aws

開發人員指南

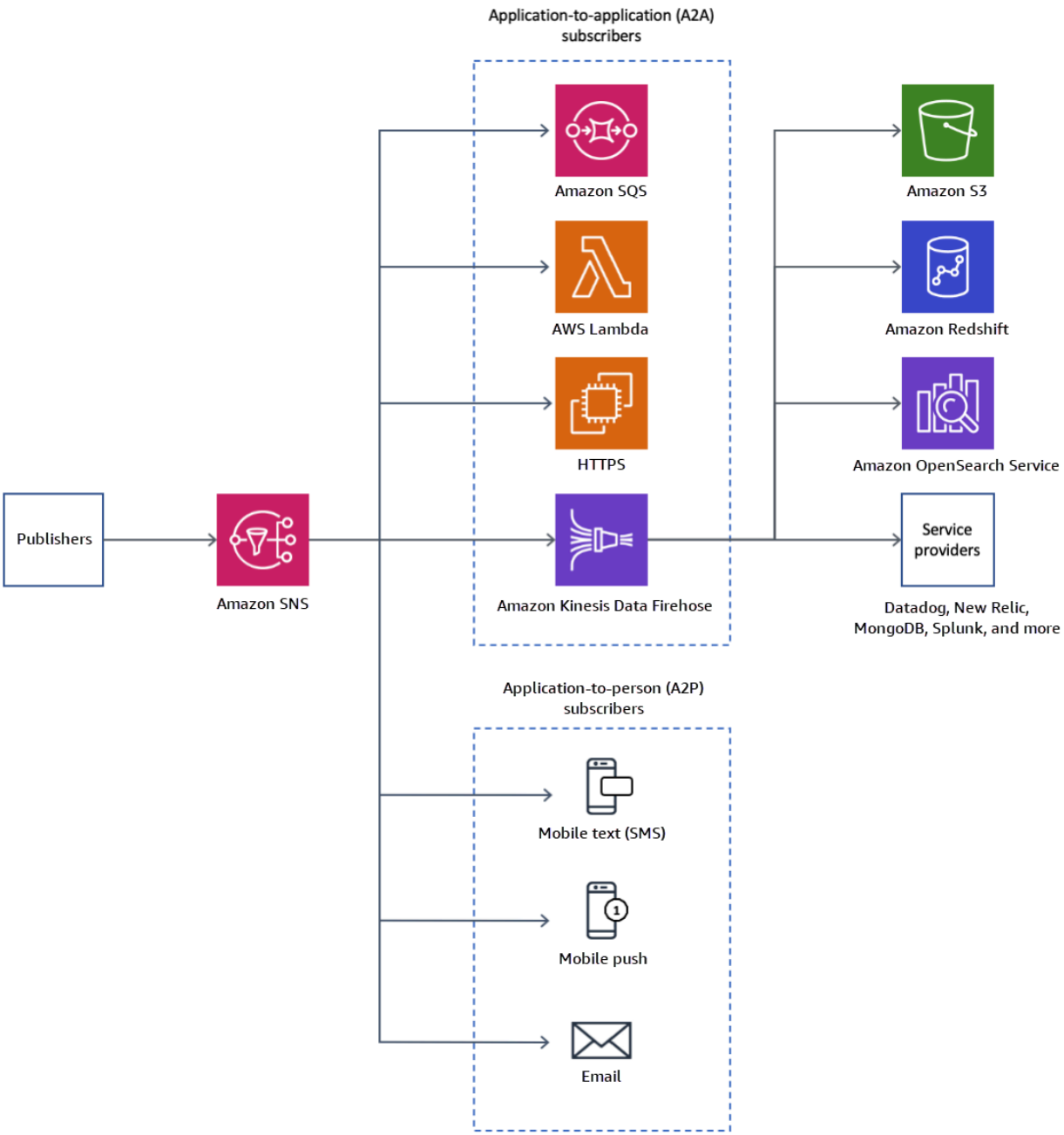
Amazon Simple Notification Service

Amazon Simple Notification Service: 開發人員指南

Table of Contents

什麼是 Amazon SNS？

Amazon Simple Notification Service (Amazon SNS) 是一種受管服務，會提供從發佈者到訂閱者的訊息傳遞 (也稱為生產者和消費者)。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。用戶端可以訂閱 SNS 主題，並使用支援的端點類型接收已發佈的訊息，例如 Amazon 資料 Firehose、Amazon SQS、HTTP AWS Lambda、電子郵件、行動推送通知和行動文字訊息 (SMS)。



主題

- 特性和功能
- 相關服務
- 存取 Amazon SNS

- Amazon SNS 的定價
- Amazon SNS 常見案例
- 搭配 AWS 開發套件使用 Amazon SNS

特性和功能

Amazon SNS 提供下列特性與功能：

- 一則pplication-to-application 訊息

  pplication-to-application 簡訊支援訂閱者，例如 Amazon 資料 Firehose 交付串流、Lambda 函數、Amazon SQS 佇列、HTTP/S 端點和 AWS 事件分叉管道。如需詳細資訊，請參閱 應用程式對應用程式 (A2A) 訊息。
- 一個pplication-to-person 通知

  pplication-to-person 通知會向訂閱者提供通知，例如行動應用程式、行動電話號碼和電子郵件地址。如需詳細資訊，請參閱 應用程式至人員 (A2P) 訊息。
- 標準與 FIFO 主題

  使用 FIFO 主題來確保嚴格的郵件順序、定義郵件群組，以及防止郵件重複。您可以同時使用 FIFO 和標準佇列訂閱 FIFO 主題。如需詳細資訊，請參閱 訊息排序與重複資料刪除 (FIFO 主題)。

  當訊息傳遞順序和可能的郵件重複不重要時，請使用標準主題。所有支援的傳遞通訊協定都可以訂閱標準主題。
- 訊息耐久性

  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和資料中心。
  - 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

  您可以透過多種方式將訊息存檔到 Amazon SNS，包括訂閱 Firehose 交付串流至 SNS 主題，這可讓您將通知傳送到分析端點，例如亞馬遜簡單儲存服務 (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 主題的訊息封存與重播功能。

- 訊息屬性

  訊息屬性可讓您提供有關訊息的任意中繼資料項目。the section called “訊息屬性”。

- 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

  伺服器端加密使用提供的加密金鑰，保護儲存在 Amazon SNS 主題中的訊息內容 AWS KMS。如需詳細資訊，請參閱 the section called “靜態加密”。

  您也可以在 Amazon SNS 和 Virtual Private Cloud (VPC) 之間建立私有連線。如需詳細資訊，請參閱 the section called “網際網路流量隱私權”。

相關服務

您可以搭配使用 Amazon SNS 與下列的服務：

- Amazon SQS 提供安全、耐用且可用的託管佇列，可讓您整合與分離分散式軟體系統和元件。Amazon SQS 與 Amazon SNS 的相關方式如下：
  - Amazon SNS 提供無效字母佇列，用於 Amazon SQS 無法傳遞的訊息。
  - 您可以訂閱 Amazon SQS 佇列到 Amazon SNS 主題。
  - 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。
- AWS Lambda 讓您建置可快速回應新資訊的應用程式。在高度可用的運算基礎設施上以 Lambda 函數執行您的應用程式代碼。如需詳細資訊，請參閱《AWS Lambda 開發人員指南》。您可以將 Lambda 函數訂閱到 SNS 主題。
- AWS Identity and Access Management (IAM) 可協助您安全地控制使用者對 AWS 資源的存取。使用 IAM 以控制誰可以使用您的 Amazon SNS 主題 (身分驗證)、他們可以使用那些主題和使用的方式 (授權)。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 CloudFormation可讓您建立 AWS 資源的模型和設定。建立描述所需 AWS 資源的範本，包括 Amazon SNS 主題和訂閱。 AWS CloudFormation 負責為您佈建和配置這些資源。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南。

存取 Amazon SNS

您可以使用 Amazon SNS 主控台、命令列工具或 AWS 開發套件來設定和管理 SNS 主題和訂閱。

- 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 提供各種語言的 SDK。如需詳細資訊，請參閱開發套件與工具組。

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。

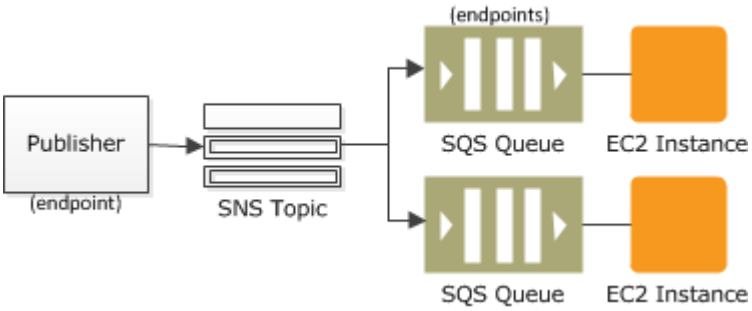
如需資訊，請參閱 Amazon SNS 定價。

Amazon SNS 常見案例

應用程式整合

散播案例是將發佈到 SNS 主題的訊息複寫並推送到多個端點時，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，您還可以為同一個 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：

- 扇出到 Firehose 交付流
- 擴散到 Lambda 函數
- 擴散到 Amazon SQS 佇列
- 擴散到 HTTP/S 端點
- 使用 Amazon SNS 和運 AWS 算、儲存、資料庫和聯網服務的事件驅動運算

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。例如，您可以在事件發生時立即收到通知，例如 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 行動裝置簡訊 (SMS) 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 行動推送通知。

