴發佈到特定的主題。如果您的合作夥伴有 AWS 帳戶，則使用 Amazon SNS 政策可能會更輕鬆。不過，擁有 AWS 安全登入資料之合作夥伴公司中的任何人，都可以將訊息發佈至主題。此範例假設您想要現這特定人員 (或應用程式) 的存取權。因此，您需要將合作夥伴當作您公司內的使用者來看待，改為使用 IAM 政策，而非 Amazon SNS 政策。

對於此範例，我們建立一個稱為 WidgetCo 的群組，代表合作夥伴公司；我們為合作夥伴公司需要存取權的特定人員建立使用者 (或應用程式)；然後我們將該使用者放入群組中。

然後，我們附加一個政策，提供群組名為 WidgetPartnerTopic 之特定主題的 Publish 存取權。

我們也希望防止 WidgetCo 群組對主題執行其他任何操作，所以我們新增一個陳述式明確拒絕任何對 WidgetPartnerTopic 以外之任何主題進行 Publish 以外的 Amazon SNS 動作的許可。只有當系統中的其他地方使用廣泛的政策，讓使用者廣泛存取 Amazon SNS 時，此步驟才為必要。

```

{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
  },
  {
    "Effect": "Deny",
    "NotAction": "sns:Publish",
    "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
  }
]
}

```

管理自訂 Amazon SNS IAM 政策

自訂 IAM 政策可讓您指定個別 IAM 使用者、群組或角色的許可，授予或限制對特定 AWS 資源和動作的存取。管理 Amazon SNS 資源時，自訂 IAM 政策可讓您根據組織的安全和操作需求量身打造存取許可。

使用下列步驟來管理 Amazon SNS 的自訂 IAM 政策：

1. 登入 AWS Management Console，並在 <https://console.aws.amazon.com/iam/> : //www. 開啟 IAM 主控台。
2. 從導覽窗格中，選擇政策。
3. 若要建立新的自訂 IAM 政策，請選擇建立政策，然後選擇 SNS。若要編輯現有政策，請從清單中選擇政策，然後選擇編輯政策。
4. 在政策編輯器中，定義存取 Amazon SNS 資源的許可。您可以根據您的特定需求指定動作、資源和條件。
5. 若要授予 Amazon SNS 動作的許可，請在 IAM 政策 `sns:DeleteTopic` 中包含相關的 Amazon SNS 動作 `sns:Publish`，例如 `sns:Subscribe`、和。定義許可套用的 Amazon SNS 主題的 ARN (Amazon Resource Name)。
6. 指定政策應連接的 IAM 使用者、群組或角色。您可以直接將政策連接至 IAM 使用者或群組，或將其與 AWS 服務或應用程式使用的 IAM 角色建立關聯。
7. 檢閱 IAM 政策組態，以確保其符合您存取控制需求。驗證後，儲存政策變更。
8. 將自訂 IAM 政策連接至中的相關 IAM 使用者、群組或角色 AWS 帳戶。這會授予他們管理 Amazon SNS 資源的政策中定義的許可。

使用臨時安全登入資料搭配 Amazon SNS

AWS Identity and Access Management (IAM) 可讓您將臨時安全登入資料授予需要存取 AWS 資源的使用者和應用程式。這些臨時安全登入資料主要用於 IAM 角色，以及透過 SAML 和 OpenID Connect (OIDC) 等產業標準通訊協定的聯合存取。

若要有效管理對 AWS 資源的存取，請務必了解下列重要概念：

- IAM 角色 – 角色用於委派對 AWS 資源的存取。角色可由 Amazon EC2 執行個體、Lambda 函數或其他使用者等實體擔任 AWS 帳戶。
- 聯合身分使用者 – 這些是使用 SAML 或 OIDC 透過外部身分提供者 (IdPs) 驗證的使用者。建議人類使用者進行聯合存取，而 IAM 角色應該用於軟體應用程式。

- Roles Anywhere – 對於需要 AWS 存取的外部應用程式，您可以使用 IAM Roles Anywhere 安全地管理存取，而無需建立長期憑證。

您可以使用臨時安全登入資料向 Amazon SNS 提出請求。SDKs和 API 程式庫使用這些登入資料來計算必要的簽章，以驗證您的請求。Amazon SNS 將拒絕憑證過期的請求。

如需臨時安全登入資料的詳細資訊，請參閱 [《IAM 使用者指南》中的使用 IAM 角色](#) 和 [提供存取權給外部驗證的使用者（聯合身分）](#)。

Example HTTPS 請求範例

下列範例示範如何使用從 AWS Security Token Service (STS) 取得的臨時安全登入資料來驗證 Amazon SNS 請求。

```
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>
```

驗證請求的步驟

1. 取得臨時安全登入資料 – AWS STS 用於擔任角色或取得聯合身分使用者登入資料。這將為您提供存取金鑰 ID、私密存取金鑰和安全性字符。
2. 建構請求 – 包含 Amazon SNS 動作的必要參數（例如 CreateTopic），並確保您使用 HTTPS 進行安全通訊。
3. 簽署請求 – 使用 AWS Signature 第 4 版程序簽署您的請求。這包括建立正式請求、string-to-sign，然後計算簽章。如需 AWS Signature 第 4 版的詳細資訊，請參閱《Amazon EBS 使用者指南》中的[使用 Signature 第 4 版簽署](#)。
4. 傳送請求 – 在您的請求標頭 X-Amz-Security-Token 中包含，以將臨時安全登入資料傳遞至 Amazon SNS。

Amazon SNS API 許可：動作和資源參考

以下清單提供 Amazon SNS 實作存取控制的特定資訊。

- 每項政策都必須只涵蓋一個主題 (在撰寫政策時，請勿包括涵蓋不同主題的聲明)
- 每項政策必須有唯一的政策 Id
- 政策中的每個聲明必須有唯一的聲明 sid

政策配額

下表列出政策陳述式的最大配額。

名稱	最大配額
位元組	30 kb
聲明	100
委託人	1 至 200 (0 為無效。)
資源	1 (0 為無效。值必須符合政策主題的 ARN。)

有效的 Amazon SNS 政策動作

Amazon SNS 支援下表所示的動作。

動作	描述
sns:AddPermission	授與新增許可至主題政策的許可。
sns>DeleteTopic	授與刪除主題的許可。
sns:GetDataProtectionPolicy	授與擷取主題資料保護政策的許可。
sns:GetTopicAttributes	授與接收所有主題屬性的許可。
sns:ListSubscriptionsByTopic	授與擷取所有訂閱至特定主題的許可。
sns:ListTagsForResource	許可列出所有新增至特定主題的標籤。

動作	描述
sns:Publish	授與發佈和批次發佈至主題或端點的許可。如需詳細資訊，請參閱《Amazon Simple Notification Service API 參考》中的 Publish 和 PublishBatch 。
sns:PutDataProtectionPolicy	授與設定主題資料保護政策的許可。
sns:RemovePermission	授與移除主題政策中任何許可的許可。
sns:SetTopicAttributes	授與設定主題屬性的許可。
sns:Subscribe	授與訂閱主題的許可。