搭配 AWS 開發套件使用 Amazon SNS

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。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
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS CLI	AWS CLI 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例
AWS Tools for PowerShell	PowerShell 程式碼範例工具
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby 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 開發套件的 Amazon SNS 的程式碼範例。

> **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。

Amazon SNS 事件來源和目的地

Amazon SNS 可以接收來自多個 AWS 來源事件驅動的通知和發送通知到應用程式至應用程式 (A2A) 及應用程式至人員 (A2P) 目的地。本節列出支援的事件來源和目的地，並提供更多資訊的連結。

主題

Amazon SNS 事件來源

此頁面列出 AWS 服務，可依其 AWS 產品類別將事件發佈到 Amazon SNS 主題。

> **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

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 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

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) 應用程式交付即時資料串流，以及將資料路由到目標 (包括 Amazon SNS) 的AWS服務和路由。 EventBridge 以前稱為「 CloudWatch 事件」。	接收 EventBridge 事件通知。有關更多信息，請參閱 Amazon EventBridge 用戶指南中的 Amazon EventBridge 目標。
AWS Step Functions - 可讓您結合 AWS Lambda 功能和其他 AWS 服務來建置關鍵業務應用程式。	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

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Billing and Cost Management - 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 • 透過 AWS 成本異常偵測來偵測不尋常的支出

商業應用程式服務

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 開發套件中的 Amazon Chime 開發套件事件通知。

運算服務

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 — 協助自動建立、管理和部署自訂、安全和 up-to-date伺服器映像，這些映像已預先安裝並預先設定軟體和設定，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 來構建網站或 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，

AWS 服務	搭配使用 Amazon SNS 的優點
	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

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

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 管理員指南中的使用 Amazon Connect 的 AWS 功能。
Amazon Pinpoint - 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 Amazon Pinpoint 使用者指南中的在 Amazon Pinpoint 中使用雙向簡訊。
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

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移轉至 AWS 雲端。	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 — 提供高效能、可調整大小且符合成本效益的記憶體內快取，同時消除與部署和管理分散式快取環境相關的複雜性。	在發生重大事件時接收通知。如需詳細資訊，請參閱 Amazon 適用 ElastiCache 於 Memcached 的使用者指南中的事件通知和 Amazon SNS。
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 - 讓 AWS 雲端中關聯式資料庫的設定、操作和擴展更加簡單。	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

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

AWS 服務	搭配使用 Amazon SNS 的優點
	閱《AWS CodeBuild使用指南》 CodeBuild中的「建置通知範例」。
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 儲存庫事件的通知。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 觸發程序。
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 部署或執行個體事件的通知。如需詳細資訊，請參閱《AWS CodeDeploy使用指南》中的「為 CodeDeploy 事件建立觸發器」。
Amazon CodeGuru — 從即時應用程式收集執行階段效能資料，並提供建議以協助您微調應用程式效能。	發生異常時接收通知。如需詳細資訊，請參閱 Amazon CodeGuru 使用者指南中的處理異常和建議報告。
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使用指南》 CodePipeline中的「管理核准動作」。
AWS CodeStar - 建立、管理和使用 AWS 上的軟體開發專案。	接收您使用的資源中所發生事件的通知。如需詳細資訊，請參閱開發人員工具主控台使用者指南中的設定 Amazon SNS 通知主題。

前端網頁與行動服務

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon Pinpoint - 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 Amazon Pinpoint 使用者指南中的在 Amazon Pinpoint 中使用雙向簡訊。

遊戲開發服務

AWS 服務	搭配使用 Amazon SNS 的優點
Amazon GameLift — 提供在雲端託管工作階段型多人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： • 如需配對通知的相關資訊，請參閱 Amazon GameLift FlexMatch 開發人員指南中的設定 FlexMatch 事件通知。 • 如需佇列通知，請參閱 Amazon GameLift 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

AWS 服務	搭配使用 Amazon SNS 的優點
AWS IoT Core - 提供雲端服務，可將您的 IoT 裝置連接至其他裝置和 AWS 雲端服務。	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 – 將 AWS 延伸到實體裝置，以便在本機上操作其產生的資料，同時繼續將雲端用於管理、分析和持久儲存。	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

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階段效能資料，並提供建議以協助您微調應用程式效能。	發生異常時接收通知。如需詳細資訊，請參閱 Amazon CodeGuru 使用者指南中的處理異常和建議報告。
Amazon DevOps Guru — 使用機器學習產生操作洞見，協助您改善操作應用程式的效能。	轉傳深入解析和確認。如需詳細資訊，請參閱透過 Amazon DevOps Guru 在AWS管理和控管部落格上，將以 ML 提供的營運見解提供給 PagerDuty 隨時待命團隊。
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 — 讓資料科學家和開發人員能夠建立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 開發人員指南中的建立串流標籤任務。

管理與治理服務

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Chatbot— 讓軟 DevOps 體開發團隊能夠使用 Amazon Chime 和 Slack 聊天室監控和回應雲端中的營運事件AWS。	將通知傳送至聊天室。如需詳細資訊，請參閱 AWS Chatbot 管理者指南中的設定 AWS Chatbot。
AWS CloudFormation - 可讓您以可預期和重複的方式建立及佈建 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 服務	搭配使用 Amazon SNS 的優點
AWS CloudTrail - 提供您 AWS 帳戶 活動的事件記錄。	將新的日誌檔 CloudTrail 發佈到 Amazon S3 儲存貯體時接收通知。如需詳細資訊，請參閱AWS CloudTrail使用者指南 CloudTrail中的設定 Amazon SNS 通知。
Amazon CloudWatch — 即時監控您的AWS資源和執行AWS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 使用 CloudWatch 者指南中的使用 Amazon CloudWatch 警示。
AWS Config - 提供您 AWS 帳戶 中 AWS 資源組態的詳細檢視。	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 主題通知和項目變更通知範例組態。
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 - 協助您在 AWS 和內部部署環境從軟體廠商集中管理軟體授權。	接收 License Manager 通知和警示。如需詳細資訊，請參閱 L icense Manager 使用指南中的 License Manager 中的設定和在AWS管理與控管部落格上建立AWS License Manager通知 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 AWS 上的基礎設施。	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

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

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Application Discovery Service - 可透過收集關於您內部部署伺服器的使用情況和組態資料，協助您規劃遷移到 AWS 雲端。	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移轉至 AWS 雲端。	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— 使用實體儲存裝置在 Amazon S3 和現場資料儲存位置之間以 faster-than-internet 快速傳輸大量資料。	接收 Snowball 工作的通知。如需詳細資訊，請參閱下列內容： • AWS Snowball 使用者指南中的 Snowball 通知 • 步驟 5：在 AWS Snowball Edge 開發人員指南中選擇通知偏好設定 • 步驟 5：在 AWSSnowcone 使用者指南中選擇通知偏好設定

聯網與內容交付服務

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自己的 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 整合中的 API Gateway REST API。
Amazon CloudFront — 加快靜態和動態 Web 內容的分發速度，例如 .html，.css，.php，圖像和媒體文件。	根據指定指 CloudFront 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您的內部網路連結至 AWS Direct Connect 據點。	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，請參閱傳統負載平衡器使用者指南中的為負載平衡器建立 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)。
Amazon Virtual Private Cloud (Amazon VPC) - 可讓您將 AWS 資源啟動到您定義的虛擬網路。	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

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Directory Service - 提供多種方式，使 Microsoft Active Directory (AD) 得以與其他 AWS 服務搭配使用。	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環境中未預期且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 AWS Security Hub 控制項或標準的通知。如需詳細資訊，請參閱透過 Amazon SNS 訂閱 AWS Security Hub 公告。

無伺服器服務

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) 應用程式交付即時資料串流，以及將資料路由到目標 (包括 Amazon SNS) 的AWS服務和路由。 EventBridge 以前稱為「 CloudWatch 事件」。	接收 EventBridge 事件通知。有關更多信息，請參閱 Amazon EventBridge 用戶指南中的 Amazon EventBridge 目標。

AWS 服務	搭配使用 Amazon SNS 的優點
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

AWS 服務	搭配使用 Amazon SNS 的優點
AWS Backup - 可讓您在雲端和內部部署環境內集中管理並自動化各 AWS 服務間的資料備份。	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。
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，當物件未複寫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)和監控複寫指標和 Amazon S3 事件通知進度。
AWS Snowball— 使用實體儲存裝置在 Amazon S3 和現場資料儲存位置之間以 faster-than-internet 快速傳輸大量資料。	接收 Snowball 工作的通知。如需詳細資訊，請參閱下列內容： • AWS Snowball 使用者指南中的 Snowball 通知

AWS 服務	搭配使用 Amazon SNS 的優點
	• 步驟 5：在 AWS Snowball Edge 開發人員指南中選擇通知偏好設定 • 步驟 5：在 AWSSnowcone 使用者指南中選擇通知偏好設定

其他事件來源

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 區域與 Ser AWS vice Quotas AWS 服務 整合的 Amazon VPC 端點、Amazon EC2 執行個體類型、Amazon 執行個體類型、Amazon SageMaker 敏捷工作室執行個體類型、Amazon RDS 資料庫引擎版本以及 Amazon MSK Apache 卡夫卡版本。AWS 服務如需詳細資訊，請參閱 AWS 新聞部落格中透過 Amazon SNS 訂閱 AWS 每日功能更新。
AWSIP 地址範圍	透過 Amazon SNS 主題接收 AWS IP 範圍變更通知。如需詳細資訊，請參閱 Amazon Web Services 一般參考 中的 AWS IP 地址範圍通知，以及 AWS 新聞部落格中的透過 Amazon SNS 訂閱 AWS 公用 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？
- AWS 運算部落格上的使用 Amazon SNS 的事件驅動運算和 AWS 運算、儲存、資料庫和聯網服務
- AWS 運算部落格上的搭配使用豐富事件驅動的架構 AWS 事件分叉管道

Amazon SNS 事件目的地

本頁列出可以接收事件資訊的所有目的地，並依 application-to-application (A2A) 訊息和 application-to-person (A2P) 通知分組。

> **Note**
>
> Amazon SNS 在 2020 年 10 月推出 FIFO 主題。目前，大多數 AWS 服務支援只接收來自 SNS 標準主題的事件。Amazon SQS 支援從 SNS 標準和 FIFO 主題接收事件。

A2A 目的地

事件目的地	搭配使用 Amazon SNS 的優點
Amazon 數據 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，您可以將事件傳遞到 Amazon 簡單儲存服務 (Amazon S3)、Amazon Redshift 和亞馬遜 OpenSearch 服務 (服OpenSearch 務) 等AWS目的地，或傳遞至第三方目的地，例如資料多格、新遺物、MongoDB 和 Splunk。如需詳細資訊，請參閱 扇出到 Firehose 交付流。
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 擴散到 Lambda 函數。
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 擴散到 Amazon SQS 佇列。
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 擴散至 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 擴散到 HTTP/S 端點。

A2P 目的地

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 行動裝置簡訊 (SMS)。
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 電子郵件通知。
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 行動推送通知。
AWS Chatbot	將活動傳遞至 Amazon Chime 聲聊天室或 Slack 頻道。如需詳細資訊，請參閱中的下列頁面 AWS Chatbot 管理員指南： • 搭配 Amazon Chime 設定 AWS Chatbot • 搭配 Slack 設定 AWS Chatbot • 搭配使用 AWS Chatbot 與其他 AWS 服務
PagerDuty	將營運洞察提供給待命團隊。如需詳細資訊，請參閱透過 Amazon DevOps Guru 在AWS管理 PagerDuty 與控管部落格上，將以 ML 提供的營運見解提供給隨時待命團隊。

Note

您可以同時提供原生 AWS 事件和自訂事件到聊天應用程式：

- 原生AWS事件 - 您可以使用 AWS Chatbot 以傳送原生 AWS 事件，透過 Amazon SNS 主題，傳送到 Amazon Chime 和 Slack。支援的原生AWS事件集包括來自AWS Billing and Cost Management、AWS Health、AWS CloudFormation CloudWatch、Amazon 等的事件。如需詳細資訊，請參閱 AWS Chatbot 管理員指南中的使用 AWS Chatbot 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？

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。

主題

步驟 1：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. 您可以使用您 AWS 帳戶 來檢視您的活動和使用情況報告，以及管理驗證和存取權。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

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

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊時 AWS 帳戶，系統會建立一個 AWS 帳戶 root 使用者。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。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。

建立具有管理權限的使用者

註冊後，請保護 AWS 帳戶 root 使用者的安全 AWS 帳戶 AWS IAM Identity Center、啟用並建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶 root 用戶

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console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)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

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 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 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 身分中心使用者登入的說明，請參閱使用AWS 登入 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 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

後續步驟

您現在已準備好使用 Amazon SNS，若要開始使用，請建立主題、建立主題的訂閱、將訊息發佈到主題，以及刪除訂閱和主題。

Amazon SNS 入門

本節顯示如何使用 Amazon SNS 主控台管理主題、訂閱和訊息，有助於您更熟悉 Amazon SNS。

主題

必要條件

開始之前，請完成 設定 Amazon SNS 存取權限 中的步驟。

步驟 1：建立主題

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，選擇 Create topic (建立主題)。
4. 根據預設，主控台會建立 FIFO 主題。選擇 Standard (標準)。
5. 在「詳細資料」區段中，輸入主題的「名稱」，例如*MyTopic*。
6. 捲動到表單結尾，然後選擇 Create topic (建立主題)。

   主控台會開啟新主題的 Details (詳細資料) 頁面。

步驟 2：建立主題的訂閱

1. 在左導覽窗格中，選擇 Subscriptions (訂閱)。
2. 在 Subscriptions (訂閱) 頁面，選擇 Create subscription (建立訂閱)。

3. 在 Create subscription (建立訂閱) 頁面中，選擇 Topic ARN (ARN 主題) 欄位，以檢視 AWS 帳戶中的主題清單。
4. 選擇您在之前步驟所建立的主題。
5. 對於 Protocol (通訊協定)，選擇 Email (電子郵件)。
6. 針對 Endpoint (端點)，請輸入可用於接收通知的電子郵件地址。
7. 選擇建立訂閱。

   主控台會開啟新訂閱的 Details (詳細資訊) 頁面。
8. 檢查您的電子郵件收件匣並在 AWS 通知的電子郵件中選擇 Confirm subscription (確認訂閱)。寄件者 ID 通常是「no-reply@sns.amazonaws.com」。
9. 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。

步驟 3：將訊息發佈到主題

1. 在左側導覽窗格中，選擇 Topics (主題)。
2. 在 Topics (主題) 頁面上，選擇您稍早建立的主題，然後選擇 Publish message (發佈訊息)。

   主控台會開啟 Publish message to topic (將訊息發佈至主題) 頁面。
3. (選用) 在 Message details (訊息詳細資訊) 區段中，輸入 Subject (主旨)，例如：

   ```
   Hello from Amazon SNS!
   ```
4. 在 Message body (訊息內文) 區段中，選擇 Identical payload for all delivery protocols (所有傳送通訊協定的相同酬載量)，然後輸入訊息內文，例如：

   ```
   Publishing a message to an SNS topic.
   ```
5. 選擇 Publish message (發佈訊息)。

   訊息會發佈到主題，主控台會開啟主題的 Details (詳細資訊) 頁面。
6. 檢查您的電子郵件收件匣，並確認您收到來自 Amazon SNS 的電子郵件，其中包含已發佈訊息。

步驟 4：刪除訂閱和主題

1. 在導覽面板上，選擇 Subscriptions (訂閱)。

2. 在 Subscriptions (訂閱) 頁面上，選擇 confirmed (確認) 訂閱，再選擇 Delete (刪除)。

> **Note**
>
> 您無法刪除待定的確認。48 小時後，Amazon SNS 會自動將其刪除。

3. 在 Delete subscription (刪除訂閱) 對話方塊中，選擇 Delete (刪除)。

   訂閱已刪除。
4. 在導覽面板上，選擇 Topics (主題)。
5. 在 Topics (主題) 頁面上，選擇主題，然後選擇 Delete (刪除)。

> **Important**
>
> 當您刪除一個主題，同時也刪除所有到該主題的訂閱。

6. 在 [刪除主題] *MyTopic*對話方塊中，輸入，`delete me`然後選擇 [刪除]。

   已刪除主題。

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。

設定 Amazon SNS

使用 Amazon SNS 主控台以建立和設定 Amazon SNS 主題和訂閱。如需有關 Amazon SNS 的詳細資訊，請參閱 什麼是 Amazon SNS？。

主題

- 建立 Amazon SNS 主題
- 訂閱 Amazon SNS 主題
- 刪除 Amazon SNS 主題和訂閱
- Amazon SNS 主題標記

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分組 (例如 AWS Lambda Amazon SQS、HTTP/S 或電子郵件地址)。

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。本頁顯示如何使用 AWS Management Console AWS SDK for Java、和 AWS SDK for .NET 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。主題名稱可供其他 Amazon Web Services 存取，包括 CloudWatch 日誌。主題名稱不適用於私有或敏感資料。

主題

- 若要使用建立主題 AWS Management Console
- 若要使用 AWS SDK 建立主題

若要使用建立主題 AWS Management Console

1. 登入 Amazon SNS 主控台。
2. 執行以下任意一項：
   - 如果您 AWS 帳戶 之前沒有建立任何主題，請閱讀首頁上 Amazon SNS 的說明。
   - 如果主題已在您 AWS 帳戶 之前的下方建立，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 主題的重複訊息刪除。
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 靜態加密。
   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。

- 預設會選取適用於 Amazon SNS 的 AWS 受管理 KMS (預設) 別名/AWS/SNS。

> **Note**
>
> 請謹記以下幾點：
>
> - 當您第一次使用 AWS Management Console 為某個主題指定適用於 Amazon SNS 的 AWS 受管 KMS 時， AWS KMS 會建立適用於 Amazon SNS 的 AWS 受管 KMS。
> - 或者，當您第一次對已啟用 SSE 的主題使用`Publish`動作時， AWS KMS 會建立適用於 Amazon SNS 的 AWS 受管 KMS。

- 若要從您的 AWS 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。

> **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 AWS 帳戶或其他 AWS 帳戶使用自訂 KMS ARN，請在 KMS 金鑰欄位中輸入該自訂 KMS ARN。

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. (選擇性) 若要設定 Amazon SNS 記錄訊息交付的方式 CloudWatch，請展開交付狀態記錄區段。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。

    隨即建立主題並顯示***MyTopic***頁面。

    主題的 [名稱]、[ARN] (選用) [顯示名稱] 和 [主題擁有者的 AWS 帳號 ID] 會顯示在 [詳細資料] 區段中。
11. 將主題 ARN 複製到剪貼簿，例如：

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

若要使用 AWS SDK 建立主題

要使用 AWS SDK，您必須使用憑據對其進行配置。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。

下列程式碼範例會示範如何使用`CreateTopic`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考CreateTopic中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考CreateTopic中的。

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，請參閱命 AWS 令列介面使用者指南中的 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CreateTopic中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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