服務特定金鑰

Amazon SNS 使用下列服務特定金鑰。您可以在限制對 Subscribe 請求的存取的政策中使用這些。

- sns:endpoint - Subscribe 請求或先前已確認之訂閱的 URL、電子郵件地址或 ARN。使用字串條件 (請參閱 [用於 Amazon SNS 的政策範例](#)) 來限制對特定端點 (例如：*@example.com) 的存取。
- sns:protocol - protocol 請求或先前已確認之訂閱的 Subscribe 值。使用字串條件 (請參閱 [用於 Amazon SNS 的政策範例](#)) 來限制發佈至特定傳送協定 (例如：https)。

Important

當您使用政策來控制 sns:Endpoint 的存取，請注意 DNS 問題可能會影響未來端點的名稱解析。

對 Amazon Simple Notification Service 身分和存取進行故障診斷

請使用以下資訊來協助您診斷和修正使用 Amazon SNS 和 IAM 時可能遇到的常見問題。

我未獲授權，不得在 Amazon SNS 中執行動作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在 mateojackson 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 sns:*GetWidget* 許可時發生。


```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
sns:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 Mateo 政策，允許他使用 `sns:GetWidget` 動作存取 `my-example-widget` 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我未獲得執行 `iam:PassRole` 的授權

如果您收到錯誤，告知您無權執行 `iam:PassRole` 動作，您的政策必須更新，允許您將角色傳遞給 Amazon SNS。

有些 AWS 服務可讓您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為 `marymajor` 的 IAM 使用者嘗試使用主控台在 Amazon SNS 中執行動作時，發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想要允許以外的人員 AWS 帳戶存取我的 Amazon SNS 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon SNS 是否支援這些功能，請參閱 [Amazon SNS 如何與 IAM 搭配使用](#)。
- 若要了解如何 AWS 帳戶在您的擁有的資源之間提供存取權，請參閱《[IAM 使用者指南](#)》中的[在您的擁有 AWS 帳戶的另一個 IAM 使用者中提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《[IAM 使用者指南](#)》中的[將存取權提供給第三方 AWS 帳戶擁有](#)。

- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的[將存取權提供給在外部進行身分驗證的使用者 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的[IAM 中的跨帳戶資源存取](#)。

在 Amazon SNS 中記錄和監控

Amazon SNS 可讓您使用 CloudTrail 記錄 API 呼叫，以及使用 CloudWatch 監控主題，以追蹤和監控訊息活動。這些工具可協助您深入了解訊息傳遞、疑難排解問題，並確保訊息工作流程的運作狀態。本主題涵蓋下列項目：

- [使用記錄 AWS SNS API 呼叫 AWS CloudTrail](#)。此記錄可讓您追蹤在 Amazon SNS 主題上執行的動作，例如主題建立、訂閱管理和訊息發佈。透過分析 CloudTrail 日誌，您可以識別提出特定 API 請求的人員，以及提出這些請求的時間，協助您稽核和疑難排解 Amazon SNS 使用情況。
- [使用 Amazon CloudWatch 監控 Amazon SNS 主題](#)。CloudWatch 提供指標，可讓您即時觀察 Amazon SNS 主題的效能和運作狀態。根據這些指標設定警示，讓您能夠快速回應任何異常，例如交付失敗或高訊息延遲。此監控功能可確保您可以透過主動解決潛在問題來維護 SNS 型傳訊系統的可靠性。

使用記錄 AWS SNS API 呼叫 AWS CloudTrail

AWS SNS 已與整合[AWS CloudTrail](#)，這項服務提供由使用者、角色或所採取動作的記錄 AWS 服務。CloudTrail 會將 SNS 的所有 API 呼叫擷取為事件。擷取的呼叫包括 SNS 主控台的呼叫，以及對 SNS API 操作的程式碼呼叫。您可以使用 CloudTrail 所收集的資訊，判斷向 SNS 提出的請求、提出請求的 IP 地址、提出請求的時間，以及其他詳細資訊。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 該請求是否使用根使用者還是使用者憑證提出。
- 請求是否代表 IAM Identity Center 使用者提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 該請求是否由另一項 AWS 服務提出。

當您建立帳戶 AWS 帳戶時 CloudTrail 會在中處於作用中狀態，而且您會自動存取 CloudTrail 事件歷史記錄。CloudTrail 事件歷史記錄為 AWS 區域中過去 90 天記錄的管理事件，提供可檢視、可搜尋、

可下載且不可變的記錄。如需詳細資訊，請參閱「AWS CloudTrail 使用者指南」中的[使用 CloudTrail 事件歷史記錄](#)。檢視事件歷史記錄不會產生 CloudTrail 費用。

如需 AWS 帳戶 過去 90 天內持續記錄的事件，請建立線索或 [CloudTrail Lake](#) 事件資料存放區。

CloudTrail 追蹤

線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。使用 建立的所有線索 AWS Management Console 都是多區域。您可以使用 AWS CLI 建立單一或多區域追蹤。建議您建立多區域線索，因為您擷取 AWS 區域 帳戶中所有的活動。如果您建立單一區域追蹤，您只能檢視追蹤 AWS 區域中記錄的事件。如需追蹤的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的[為您](#) [的 AWS 帳戶建立追蹤](#)和[為組織建立追蹤](#)。

您可以透過建立追蹤，免費將持續管理事件的一個複本從 CloudTrail 傳遞至您的 Amazon S3 儲存貯體，但這樣做會產生 Amazon S3 儲存費用。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。如需 Amazon S3 定價的相關資訊，請參閱 [Amazon S3 定價](#)。

CloudTrail Lake 事件資料存放區

CloudTrail Lake 讓您能夠對事件執行 SQL 型查詢。CloudTrail Lake 會將分列式 JSON 格式的現有事件轉換為 [Apache ORC](#) 格式。ORC 是一種單欄式儲存格式，針對快速擷取資料進行了最佳化。系統會將事件彙總到事件資料存放區中，事件資料存放區是事件的不可變集合，其依據為您透過套用[進階事件選取器](#)選取的條件。套用於事件資料存放區的選取器控制哪些事件持續存在並可供您查詢。如需 CloudTrail Lake 的詳細資訊，請參閱 AWS CloudTrail 《使用者指南》中的[使用 AWS CloudTrail Lake](#)。

CloudTrail Lake 事件資料存放區和查詢會產生費用。建立事件資料存放區時，您可以選擇要用於事件資料存放區的[定價選項](#)。此定價選項將決定擷取和儲存事件的成本，以及事件資料存放區的預設和最長保留期。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。

CloudTrail 中的 SNS 資料事件

[資料事件](#)提供在資源上或在資源中執行的資源操作的相關資訊 (例如，讀取或寫入 Amazon S3 物件)。這些也稱為資料平面操作。資料事件通常是大量資料的活動。根據預設，CloudTrail 不會記錄資料事件。CloudTrail 事件歷史記錄不會記錄資料事件。

資料事件需支付額外的費用。如需 CloudTrail 定價的詳細資訊，請參閱 [AWS CloudTrail 定價](#)。

您可以使用 CloudTrail 主控台 AWS CLI 或 CloudTrail API 操作來記錄 SNS 資源類型的資料事件。如需如何記錄資料事件的詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的[使用 AWS Management Console 記錄資料事件](#)和[使用 AWS Command Line Interface 記錄資料事件](#)。

下表列出您可以記錄資料事件的 SNS 資源類型。資料事件類型 (主控台) 資料行會顯示從 CloudTrail 主控台上的資料事件類型清單中選擇的值。resources.type 值欄會顯示值，您會在使用 AWS CLI 或 CloudTrail APIs 設定進階事件選取器時指定該resources.type值。記錄到 CloudTrail 的資料 API 資料行會針對資源類型顯示記錄到 CloudTrail 的 API 呼叫。

資料事件類型 (主控台)	resources.type 值	記錄到 CloudTrail 的資料 API
SNS 主題	AWS::SNS::Topic	• Publish • PublishBatch
SNS 平台端點	AWS::SNS::Platform Endpoint	• Publish 如需其他詳細資訊，請參閱《AWS CloudTrail API 參考AdvancedEventSelector》中的。

Note

CloudTrail AWS::SNS::PhoneNumber 不會記錄 SNS 資源類型。

您可以設定進階事件選取器來篩選 eventName、readOnly 和 resources.ARN 欄位，以僅記錄對您重要的事件。如需這些欄位的詳細資訊，請參閱AWS CloudTrail API 參考中的[AdvancedFieldSelector](#)。

如需有關記錄資料事件的資訊，請參閱《CloudTrail 使用者指南 AWS CLI》中的使用 記錄資料事件 AWS Management Console 和使用 記錄資料事件。

CloudTrail 中的 SNS 管理事件

[管理事件](#)提供在 資源上執行的管理操作的相關資訊 AWS 帳戶。這些也稱為控制平面操作。根據預設，CloudTrail 記錄管理事件。

AWS SNS 會將下列 SNS 控制平面操作記錄到 CloudTrail 做為管理事件。

- [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

當您未登入 Amazon Web Services（未驗證模式）且 [ConfirmSubscription](#) 或 [Unsubscribe](#) 動作被叫用時，則不會將其記錄到 CloudTrail。例如，當您選擇電子郵件通知提供的連結，以確認對某一主題的等待訂閱時，系統會以未驗證模式叫用 `ConfirmSubscription` 動作。在此範例中，`ConfirmSubscription` 動作不會記錄到 CloudTrail 中。

SNS 事件範例

一個事件代表任何來源提出的單一請求，並包含請求 API 操作的相關資訊、操作的日期和時間、請求參數等。CloudTrail 日誌檔案不是公有 API 呼叫的已排序堆疊追蹤，因此事件不會以任何特定順序顯示。

下列範例顯示示範 **ListTopics**、**CreateTopic** 和 **DeleteTopic** 動作的 CloudTrail 事件。

```
{
  "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"
  }
]
}

```

下列範例顯示示範 **Publish** 動作的 CloudTrail 事件。

```

{
  "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"
  }
}

```

```
}
```

下列範例顯示示範 **PublishBatch** 動作的 CloudTrail 事件。

```
{
  "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"
      }
    },
    "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"
  }
}

```

如需有關 CloudTrail 記錄內容的資訊，請參閱《AWS CloudTrail 使用者指南》中的 [CloudTrail record contents](#)。

使用 Amazon CloudWatch 監控 Amazon SNS 主題

Amazon SNS 和 Amazon CloudWatch 已經整合，讓您可收集、檢視和分析每個作用中 Amazon SNS 通知的指標。一旦您為 Amazon SNS 設定好 CloudWatch 之後，就可以對於您的 Amazon SNS 主題、推送通知和簡訊傳送獲得更深入的效能洞察。例如，您可以設定警示，在達到 Amazon SNS 指標的指定閾值 (例如 `NumberOfNotificationsFailed`) 時傳送電子郵件通知。有關 Amazon SNS 傳送至 CloudWatch 之所有指標的清單，請參閱 [Amazon SNS 指標](#)。如需使用 Amazon SNS 推送通知的詳細資訊，請參閱 [使用 Amazon SNS 傳送行動推播通知](#)。

Note

自動收集您為 Amazon SNS 主題設定 CloudWatch 的指標，並在 1 分鐘間隔推送到 CloudWatch。這些指標會針對所有符合 CloudWatch 作用中的要求的主題進行收集。被 CloudWatch 視為作用中的主題為自上次在主題上活動 (例如任何 API 呼叫) 已長達六小時者。CloudWatch 中所報告的 Amazon SNS 指標不會產生費用；這是做為 Amazon SNS 服務的一部分提供。

檢視 Amazon SNS 的 CloudWatch 指標

您可以使用 CloudWatch 主控台、CloudWatch 本身的命令列界面 (CLI) 或使用 CloudWatch API 的程式設計方式來監控 Amazon SNS 指標。下列程序顯示如何使用 AWS Management Console 存取指標。

使用 CloudWatch 主控台檢視指標

1. 登入 [CloudWatch 主控台](#)。
2. 在導覽面板上，選擇 Metrics (指標)。
3. 在 All metrics (所有指標) 標籤上，選擇 SNS，然後選擇下列其中一個維度：
 - Country, SMS Type (國家/地區，簡訊類型)
 - PhoneNumber
 - Topic Metrics (主題指標)
 - Metrics with no dimensions (無維度的指標)
4. 若要檢視更多詳細資訊，請選擇特定項目。例如，如果您選擇 Topic Metrics (主題指標)，然後選擇 `NumberOfMessagesPublished`，則會顯示 6 小時範圍內一分鐘時間所發佈 Amazon SNS 訊息的平均數量。

- 若要檢視 Amazon SNS 用量指標，請在 All metrics (所有指標) 索引標籤上選擇 Usage (用量)，然後選取 target Amazon SNS usage metric (針對 Amazon SNS 用量指標) (例如 NumberOfMessagesPublishedPerAccount)。

設定 Amazon SNS 指標的 CloudWatch 警示

CloudWatch 亦可讓您設定到達指標的閾值時的警示。例如，您可以設定指標 NumberOfNotificationsFailed 的警示，以便在抽樣週期內到達指定的閾值時，傳送電子郵件通知給您通知您發生事件。

使用 CloudWatch 主控台設定警示

- 登入 AWS Management Console，並在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
- 選擇 Alarms (警示)，然後選擇 Create Alarm (建立警示) 按鈕。這會啟動 Create Alarm (建立警示) 精靈。
- 捲動 Amazon SNS 指標以找到您要設定警示的指標。選取要建立警示的指標，然後選擇 Continue (繼續)。
- 填入指標的 Name (名稱)、Description (描述)、Threshold (閾值) 和 Time (時間) 值，然後選擇 Continue (繼續)。
- 選擇 Alarm (警示) 做為警示狀態。如果您希望 CloudWatch 在達到警示狀態時傳送電子郵件給您，請選擇現有的 Amazon SNS 主題或選擇 Create New Email Topic (建立新的電子郵件主題)。如果您選擇 Create New Email Topic (建立新的電子郵件主題)，即可為新的主題設定名稱和電子郵件地址。此清單將會儲存並顯示在下拉式方塊中，供未來警示用。選擇 Continue (繼續)。

Note

如果您使用 Create New Email Topic (建立新的電子郵件主題) 來建立新的 Amazon SNS 主題，電子郵件地址必須先經過驗證才會接收通知。電子郵件只有在警示進入警示狀態時才會傳送。如果此警示狀態在驗證電子郵件地址之前發生變更，就不會收到通知。

- 此時，Create Alarm (建立警示) 精靈會提供您機會檢閱您將建立的警示。如果您需要進行任何變更，可以使用右邊的 Edit (編輯) 連結。編輯好之後，選擇 Create Alarm (建立警示)。

如需使用 CloudWatch 和警示的詳細資訊，請參閱 [CloudWatch 文件](#)。

Amazon SNS 指標

Amazon SNS 會傳送下列指標至 CloudWatch。

命名空間	指標	描述
AWS/SNS	NumberOfMessagesPublished	發佈至 Amazon SNS 主題的訊息數量。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和
AWS/SNS	NumberOfNotificationsDelivered	成功從 Amazon SNS 主題傳送至訂閱端點的訊息數量。 為了讓傳遞嘗試成功，端點的訂閱必須接受訊息。訂閱接受訊息的情況為 a.) 它缺少篩選條件政策或 b.) 其篩選條件政策包含符合指派給該訊息的屬性。如果訂閱拒絕訊息，此指標不會計入傳遞嘗試次數。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和
AWS/SNS	NumberOfNotificationsFailed	Amazon SNS 無法傳送之訊息的數量。 對於 Amazon SQS、電子郵件、簡訊或行動推送端點，當 Amazon

命名空間	指標	描述