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

 return topicArn, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CreateTopic中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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CreateTopic中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CreateTopic中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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()
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateTopic中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreateTopic中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件CreateTopic中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreateTopic中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱 AWS SDK CreateTopic中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK CreateTopic中的 SAP ABAP API 參考資料。

訂閱 Amazon SNS 主題

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**
>
> HTTP (S) 端點、電子郵件地址和其他 AWS 帳戶 的 AWS 資源需要確認訂閱才能接收訊息。

讓端點訂閱 Amazon SNS 主題

1. 登入 Amazon SNS 主控台。
2. 在左導覽窗格中，選擇 Subscriptions (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. 對於ARN 主題，選擇主題的 Amazon Resource Name (ARN)。此值是 AWS ARN，它是在您建立 Amazon SNS 主題 (例如 `arn:aws:sns:us-east-2:123456789012:your_topic`) 時產生。

   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 數據 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：對於訂閱角色 ARN，請指定您為寫入 Firehose 交付串流而建立的 IAM 角色的 ARN。如需詳細資訊，請參閱 訂閱 Amazon SNS 主題的 Firehose 交付串流的先決條件。

e. (選擇性) 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 (DLQ)。

i. 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

刪除 Amazon SNS 主題和訂閱

刪除主題時，會以非同步方式刪除其關聯的訂閱。雖然客戶仍然可以存取這些訂閱，但訂閱不再與主題相關聯，即使您使用相同名稱重新建立主題也是如此。

如果訂閱用戶嘗試將訊息發佈到已刪除的主題，發布者將會收到錯誤訊息，指出該主題不存在。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。

主題

- 若要使用刪除 Amazon SNS 主題或訂閱 AWS Management Console

- 使用 AWS SDK 刪除訂閱和主題

若要使用刪除 Amazon SNS 主題或訂閱 AWS Management Console

若要使用刪除主題 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。

若要使用刪除訂閱 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

要使用 AWS SDK，您必須使用憑據對其進行配置。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。

下列程式碼範例會示範如何使用`DeleteTopic`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考DeleteTopic中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考DeleteTopic中的。

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DeleteTopic中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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



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

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteTopic中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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DeleteTopic中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteTopic中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteTopic中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考DeleteTopic中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件DeleteTopic中的 Python (博托 3) API 參考。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK DeleteTopic中的 SAP ABAP API 參考資料。

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主題相關的成本、在您的 AWS Identity and Access Management (IAM) 政策中提供更高的安全性，並且讓您輕鬆搜尋或篩選數以千計的主題。標記功能可讓您使用 AWS Resource Groups 來管理 Amazon SNS。如需 Resource Groups 的詳細資訊，請參閱《AWS Resource Groups 使用者指南》。

主題

- 為分配成本加上標籤
- 為存取控制加上標籤
- 為資源搜尋和過濾加上標籤
- 設定 Amazon SNS 主題標籤

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整理您的 AWS 帳單，以反映您自己的成本結構。若要這麼做，請註冊取得 AWS 帳戶帳單，以納入標籤索引鍵和鍵值。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	索引鍵	值
主題 1	成本中心	43289
	應用	訂單處理
主題 2	成本中心	43289
	應用	付款處理
主題 3	成本中心	76585

資源	索引鍵	值
	應用	存檔

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 支援以標籤控制資源的存取。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 帳戶可能會有成千上萬個 Amazon SNS 主題 (請參閱 Amazon SNS 配額以取得詳細資訊）。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

本頁說明如何使用 AWS Management Console、 AWS SDK 和 AWS CLI 來設定 Amazon SNS 主題的標籤。

> **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。

主題

- 使用列出、新增和移除 Amazon SNS 主題的標籤 AWS Management Console
- 使用 AWS SDK 新增標籤到主題
- 以 Amazon SNS API 動作管理標籤
- 支援 ABAC 的 API 動作

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. 選擇儲存變更。

使用 AWS SDK 新增標籤到主題

要使用 AWS SDK，您必須使用憑據對其進行配置。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。

下列程式碼範例會示範如何使用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TagResource中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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TagResource中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK TagResource中的 Kotlin API 參考。

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。有關 ABAC 的更多詳細信息，請參閱 ABAC 的用途是什麼？ AWS 在 IAM 使用者指南中。

- AddPermission
- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes

- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource
- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

訊息排序與重複資料刪除 (FIFO 主題)

您可以使用 Amazon SNS FIFO (先進先出) 主題和 Amazon SQS FIFO 佇列，以提供嚴格的訊息順序和重複訊息刪除功能。這些服務的 FIFO 功能共同運作，做為全受管服務，整合需要幾乎即時資料一致性的分散式應用程式。訂閱 Amazon SQS 標準佇列至 Amazon SNS FIFO 主題，可提供最佳的訂購方式和至少一次交付。