		SNS 停止嘗試傳遞訊息，此指標會以 1 的方式遞增。對於 HTTP 或 HTTPS 端點，指標包含每個失敗的傳遞嘗試，包括首次嘗試之後的重試次數。對於所有其他端點，計數會在訊息傳遞失敗時增加 1 (不論嘗試次數多寡)。 這個指標不包含被訂閱篩選政策拒絕的訊息。 您可以控制 HTTP 端點的重試數目。如需詳細資訊，請參閱Amazon SNS 訊息傳遞重試。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均
AWS/SNS	NumberOfNotificationsFilteredOut	被訂閱篩選政策拒絕的訊息數。當訊息屬性不符政策屬性時，篩選政策會拒絕訊息。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均

命名空間	指標	描述
AWS/SNS	NumberOfNotificationsFilteredOut-MessageAttributes	被以屬性為基礎篩選的訂閱篩選政策拒絕的訊息數。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均
AWS/SNS	NumberOfNotificationsFilteredOut-MessageBody	被以承載為基礎篩選的訂閱篩選政策拒絕的訊息數。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidAttributes	因為訊息屬性無效而被訂閱篩選條件政策拒絕的訊息數量，例如，因為屬性 JSON 的格式不正確。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均

命名空間	指標	描述
AWS/SNS	NumberOfNotificationsFilteredOut-NoMessageAttributes	因為訊息沒有屬性而被訂閱篩選條件政策拒絕的訊息數量。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidMessageBody	因訊息內文無效而被訂閱篩選政策拒絕的訊息數，例如，JSON 訊息內文無效。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均
AWS/SNS	NumberOfNotificationsRedrivenToDlq	已移至無效字母佇列的訊息數量。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均

命名空間	指標	描述
AWS/SNS	NumberOfNotificationsFailedToRedriveToDlq	無法移至無效字母佇列的訊息數量。 單位：計數 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：總和、平均
AWS/SNS	PublishSize	已發佈訊息的大小。 單位：位元組 有效維度：Application、PhoneNumber、Platform 和 TopicName 有效統計資料：最小值、最大值、平均值和計數

命名空間	指標	描述
AWS/SNS	SMSMonthToDateSpentUSD	您從當月月初開始傳送簡訊所累計的費用。 您可以為此指標設定警示，以得知本月至今的費用是否已接近您帳戶的每月簡訊費用配額。當 Amazon SNS 判定傳送簡訊會致使成本超過此配額時，就會在幾分鐘內停止發步簡訊。 如需有關每月簡訊費用配額的設定資訊，或者要求 AWS 提高費用配額的相關資訊，請參閱 在 Amazon SNS 中設定簡訊偏好設定。 單位：USD 有效維度：無 有效統計資料：總和
AWS/SNS	SMSSuccessRate	成功傳送簡訊的比率。 單位：計數 有效維度：PhoneNumber 有效統計資料：總和、平均、資料範例

Amazon SNS 指標的維度

Amazon Simple Notification Service 會傳送下列維度至 CloudWatch。

維度	描述
Application	篩選應用程式物件，該物件代表向其中一個支援的推送通知服務 (例如 APN 和 FCM) 註冊的應用程式和裝置。

維度	描述
Application, Platform	篩選應用程式和平台物件，其中平台物件適用於支援的推送通知服務，例如 APN 和 FCM。
Country	篩選簡訊的目的地國家/地區或區域。國家/地區或區域是以其 ISO 3166-1 alpha-2 代碼表示。
PhoneNumber	當您直接將簡訊發佈至電話號碼 (不含主題) 時，篩選電話號碼。
Platform	篩選適用於推送通知服務 (例如 APN 和 FCM) 的平台物件。
TopicName	篩選 Amazon SNS 主題名稱。
SMSType	篩選簡訊的訊息類型。可以是促銷型或交易型。

Amazon SNS 用量指標

Amazon Simple Notification Service 會傳送下列用量指標至 CloudWatch。

命名空間	服務	指標	資源	類型	描述
AWS/用量	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	資源	- 在您的 AWS 帳戶中發佈至 Amazon SNS 主題的訊息數量。 - 單位：無 - 有效的統計資訊：總和
AWS/用量	SNS	ResourceCount	ApproximateNumberOfTopics	資源	- 您 AWS 帳戶中的大致主題數量。 - 單位：無

命名空間	服務	指標	資源	類型	描述
					有效統計資訊：平均數、下限、上限、總和
AWS/用量	SNS	ResourceCount	ApproximateNumberOfFilterPolicies	資源	- 您 AWS 帳戶中的大致篩選政策數量。 - 單位：無 - 有效統計資訊：平均數、下限、上限、總和
AWS/用量	SNS	ResourceCount	ApproximateNumberOfPendingSubscriptions	資源	- 您 AWS 帳戶中等待訂閱的大致數量。 - 單位：無 - 有效統計資訊：平均數、下限、上限、總和

命名空間	服務	指標	資源	類型	描述
AWS/用量	SNS	CallCount	- AddPermission - CheckIfPhoneNumberIsOptedOut - CreatePlatformApplication - CreatePlatformEndpoint - ConfirmSubscription - CreateSMSSandboxPhoneNumber - CreateTopic - DeleteEndpoint - DeletePlatformApplication - DeleteSMSSandboxPhoneNumber - DeleteTopic	API	- 您 AWS 帳戶中所選 Amazon SNS API 的 API 呼叫次數。 - 結尾區段不允許內容。 單位：無 - 有效的統計資訊：總和

命名空間	服務	指標	資源	類型	描述
			• <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>		

命名空間	服務	指標	資源	類型	描述
			- ListSMSSandboxPhoneNumbers - ListSubscriptions - ListSubscriptionsByTopic - ListTagsForResource - ListTopics - OptInPhoneNumber - RemovePermission - SetEndpointAttributes - SetPlatformApplicationAttributes - SetSMSAttributes - SetSubscriptionAttributes - SetTopicAttributes		

命名空間	服務	指標	資源	類型	描述
			- Subscribe - Unsubscribe - UntagResource - VerifySMSSandboxPhoneNumber		

Amazon SNS 的合規驗證

若要了解 是否 AWS 服務 在特定合規計劃的範圍內，請參閱[AWS 服務 合規計劃範圍內](#)然後選擇您感興趣的合規計劃。如需一般資訊，請參閱 [AWS Compliance Programs](#)。

您可以使用 下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱[下載報告 in AWS Artifact](#)

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。AWS 提供下列資源來協助合規：

- [安全合規與治理](#) - 這些解決方案實作指南內容討論了架構考量，並提供部署安全與合規功能的步驟。
- [HIPAA 合格服務參考](#) - 列出 HIPAA 合格服務。並非所有 AWS 服務 都符合 HIPAA 資格。
- [AWS 合規資源](#) - 此工作手冊和指南集合可能適用於您的產業和位置。
- [AWS 客戶合規指南](#) - 透過合規的角度了解共同責任模型。本指南摘要說明跨多個架構（包括國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 保護 AWS 服務 和映射指南至安全控制的最佳實務。
- 《AWS Config 開發人員指南》中的[使用規則評估資源](#) - AWS Config 服務會評估資源組態符合內部實務、產業準則和法規的程度。
- [AWS Security Hub](#) - 這 AWS 服務 可讓您全面檢視其中的安全狀態 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱「[Security Hub 控制參考](#)」。

- [Amazon GuardDuty](#) – 這會監控您的環境是否有可疑和惡意活動，以 AWS 服務 偵測對您 AWS 帳戶、工作負載、容器和資料的潛在威脅。GuardDuty 可滿足特定合規架構所規定的入侵偵測需求，以協助您因應 PCI DSS 等各種不同的合規需求。
- [AWS Audit Manager](#) – 這 AWS 服務 可協助您持續稽核 AWS 用量，以簡化您管理風險的方式，以及符合法規和業界標準的方式。

Amazon SNS 的恢復能力

Amazon SNS 中的彈性是透過利用 AWS 全球基礎設施來確保，該基礎設施圍繞 AWS 區域 和可用區域。AWS 區域 提供以低延遲、高輸送量和高備援聯網連接的實際隔離和隔離可用區域。此架構允許在可用區域之間無縫容錯移轉，而不會中斷，因此與傳統資料中心基礎設施相比，應用程式和資料庫本質上可容錯且可擴展。透過使用可用區域，Amazon SNS 訂閱者可從增強的可用性和可靠性中受益，即使潛在的中斷，仍保證訊息交付。如需 AWS 區域 和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

此外，Amazon SNS 主題的訂閱可以設定交付重試和無效字母佇列，以便自動處理暫時性故障，並確保訊息可靠地到達其預期目的地。

Amazon SNS 也支援訊息篩選和訊息屬性，可讓您根據其特定使用案例量身打造彈性策略，增強應用程式的整體健全性。

Amazon SNS 的基礎設施安全性

Amazon SNS 是受管服務，受到[安全性、身分和合規最佳實務](#)文件中的 AWS 全球網路安全程序保護。

使用 AWS API 動作透過網路存取 Amazon SNS。用戶端必須支援 Transport Layer Security (TLS) 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。

您必須使用存取金鑰 ID 和與 IAM 委託人相關聯的私密存取金鑰來簽署請求。或者，您可以使用 [AWS Security Token Service](#) (AWS STS) 來產生暫時安全登入資料以便簽署請求。

您可從任何聯網位置呼叫這些 API 動作，但 Amazon SNS 支援資源型存取政策，這類政策可以根據來源 IP 地址納入限制。您也可以使用 Amazon SNS 政策從特定 Amazon VPC 端點或特定 VPC 控制存取。這可有效地隔離網路對指定 Amazon SNS 主題的網路存取，以及 AWS 網路內的特定 VPC。如需詳細資訊，請參閱[限制僅從特定 VPC 端點發佈至 Amazon SNS 主題](#)。

Amazon SNS 的安全最佳實務

AWS 為 Amazon SNS 提供許多安全功能。在您自己的安全政策內容中檢閱這些安全功能。

Note

這些安全功能的指引適用於一般使用案例和實作。我們建議您在特定使用案例、架構和威脅模型的內容中檢閱這些最佳實務。

預防性最佳實務

以下是 Amazon SNS 的預防性安全最佳實務。

主題

- [確保主題無法公開存取](#)
- [實作最低權限存取](#)
- [針對需要 Amazon SNS 存取的應用程式 AWS 和服務使用 IAM 角色](#)
- [實作伺服器端加密](#)
- [強制加密傳輸中的資料](#)
- [考慮使用 VPC 端點來存取 Amazon SNS](#)
- [確保訂閱未設定為傳遞至原始 http 端點](#)

確保主題無法公開存取

除非您明確要求網際網路上的任何人都能讀取或寫入您的 Amazon SNS 主題，否則您應該確保您的主題不可公開存取（全球所有人或任何已驗證 AWS 的使用者皆可存取）。

- 避免建立 Principal 設定為 "" 的政策。
- 避免使用萬用字元 (*)。請改為命名特定使用者或使用者。

實作最低權限存取

當您授與許可時，您可決定接收許可的人員、許可適用的主題，以及您要對這些主題允許的特定 API 動作。實作最低權限原則對於降低安全風險非常重要。它還有助於減少錯誤或惡意意圖的負面影響。

遵循授與最低權限的標準安全建議。也就是說，只授與執行特定工作所需的許可。您可以使用與使用者存取相關的安全政策組合來實作最低權限。

Amazon SNS 會使用發布者訂閱者模型，並需要三種類型的使用者帳戶存取權：

- 管理員 – 建立、修改和刪除主題的存取權。管理員也會控制主題政策。
- 發布者 – 傳送訊息至主題的存取權。
- 訂閱者 – 訂閱主題的存取權。

如需詳細資訊，請參閱下列章節：

- [Amazon SNS 中的 Identity and Access Management](#)
- [Amazon SNS API 許可：動作和資源參考](#)

針對需要 Amazon SNS 存取的應用程式 AWS 和服務使用 IAM 角色

對於 Amazon EC2 等應用程式 AWS 或服務，若要存取 Amazon SNS 主題，他們必須在其 AWS API 請求中使用有效的 AWS 登入資料。由於這些登入資料不會自動輪換，因此您不應該直接將 AWS 登入資料存放在應用程式或 EC2 執行個體中。