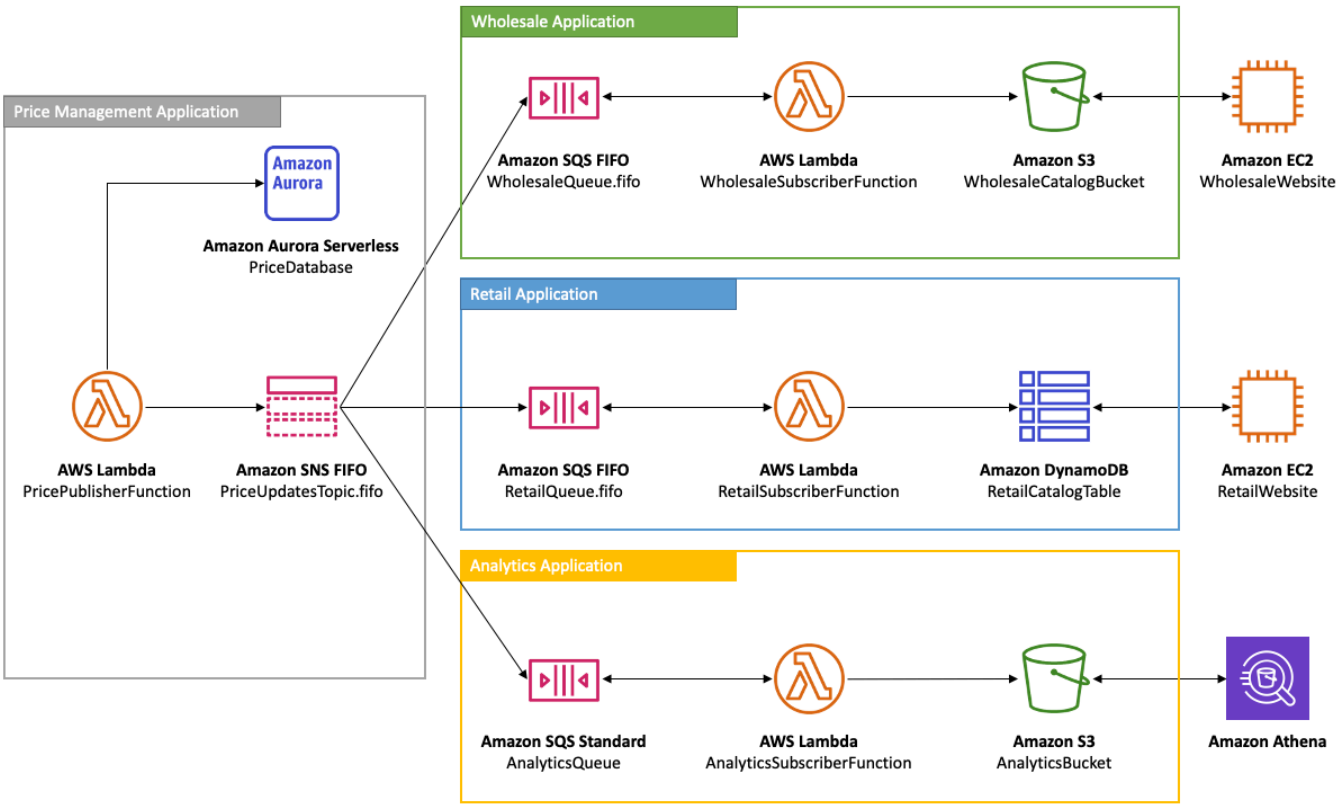
主題

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。

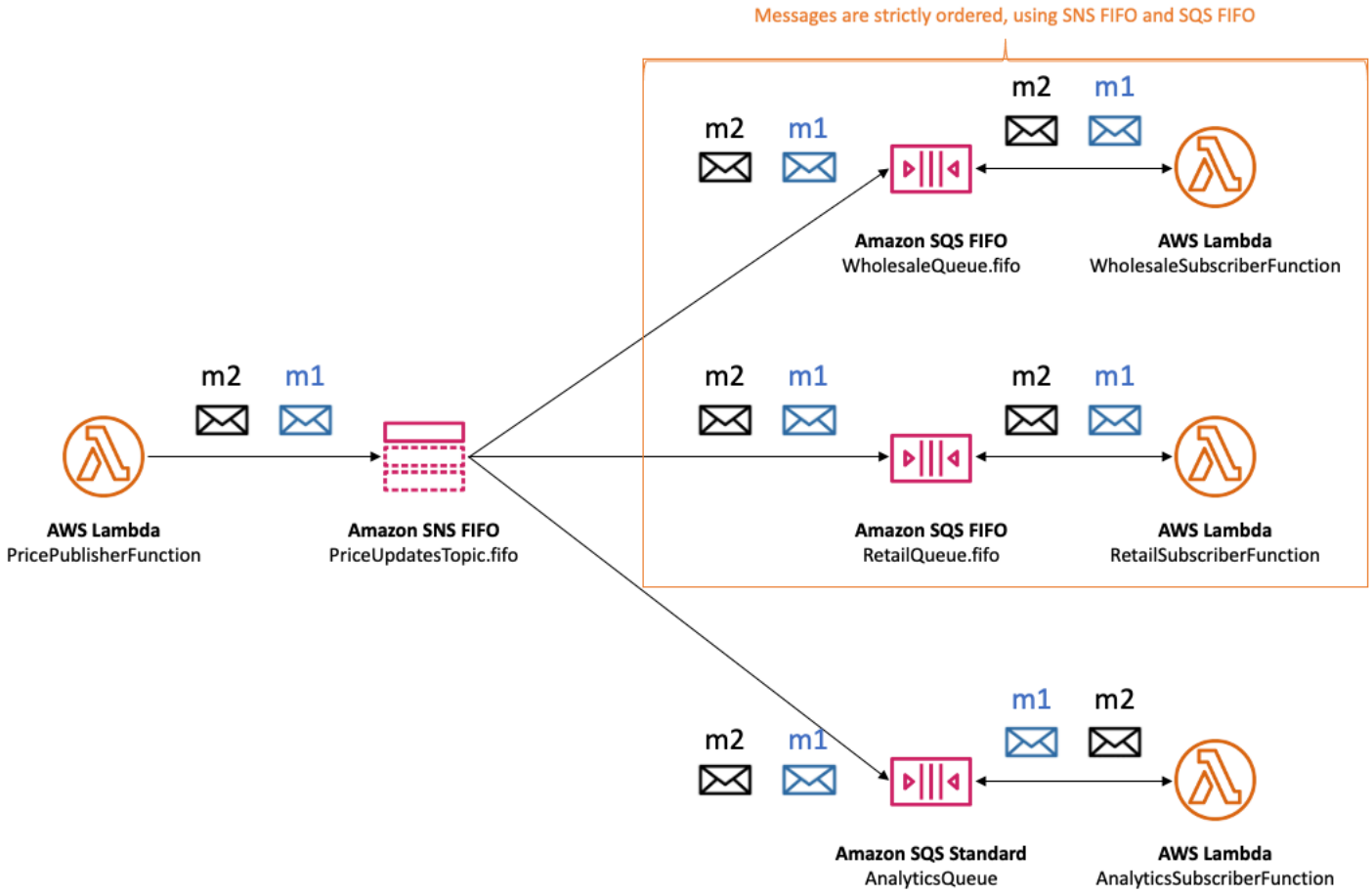
- 一種可彙總價格更新並將其存放到 Amazon S3 儲存貯體的分析應用程式，讓 Amazon Athena 能夠查詢儲存貯體以用於商業智慧 (BI) 目的。為了獲得價格變化通知，分析應用程式將其 Amazon SQS 標準佇列訂閱到價格管理應用程式的 Amazon SNS FIFO 主題。與其他應用程式不同，分析應用程式不需要嚴格排序價格更新。



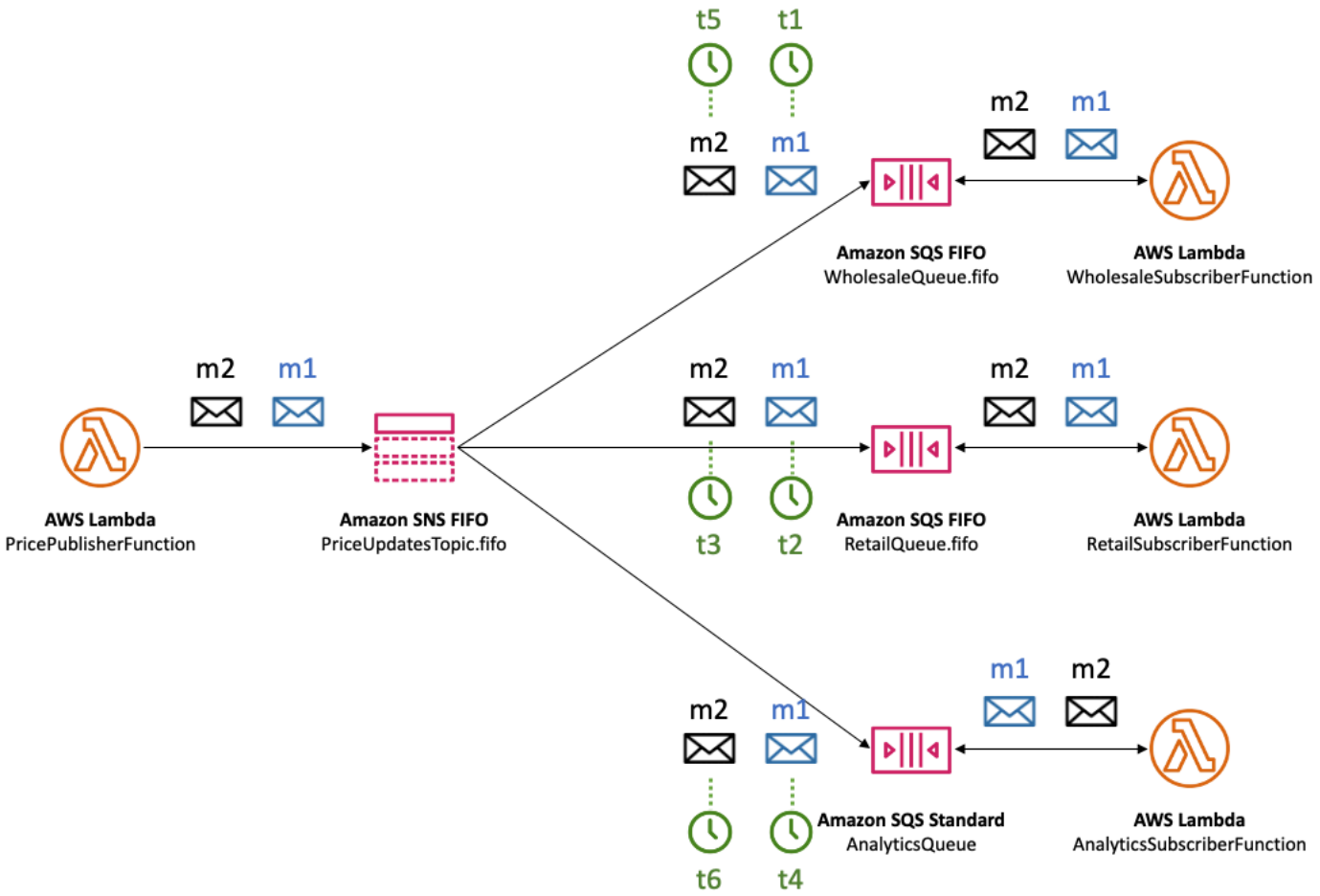
為了讓批發和零售應用程序以正確的順序接收價格更新，價格管理應用程式必須使用嚴格訂購的訊息分配系統。使用 Amazon SNS FIFO 主題和 Amazon SQS FIFO 佇列可以按順序處理訊息，而不會重複。如需更多詳細資訊，請參閱 FIFO 主題的訊息排序詳細資料。如需實作此使用案例的代碼片段，請參閱 FIFO 主題的代碼範例。

FIFO 主題的訊息排序詳細資料

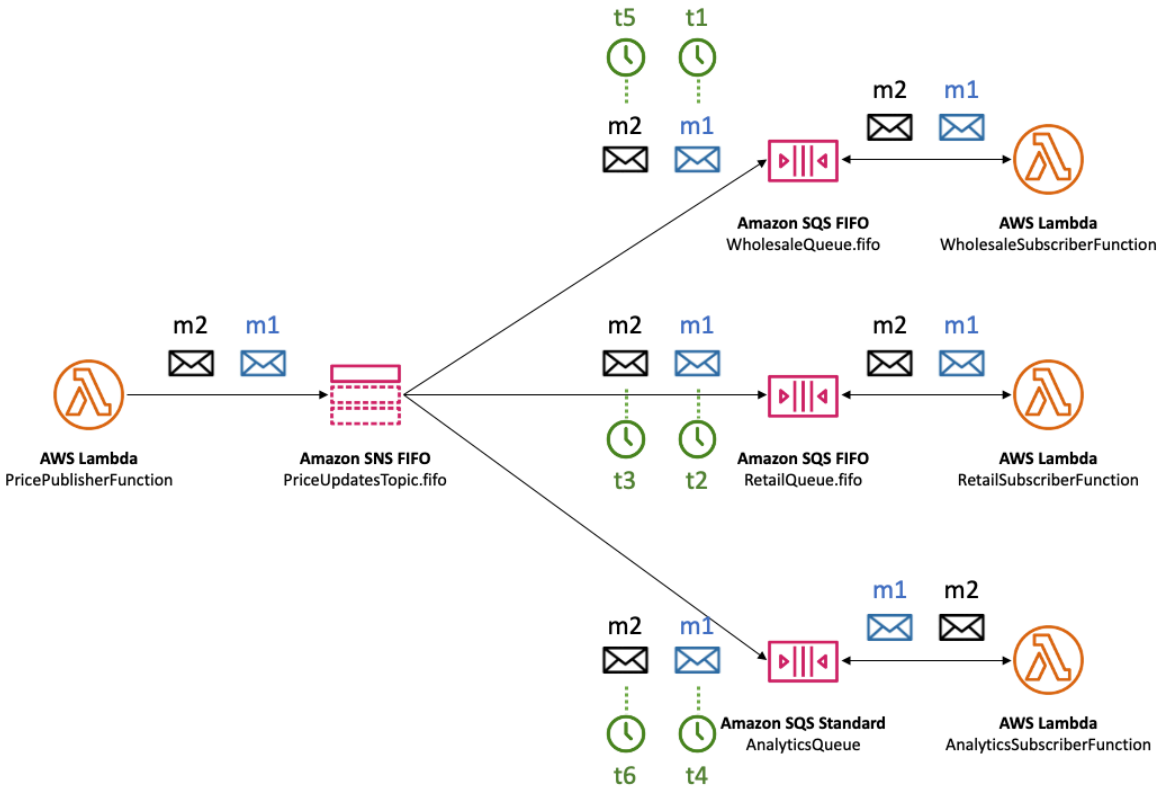
Amazon SNS FIFO 主題一律會依訊息發布到主題的確切順序，將訊息傳送到訂閱的 Amazon SQS 佇列，並僅傳送一次。訂閱 Amazon SQS FIFO 佇列後，佇列的取用者會以訊息傳送到佇列接收訊息的確切順序，且不會重複。不過，如果訂閱了 Amazon SQS 標準佇列，佇列的取用者可能會接收非按順序排列的訊息，而且會接收不止一次。如此可進一步使發布者和訂閱用戶解耦，讓訂閱用戶在訊息消耗和成本最佳化方面具有更大的彈性，如下圖所示，根據 FIFO 主題範例使用案例。



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。

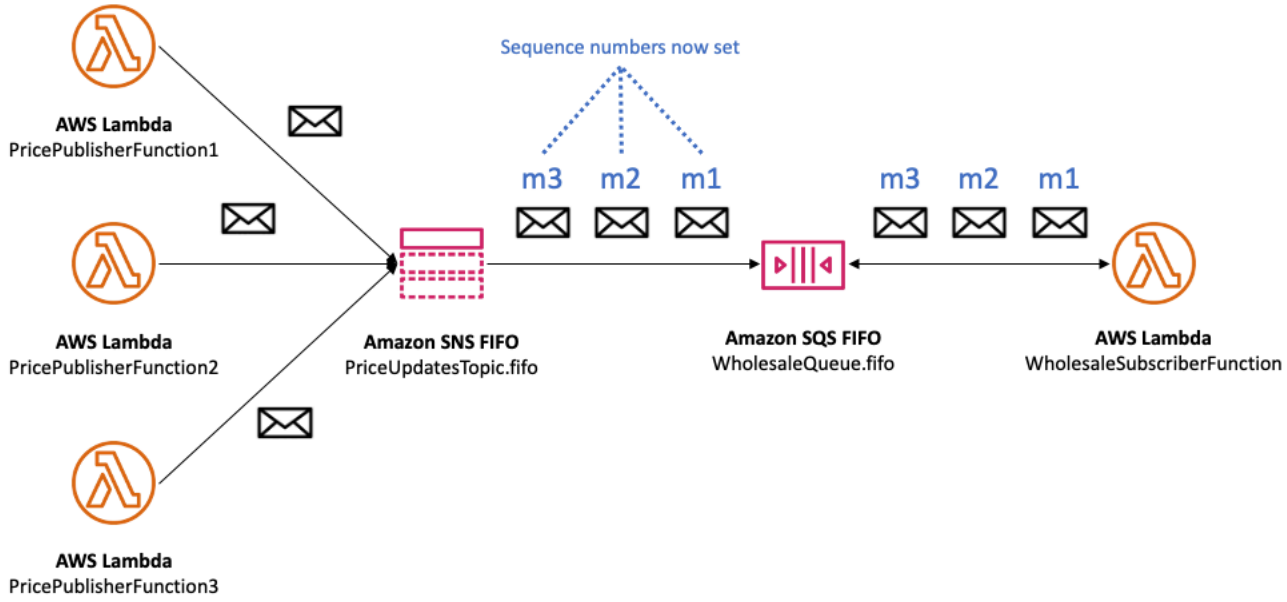


如果 Amazon SQS 佇列訂閱用戶無法連線，則可能會不同步。例如，假設批發應用程式佇列擁有者錯誤地改變了Amazon SQS 佇列政策，以防止 Amazon SNS 服務主體將訊息傳遞到佇列。在這種情況下，傳遞到批發佇列的價格更新失敗，而零售和分析佇列的傳遞成功，導致訂閱用戶不同步。當批發應用程式佇列擁有者更正其佇列政策時，Amazon SNS 會繼續將訊息傳送到訂閱佇列。任何在目標佇列設定不正確時發布至主題的訊息都會被捨棄，除非對應的訂閱具有已設定的無效字母佇列。



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。如需更多詳細資訊，請參閱 FIFO 主題的訊息群組。

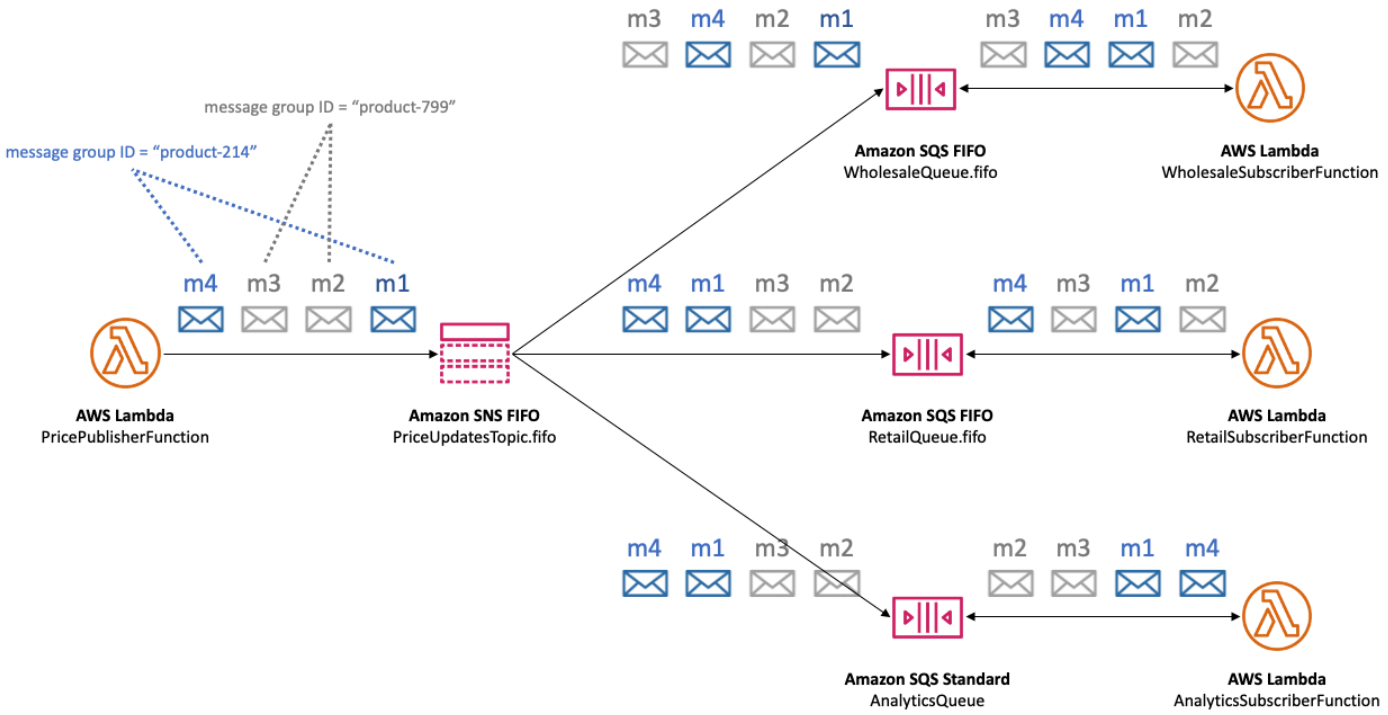
FIFO 主題的訊息群組

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 主題的訊息傳遞 及 FIFO 主題的訊息篩選。

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，對於具有產品-214 訊息群組 ID 的產品，m1 訊息一律會在 m4 訊息前處理。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 為產品-799 的產品，只要工作流程使用 Amazon SNS FIFO 和 Amazon SQS FIFO，就會在訊息 m3 之前處理訊息 m2。但是，使用 Amazon SQS 標準佇列時，訊息順序將不再保證，訊息群組也不存在。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 主題經過最佳化，可在訊息群組 ID 之間統一分配訊息，而不論群組數目為何。AWS 建議您使用大量不同的訊息群組 ID 以最佳化效能。

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 Amazon SNS FIFO 主題的訊息散佈到 AWS Lambda 函數，需要額外步驟。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。如需更多詳細資訊，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。如需更多詳細資訊，請參閱 FIFO 主題的重複訊息刪除。

FIFO 主題的重複訊息刪除

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 at-most-once 交付，因為可以根據您的訂閱篩選政策篩選出訊息。
- 沒有網路中斷會阻止確認訊息傳遞。

> **Note**
>
> 重複訊息刪除功能適用於整個 Amazon SNS FIFO 主題，而不適用於個別訊息群組。

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 的訊息成功發布至 Amazon SNS FIFO 主題，則在五分鐘的重複資料刪除間隔內以相同重複資料刪除 ID 發布的任何訊息都會被接受，但不會傳遞。Amazon SNS FIFO 主題會繼續追蹤重複資料刪除 ID，即使郵件傳遞至訂閱的端點也一樣。