反之，您應使用 IAM 角色來為需存取 Amazon SNS 的應用程式和服務管理暫時性登入資料。使用角色時，您不必將長期憑證 (如使用者名稱、密碼和存取金鑰) 分發至 EC2 執行個體或 AWS Lambda 等 AWS 服務。相反地，該角色會提供暫時許可，供應用程式在呼叫其他 AWS 資源時使用。

如需詳細資訊，請參閱 [IAM 角色](#) 和 [常見的角色方案：使用者、應用程式和服務](#) 中的 IAM 使用者指南。

實作伺服器端加密

若要減輕資料外洩問題，請使用靜態加密，並利用與訊息儲存在不同位置的金鑰來加密訊息。伺服器端加密 (SSE) 會提供靜態資料加密。Amazon SNS 會在儲存資料時於訊息層級進行加密，並在您存取訊息時為您進行解密。SSE 使用中管理的金鑰 AWS Key Management Service。當您驗證請求並具備存取許可時，存取加密與未加密主題的方式並無不同。

如需更多詳細資訊，請參閱 [使用伺服器端加密保護 Amazon SNS 資料](#) 及 [管理 Amazon SNS 加密金鑰和成本](#)。

強制加密傳輸中的資料

有可能 (但不建議) 發佈使用 HTTP 在傳輸期間未加密的訊息。不過，使用靜態加密主題時 AWS KMS，必須使用 HTTPS 來發佈訊息，以確保靜態和傳輸中加密。雖然主題不會自動拒絕 HTTP 訊息，但必須使用 HTTPS 來維護安全標準。

AWS 建議您使用 HTTPS 而非 HTTP。使用 HTTPS 時，郵件會在傳輸期間自動加密，即使 SNS 主題本身並未加密。如果沒有 HTTPS，則網路型攻擊者可以利用中間人之類的攻擊，竊聽網路流量或操控它。

若只要透過 HTTPS 強制執行加密連線，請將 [aws:SecureTransport](#) 新增至未加密 SNS 主題的 IAM 政策中的條件。這會強制訊息發佈者使用 HTTPS 而非 HTTP。您可使用以下範例政策做為指南：

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:1234567890:test-topic"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

考慮使用 VPC 端點來存取 Amazon SNS

如果您的主題必須能夠與之互動，但是這些主題絕對不能公開於網際網路，請使用 VPC 端點，將主題存取權限制為特定 VPC 內的主機。您可以使用主題政策，從特定的 VPC 端點或特定的 VPC 控制對主題的存取。

Amazon SNS VPC 端點會提供兩種控制訊息存取的方式：

- 您可以控制經由特定 VPC 端點允許的請求、使用者或群組。
- 您可以使用主題政策，控制哪些 VPC 或 VPC 端點可存取您的主題。

如需更多詳細資訊，請參閱 [建立端點](#) 及 [為 Amazon SNS 建立 VPC 端點政策](#)。

確保訂閱未設定為傳遞至原始 http 端點

避免將訂閱設定為傳遞至原始 http 端點。永遠有訂閱傳遞至端點網域名稱。例如，設定為傳遞至端點的訂閱 `http://1.2.3.4/my-path`，應該更改為 `http://my.domain.name/my-path`。

使用 對 Amazon SNS 主題進行故障診斷 AWS X-Ray

AWS X-Ray 會收集您應用程式提供之請求的資料，並可讓您檢視和篩選資料，以識別潛在問題和最佳化的機會。您可以查看任何應用程式追蹤請求的詳細資訊；包含請求和回應，以及對下游 AWS 資源、微服務、資料庫和 HTTP Web API 的呼叫等相關資訊。

您可以使用 X-Ray 和 Amazon SNS 來追蹤並分析透過您應用程式傳輸的訊息。您可以使用 AWS Management Console 來檢視 Amazon SNS 與應用程式使用之其他服務之間的連線映射。您也可以使用主控台來檢視指標，例如平均延遲和失敗率。如需更多詳細資訊，請參閱 AWS X-Ray 開發人員指南中的[將 Amazon SNS 與 AWS X-Ray 搭配使用](#)。

Amazon SNS 中的主動追蹤

使用 AWS X-Ray 追蹤和分析使用者在將 Amazon SNS 主題傳遞至 [Amazon Data Firehose](#)、[Amazon SQS](#) 和 HTTP/S [AWS Lambda](#) 端點訂閱時的要求。

使用 X-Ray，您可以取得每個請求的 end-to-end 檢視，讓您：

- 識別呼叫 Amazon SNS 主題的內容，以及訂閱下游的服務。
- 分析延遲，例如：
 - 處理之前在 Amazon SNS 主題中花費的時間。
 - 每個訂閱端點的交付時間。

Important

具有眾多訂閱的 Amazon SNS 主題可能達到大小限制，而且無法完全追蹤。如需追蹤文件大小限制的資訊，請參閱《AWS 一般參考》中的 [X-ray 服務配額](#)。

如果您從已被追蹤的服務呼叫 Amazon SNS API，即使 API 上未啟用 X-Ray 追蹤，Amazon SNS 也會傳遞追蹤。

Amazon SNS 支援標準和 FIFO 主題的 X-Ray 追蹤。您可以使用 [Amazon SNS 主控台](#)、[Amazon SNS SetTopicAttributes API](#)、[Amazon Simple Notification Service CLI 參考](#)，或 [AWS CloudFormation](#) 為 Amazon SNS 主題啟用 X-Ray。

要進一步了解將 Amazon SNS 與 X-Ray 搭配使用的更多資訊，請參閱《AWS X-Ray 開發人員指南》中的 [Amazon SNS 和 AWS X-Ray](#)。

主動追蹤許可

使用 Amazon SNS 主控台時，Amazon SNS 會嘗試為 Amazon SNS 主題建立必要的許可以呼叫 X-Ray。如果您沒有足夠的許可可以使用 Amazon SNS 主控台，則可以拒絕該嘗試。如需詳細資訊，請參閱 [Amazon SNS 中的 Identity and Access Management](#) 和 [Amazon SNS 存取控制的範例案例](#)。

使用 CLI 時，您必須手動設定許可。這些許可是使用資源策略設定的。有關在 X-Ray 中使用所需許可的詳細資訊，請參閱 [Amazon SNS 和 AWS X-Ray](#)。

使用 AWS 主控台在 Amazon SNS 主題上啟用主動追蹤

在 Amazon SNS 主題上啟用主動追蹤時，它會讀取追蹤 ID、根據追蹤 ID 將資料傳送給客戶，然後將追蹤 ID 傳播至下游服務。

1. 登入 [Amazon SNS 主控台](#)。
2. 選擇主題或建立新主題。如需建立主題的詳細資訊，請參閱[建立 Amazon SNS 主題](#)。
3. 在建立主題頁面的詳細資訊區段中，選擇主題類型：FIFO 或標準。
 - a. 輸入新主題的名稱 (Name)。
 - b. (選用) 為主題輸入 Display name (顯示名稱)。
4. 展開 Active tracing (主動追蹤)，然後選擇 Use active tracing (使用主動追蹤)。

在為您的 Amazon SNS 主題啟用 X-Ray 後，您可以使用 [X-Ray 服務映射](#) 檢視該主題的端到端追蹤和服務映射。

使用 AWS SDK 在 Amazon SNS 主題上啟用主動追蹤

下列程式碼範例示範如何使用適用於 Java 的 AWS 開發套件，在 Amazon SNS 主題上啟用主動追蹤。

```
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());
    }
}
```

使用 CLI 在 Amazon SNS AWS 主題上啟用主動追蹤

下列程式碼範例示範如何使用 CLI 在 Amazon SNS AWS 主題上啟用主動追蹤。

```
aws sns set-topic-attributes \
  --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
  --attribute-name TracingConfig \
  --attribute-value Active
```

使用 在 Amazon SNS 主題上啟用主動追蹤 AWS CloudFormation

下列 AWS CloudFormation 堆疊顯示如何在 Amazon SNS 主題上啟用主動追蹤。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

確認您的主題已啟用主動追蹤

您可以使用 Amazon SNS 主控台來驗證是否為您的主題啟用主動跟蹤，或者何時無法新增資源政策。

1. 登入 [Amazon SNS 主控台](#)。

2. 在左側導覽窗格中，選擇 Topics (主題)。
3. 在 Topics (主題) 頁面上，選擇一個主題。
4. 選擇 Integrations (整合) 索引標籤。

啟用主動追蹤後，會顯示綠色的 Active (作用中) 圖示。

5. 如果您已啟用主動追蹤，但沒有看到資源策略已新增，請選擇 Create policy (建立原則) 以新增其他必要許可。

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	

< [Subscription 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

測試主動追蹤

1. 登入 [Amazon SNS 主控台](#)。
2. 建立 Amazon SNS 主題。如需詳細作法，請參閱[使用 建立主題 AWS Management Console](#)。

3. 展開 Active tracing (主動追蹤)，然後選擇 Use active tracing (使用主動追蹤)。
4. 發佈訊息至 Amazon SNS 主題。如需詳細作法，請參閱[將訊息發布到使用 AWS Management Console 的 Amazon SNS 主題](#)。
5. 使用 [X-Ray 服務映射](#) 來檢視主題的端對端追蹤和服務映射。



Amazon SNS 文件歷史記錄

下表說明 Amazon Simple Notification Service 開發人員指南最近的變更。

服務功能有時會逐步推出至提供服務 AWS 的區域。我們僅針對第一個版本更新此文件。我們不會提供有關區域可用性的資訊，也不會宣布後續區域的推展情況。如需服務功能的區域可用性和訂閱更新通知的相關資訊，請參閱[最新消息 AWS](#)？。

變更	描述	日期
新增對雙堆疊 (IPv4 和 IPv6) 端點的支援	Amazon SNS 現在支援 API 請求的 IPv6，允許用戶端使用 IPv4, IPv6 或雙堆疊連線至公有端點。	2025 年 4 月 3 日
Amazon SNS 新增了 Amazon SNS FIFO 主題的 FifoThroughputScope 屬性	Amazon SNS 支援 FifoThroughputScope 屬性，指定要套用至 FIFO 主題的輸送量配額和重複資料刪除行為。有效值為 Topic 或 MessageGroup 。	2025 年 1 月 21 日
AmazonSNSFullAccess 和 AmazonSNSReadOnlyAccess 受管政策更新	Amazon SNS 新增了 AmazonSNSFullAccess和 AmazonSNSReadOnlyAccess 受管政策的新許可，允許透過對 Amazon SNS 進行額外存取 AWS Management Console。	2024 年 9 月 24 日
Amazon SNS 與 整合 AWS End User Messaging SMS 以傳遞簡訊	Amazon SNS 支援新功能，例如簡訊資源管理、雙向簡訊、精細資源許可、國家/地區區塊規則，以及所有 AWS 簡訊的集中式帳單，而無需對 Amazon SNS 使用的組態或全域 AWS 簡訊網路進行任何變更。	2024 年 9 月 24 日

加拿大西部（卡加利）支援 FIFO 主題	Amazon SNS 在加拿大西部（卡加利）支援 FIFO 主題。	2024 年 3 月 28 日
五個新區域的 Amazon SNS SMS 支援	Amazon SNS 已將 SMS 支援新增至下列區域：亞太區域（海德拉巴）、亞太區域（墨爾本）、中東（阿拉伯聯合大公國）、歐洲（蘇黎世）和歐洲（西班牙）。	2024 年 2 月 8 日
Firebase Cloud Messaging (FCM) HTTP v1 支援	Amazon SNS 支援 FCM v1 登入資料。	2024 年 1 月 18 日
亞太區域 (雅加達) 支援 Amazon SNS SMS	Amazon SNS 在亞太區域 (雅加達) 支援簡訊傳遞。	2023 年 12 月 14 日
AWS CloudFormation 支援 DeliveryStatusLogging 設定 Amazon SNS 主題	AWS CloudFormation 支援可在建立或更新 Amazon SNS 主題 DeliveryStatusLogging 時進行設定。	2023 年 12 月 7 日
已新增新訊息篩選運算子	您現在篩選 Amazon SNS 訊息時，可以使用後綴比對，等於忽略大小寫和 OR 運算子。	2023 年 11 月 16 日
新增了訊息封存與重播功能的支援	主題擁有者可將訊息封存至主題長達 365 天。主題訂閱用戶可以將封存的訊息重播回訂閱的端點，以復原因下游應用程式失敗而遺失的訊息，或複寫現有應用程式的狀態。	2023 年 10 月 26 日
新增對標準佇列訂閱 FIFO 主題的支援	您可以訂閱 Amazon SQS FIFO 佇列或標準佇列到 Amazon SNS FIFO 主題。只有 Amazon SQS FIFO 佇列才能保證依序接收訊息且不會重複。	2023 年 9 月 14 日

新增了以色列 (特拉維夫) 的簡訊支援	以色列 (特拉維夫) 地區現在可支援 Amazon SNS 簡訊。	2023 年 8 月 28 日