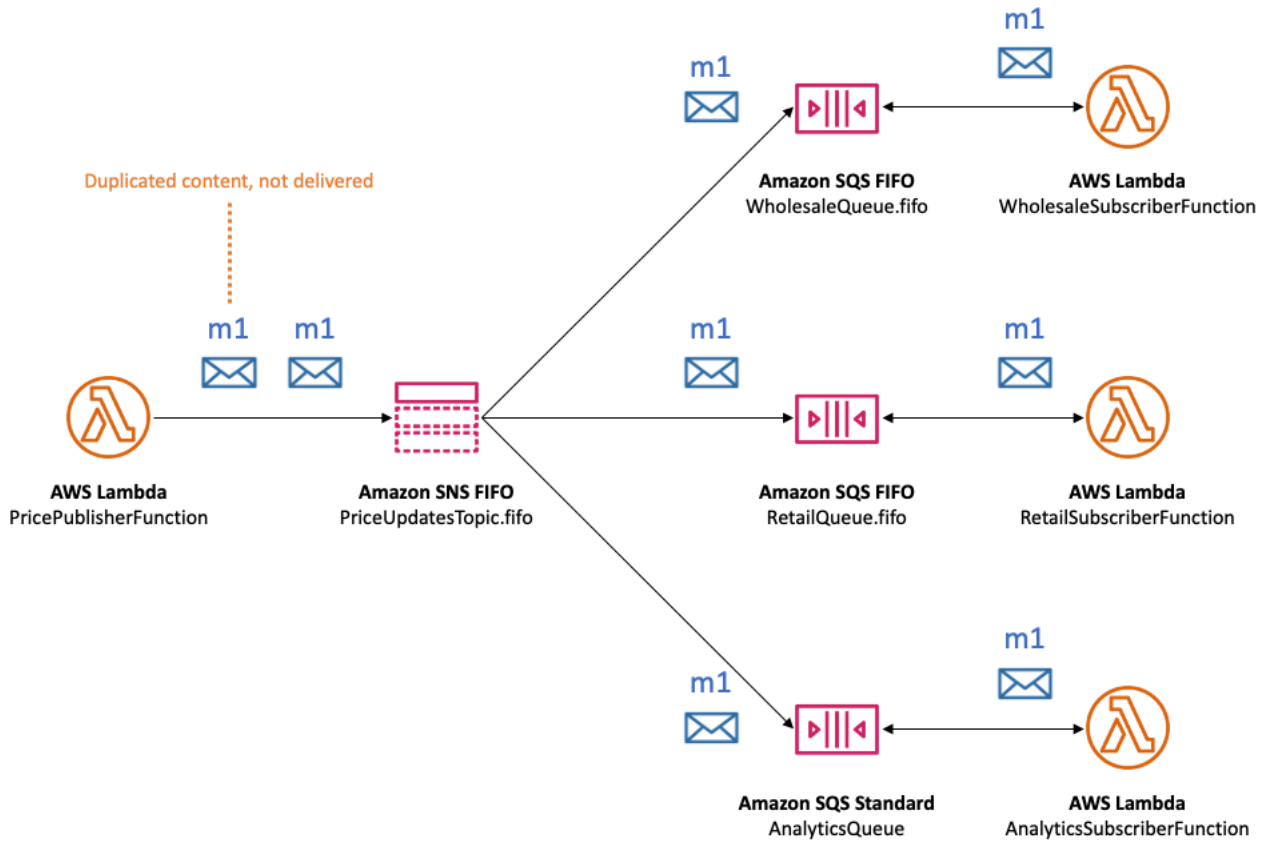
如果確保每封已發布訊息的郵件本文都是唯一的，您可以針對 Amazon SNS FIFO 主題和訂閱的 Amazon SQS FIFO 佇列啟用內容型重複資料刪除功能。Amazon SNS 使用訊息主體產生唯一的雜湊值，用作每封訊息的重複資料刪除 ID，因此您在傳送每封訊息時不需要設定重複資料刪除 ID。

> **Note**
>
> 訊息屬性不包含在雜湊計算中。

當 Amazon SNS FIFO 主題啟用以內容為基礎的重複資料刪除功能，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 會覆寫產生的以內容為基礎的重複資料刪除 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，並在產品 ID 和產品價格旁邊，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 主題還支援含 AWS KMS 金鑰的訊息伺服器端加密 (SSE)，以及透過 VPC 端點使用的訊息隱私權。 AWS PrivateLink如需更多詳細資訊，請參閱 FIFO 主題的訊息安全。

FIFO 主題的訊息安全

您可以選擇讓 Amazon SNS 和 Amazon SQS 加密訊息使用 AWS Key Management Service (AWS KMS) 將客戶主金鑰 (CMK)傳送到 FIFO 主題和佇列。您可以建立加密的 FIFO 主題和佇列，或選擇加密現有的 FIFO 主題和佇列。Amazon SNS 和 Amazon SQS 只會加密訊息的主體。它們不會加密訊息屬性、資源中繼資料或資源指標。

> **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 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，以增強應用程式的整體安全性。當您使用 AWS PrivateLink，無須設定網際網路閘道、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 安全性部落格上的網際網路流量隱私權與透過 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。如需更多詳細資訊，請參閱 FIFO 主題的訊息耐久性。

FIFO 主題的訊息耐久性

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。

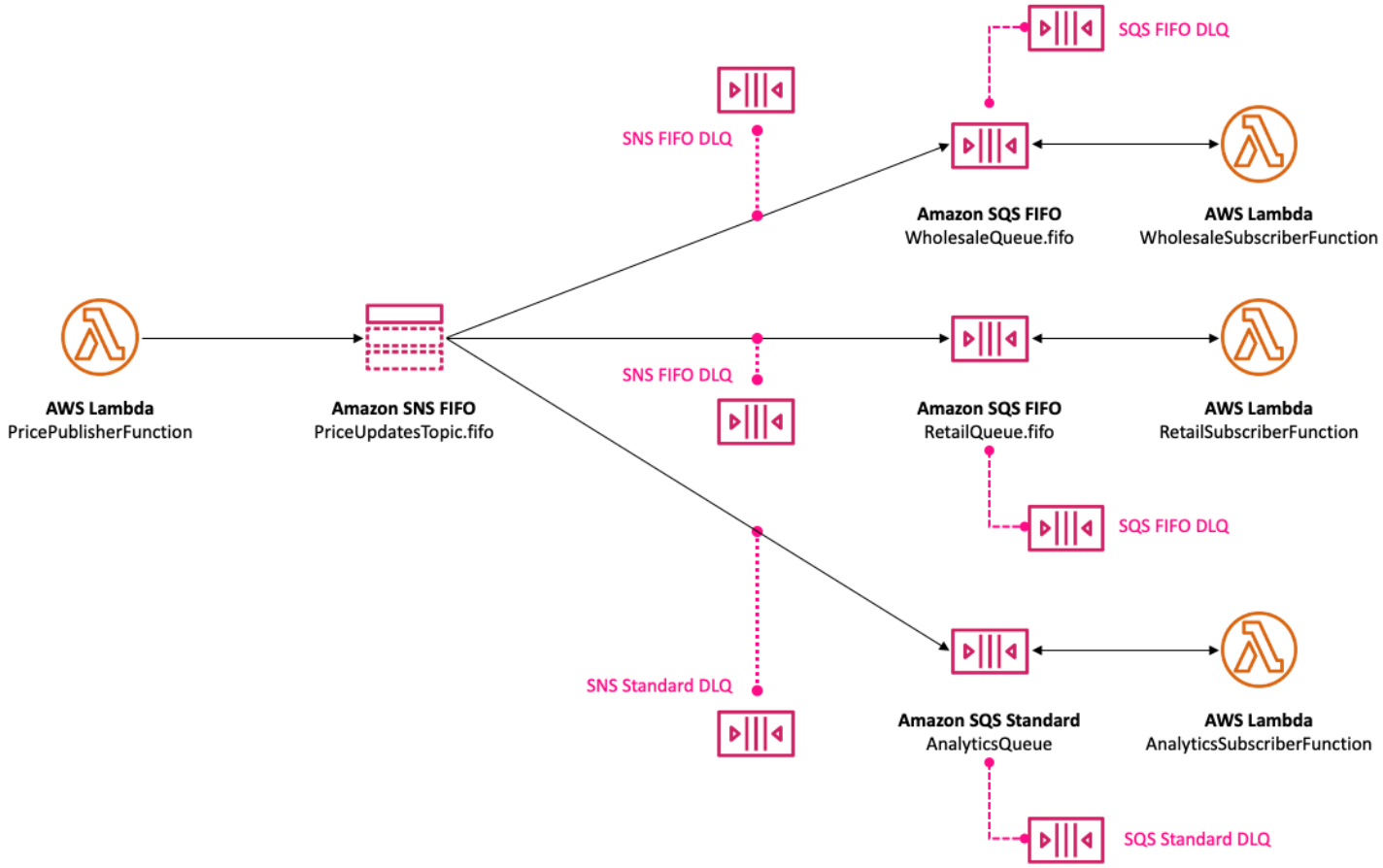
  Amazon SNS 不會重試傳遞因用戶端錯誤而失敗的訊息。
- 伺服器端錯誤可能會發生在下列情況：
  - Amazon SQS 服務無法使用。
  - Amazon SQS 無法處理來自 Amazon SNS 服務的有效請求。

  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。如需更多詳細資訊，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 (DLQ) 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 主題的訊息封存與重播功能。

FIFO 主題的訊息封存與重播功能

主題

什麼是訊息封存與重播功能？

Amazon SNS 訊息封存與重播功能是一種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，後續可用來：

- 復原可能因下游應用程式失敗而遺失的訊息。
- 透過訂閱新端點並選取要複寫的來源時間戳記，將現有應用程式的狀態複寫至新的應用程式。

您可以使用訊息封存與重播功能搭配 AWS API、SDK、AWS CloudFormation 和 AWS Management Console。

> **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。

訊息封存與重播功能包含兩個主要元件：

1. 訊息封存 - 主題擁有者針對主題啟用封存與重播功能，並設定訊息保留期間 (最長 365 天)。主題擁有者還可以使用 Amazon CloudWatch 指標來監控封存的訊息。如需更多資訊，請參閱 FIFO 主題擁有者的訊息封存。
2. 訊息重播 - 主題訂閱用戶對其訂閱的端點重播來自主題的一組訊息。如需更多資訊，請參閱 FIFO 主題訂閱用戶的訊息重播。