支援 X-Ray 主動追蹤。	先前僅支援 Amazon SNS 標準主題，AWS X-Ray 現在會在使用者請求通過 FIFO 主題到 Amazon Data Firehose、Amazon SQS 和 HTTP/S AWS Lambda 端點訂閱時追蹤和分析使用者請求。	2023 年 5 月 31 日
增強 Content-Type 標頭支援	您可以在請求政策中設定 Content-Type 標頭，以指定通知的媒體類型。	2023 年 3 月 23 日
已新增主動追蹤支援	AWS X-Ray 會在使用者透過 Amazon SNS 標準主題前往 Amazon Data Firehose AWS Lambda、Amazon SQS 和 HTTP/S 端點訂閱時，追蹤和分析使用者請求。	2023 年 2 月 8 日
新加坡寄件者 ID 註冊	已新增在新加坡註冊寄件者 ID 的說明。	2023 年 1 月 10 日
以承載為基礎的訊息篩選	以承載為基礎的篩選可讓您根據訊息承載篩選訊息，並避免因處理不想要的資料，而產生額外成本。	2022 年 11 月 22 日
為 Amazon SNS 訊息簽署新增了 SHA256 雜湊演算法	支援新增使用 Amazon SNS 訊息簽署時的 SHA256 雜湊演算法。	2022 年 9 月 15 日

[已新增其他區域至簡訊](#)

Amazon SNS 在下列區域支援簡訊：非洲（開普敦）、亞太區域（大阪）、歐洲（米蘭）和 AWS GovCloud（美國東部）。

2022 年 9 月 9 日

[已新增訊息資料保護支援](#)

訊息資料保護會使用資料保護政策來稽核和封鎖在應用程式 AWS 或服務之間移動的敏感資訊，以保護發佈至 Amazon SNS 主題的資料。

2022 年 9 月 8 日

[免付費電話號碼的新註冊流程](#)

已新增使用免付費電話號碼 (TFN) 傳送 Amazon SNS 訊息給美國收件者的支援。

2022 年 8 月 1 日

[對屬性型存取控制 \(ABAC\) 的支援](#)

加入了適用於 API 動作 (包含 Publish 和 PublishBatch) 的屬性型存取控制 (ABAC) 的支援。ABAC 是一種授權策略，可根據可連接至 IAM 資源的標籤定義存取許可，例如 IAM 使用者和角色，以及連接至 AWS 資源，例如 Amazon SNS 主題，以簡化許可管理。

2022 年 1 月 10 日

[支援適用於推播通知的 Apple 權杖型身分驗證](#)

您可以透過提供將您識別為應用程式開發人員的資訊，授權 Amazon SNS 將推播通知傳送到您的 iOS 或 macOS 應用程式。

2021 年 10 月 28 日

[簡訊的新寄件者會放置在簡訊沙盒中](#)

簡訊沙盒功能有利於防止詐騙和濫用行為，以及協助保護您身為寄件者的評價。當 AWS 您的帳戶在 SMS 沙盒中時，您只能將 SMS 訊息傳送到已驗證的目的地電話號碼。

2021 年 6 月 1 日

[簡訊的新寄件者會放置在簡訊沙盒中](#)

簡訊沙盒功能有利於防止詐騙和濫用行為，以及協助保護您身為寄件者的評價。當 AWS 您的帳戶在 SMS 沙盒中時，您只能將 SMS 訊息傳送到已驗證的目的地電話號碼。

2021 年 6 月 1 日

[傳送簡訊給印度收件人的新屬性](#)

兩個新屬性實體 ID 和範本 ID 傳送簡訊給印度收件人。

2021 年 4 月 22 日

[訊息篩選運算子的更新](#)

一個新的運算子 `cidr` 可用於符合訊息來源 IP 地址和子網路。您現在也可以檢查是否存在屬性索引鍵，並使用帶有 `anything-but` 屬性字串匹配運算子。

2021 年 4 月 7 日

[終止支援美國目的地的 P2P 長代碼](#)

自 2021 年 6 月 1 日起，美國電信供應商不再支援使用人對人 (P2P) 長代碼來進行應用程式對人 (A2P) 通訊至美國目的地。相反地，您可以使用短碼、免付費電話號碼或新類型的來源號碼，稱為 10DLC。

2021 年 2 月 16 日

[支援 1 分鐘的 Amazon CloudWatch 指標](#)

Amazon SNS 的 1 分鐘 CloudWatch 指標現在適用於所有 AWS 區域。

2021 年 1 月 28 日

[支援 Amazon Data Firehose 端點](#)

您可以訂閱 Firehose 交付串流到 SNS 主題。這可讓您傳送通知到封存和分析端點，例如 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon Redshift 表格、Amazon RDS 表格、Amazon OpenSearch Service (OpenSearch Service) 等等。

2021 年 1 月 12 日

[提供來源編號](#)

您可以在傳送簡訊 (SMS) 時使用來源編號。

2020 年 10 月 23 日

[Amazon SNS FIFO 主題的支援](#)

若要整合需要幾乎即時資料一致性的分散式應用程式，您可以將 Amazon SNS 先進先出 (FIFO) 主題搭配 Amazon SQS FIFO 佇列使用。

2020 年 10 月 22 日

[適用於 Java 的 Amazon SNS 擴充用戶端程式庫可以使用](#)

您可以使用此程式庫發佈大型 Amazon SNS 訊息。

2020 年 8 月 25 日

[SSE 在中國區域中可供使用。](#)

Amazon SNS 的伺服器端加密 (SSE) 提供適用於中國區域。

2020 年 1 月 20 日

[使用 DLQ 擷取無法傳遞的郵件的支援](#)

您可以將 Amazon SQS 無效字母佇列與 Amazon SNS 訂閱搭配使用來擷取無法傳遞的訊息。

2019 年 11 月 14 日

[指定自訂 APN 標頭值的支援](#)

您可以指定自訂 APN 標頭值。

2019 年 10 月 18 日

[APN 的「apns-push-type」標頭欄位的支援](#)

您可以將 apns-push-type 標頭欄位用於透過 APN 傳送的行動通知。

2019 年 9 月 10 日

支援使用 進行主題故障診斷 AWS X-Ray	您可以使用 X-Ray 透過 SNS 主題傳送故障診斷訊息。	2019 年 7 月 24 日
使用「exists」運算子進行屬性索引鍵比對支援	您可以使用 exists 運算子來檢查傳入訊息是否具有其索引鍵列在篩選政策中的屬性。	2019 年 7 月 5 日
支援任何項目，但不會比對多個數值	除了多個字串外，Amazon SNS 會允許任何項目，但不會比對多個數值。	2019 年 7 月 5 日
Amazon SNS 版本備註亦支援 RSS 摘要。	遵循此頁面上的標題 (文件歷史記錄)，選擇 RSS。	2019 年 6 月 22 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。