FIFO 主題擁有者的訊息封存

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。

主題

- 使用 AWS Management Console 建立訊息封存政策
- 使用 API 建立訊息封存政策
- 使用 SDK 建立訊息封存政策
- 使用 AWS CloudFormation 建立訊息封存政策
- 授權存取加密的封存
- 使用 Amazon CloudWatch 監控訊息封存指標

使用 AWS Management Console 建立訊息封存政策

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 SDK，您必須使用您的認證進行設定。如需詳細資訊，請參閱《AWS SDK 和工具參考指南》中的共用 config 和 credentials 檔案。

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 AWS CloudFormation 建立訊息封存政策

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

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，以便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. 為 Amazon SNS 啟用 AWS KMS。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。
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 主題訂閱用戶的訊息重播

Amazon SNS 重播可讓主題訂閱用戶從主題資料存放區擷取封存的訊息，並將訊息重新傳遞 (或重播) 至訂閱的端點。一旦建立訂閱，就可以重播訊息。重播的訊息與原始副本擁有相同的內容、MessageId 和 Timestamp，而且同樣包含屬性 Replayed，有助於識別這是重播的訊息。若只要重播特定訊息，您可以新增篩選政策至訂閱。如需更多篩選訊息的資訊，請參閱 篩選重播的訊息。

主題

- 使用 AWS Management Console 建立訊息重播政策
- 使用 API 新增重播政策至訂閱
- 使用 SDK 新增重播政策至訂閱
- 篩選重播的訊息
- 使用 Amazon CloudWatch 監控訊息重播指標

使用 AWS Management Console 建立訊息重播政策

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 訂閱 Amazon SNS 主題。
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 SDK，您必須使用您的認證進行設定。如需詳細資訊，請參閱《AWS SDK 和工具參考指南》中的共用 config 和 credentials 檔案。

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);
```

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。如需更多詳細資訊，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 主題的代碼範例

您可以使用下列代碼範例將使用 Amazon SNS FIFO 主題的汽車零件價格管理範例使用案例與 Amazon SQS FIFO 佇列或標準佇列整合。

主題

- 使用 AWS 開發套件
- 使用 AWS CloudFormation

使用 AWS 開發套件

使用 AWS 開發套件，您可以透過設定將其 `FifoTopic` 屬性設定為 **`true`** 建立一個 Amazon SNS FIFO 主題。您可以通過將其 `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。如需更多詳細資訊，請參閱 FIFO 主題的訊息篩選。

Java

適用於 Java 2.x 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。

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");
```

```
            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 Reference》中的下列主題。
  - CreateTopic
  - 發布
  - Subscribe

Python

適用於 Python (Boto3) 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 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),
                },
            )
            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《適用於 Python (Boto3) 的 AWS SDK API 參考資料》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_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                          "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
```

```
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

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

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

- 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 the section called “FIFO 主題使用案例” 所示，每個消費者應用程式的入口點是 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。

如需更多詳細資訊，請參閱 AWS Lambda 開發人員指南中的將 AWS Lambda 與 Amazon SQS 搭配使用。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，並將一系列的 AWS 資源一起建立和設定為單一單位。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"
          }
        ]
      }
    }
  }
}
```

如需有關使用 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 訊息發布

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。

主題

- 將訊息發布到使用 AWS Management Console的 Amazon SNS 主題
- 使用 AWS SDK 將訊息發布至主題
- 使用 Amazon SNS 和 Amazon S3 發布大型訊息
- Amazon SNS 訊息屬性
- Amazon SNS 訊息批次處理

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 使用平台特定承載進行發佈。

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

要使用 AWS SDK，您必須使用憑據對其進行配置。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。

下列程式碼範例會示範如何使用`Publish`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

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

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

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

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

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

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

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

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

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 AWS SDK for .NET API 參考中的發佈。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ 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!" \
    --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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

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



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考中的發佈。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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-xxxxxxxxxxxx'
  // }
```

```
  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, deduplicationId, choices;

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

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

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

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
```

```
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

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

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

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的發佈。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的發佈。

PowerShell

適用的工具 PowerShell

範例 1：此範例顯示以單一 MessageAttribute 宣告的內嵌方式發佈郵件。

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

示例 2：此示例顯示發布具有多個事先 MessageAttributes 聲明的消息。

```
$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 指令程式參考中發佈。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的發佈。

Rust

適用於 Rust 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

- 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。

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 擴充用戶端程式庫與標準主題和 FIFO 主題相容。

主題

適用於 Java 的擴充用戶端程式庫

主題

必要條件

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：

- 一個 AWS 開發套件。

  此頁面上的範例使用 AWS Java SDK。若要安裝和設定 SDK，請參閱 AWS SDK for Java 開發人員指南中的設定適用於 Java 的 AWS SDK。
- AWS 帳戶 具有適當憑據的。

若要建立帳戶 AWS 帳戶，請瀏覽至AWS 首頁，然後選擇 [建立 AWS 帳戶]。遵循指示。

如需認證的相關資訊，請參閱開發AWS SDK for Java 人員指南中的設定 AWS 認證和開發區域。

- 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 擴充用戶端程式庫使用承載卸載 Java 通用程式庫進 AWS 行訊息儲存和擷取。您可以配置以下 Amazon S3 訊息儲存選項：

- 自訂訊息大小臨界值 - 含有酬載資料和超過此大小的屬性的訊息會自動儲存在 Amazon S3 中。
- `alwaysThroughS3` 旗標 - 將此值設定為 `true` 強制將所有訊息承載儲存在 Amazon S3 中。例如：

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

- 自訂 KMS 金鑰 - Amazon S3 儲存貯體中用於伺服器端加密的金鑰。
- 儲存貯體名稱 - 用於儲存訊息酬載的 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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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,
                                sns ExtendedClientConfiguration);

                // 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。

無法使用承載卸載 Java 通用程式庫的端點仍然 AWS 可以發佈包含存放在 Amazon S3 中的承載的訊息。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-xxxxx-xxxxx-xxxxxx"
```

```
  }
]
```

適用於 Python 的擴充用戶端程式庫

主題

- 必要條件
- 設定訊息儲存
- 範例：將訊息發布到 Amazon SNS，其中儲存在 Amazon S3 中的有效酬載

必要條件

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：

- 一個 AWS 開發套件。

  這個頁面上的範例使用 AWS Python 開發套件 3。若要安裝和設定 SDK，請參閱 AWS SDK for Python 說明文件。
- AWS 帳戶 具有適當憑據的。

  若要建立帳戶 AWS 帳戶，請瀏覽至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` – 用來儲存大量訊息的 Amazon S3 儲存貯體名稱。
- `message_size_threshold` – 將郵件儲存在大型訊息儲存貯體中的臨界值。不可小於 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 – 用於將物件存放在 Amazon S3 中的 Boto3 Amazon S3 `resource` 物件。如果您想要控制 Amazon S3 資源 (例如自訂 Amazon S3 組態或憑證)，請使用此選項。如果先前未在第一次使用時設定，預設值為 `boto3.resource("s3")`。

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。
- 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。
- 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
import sns_extended_client
from json import loads

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

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

# Create an helper to fetch and print message SQS queue and S3
def fetch_and_print_from_sqs(sqs, queue_url):
    """Handy Helper to fetch and print message from SQS queue and S3"""
    message = sqs.receive_message(
        QueueUrl=queue_url, MessageAttributeNames=["All"], MaxNumberOfMessages=1
```

```
    ).get("Messages")[0]
    message_body = message.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is: {}\n".format(get_msg_from_s3(message_body)))

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

# Create and subscribe an SQS queue to the SNS client
sqs = boto3.client("sqs")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
demo_queue_arn = sqs.get_queue_attributes(QueueUrl=demo_queue_url,
AttributeNames=["QueueArn"])["Attributes"].get("QueueArn")
# Set the RawMessageDelivery subscription attribute to TRUE
sns_extended_client.subscribe(TopicArn=demo_topic_arn, Protocol="sqs",
Endpoint=demo_queue_arn, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

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

輸出

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

```
    }
]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the message_size_threshold limit
```

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。

如需使用 AWS Management Console 或 AWS SDK for Java 搭配屬性傳送訊息的相關資訊，請參閱將訊息發布到使用 AWS Management Console的 Amazon SNS 主題 教學。

> **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。如需更多詳細資訊，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。資料類型會區分大小寫，長度上限為 256 個位元組。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。若為字串資料類型，值的屬性在內容上的限制與訊息內文相同。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。

  此資料類型不支援 AWS Lambda 訂閱。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。

此資料類型不支援 AWS Lambda 訂閱。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	`AWS.SNS.MOBILE.ADM.TTL`
APNs [1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`
	`AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK.TTL`
	`AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP.TTL`
	`AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL`
	`AWS.SNS.MOBILE.APNS.COLLAPSE_ID`
	`AWS.SNS.MOBILE.APNS.PRIORITY`
	`AWS.SNS.MOBILE.APNS.PUSH_TYPE`
	`AWS.SNS.MOBILE.APNS.TOPIC`
	`AWS.SNS.MOBILE.APNS.TTL`
百度	`AWS.SNS.MOBILE.BAIDU.DeployStatus`
	`AWS.SNS.MOBILE.BAIDU.MessageKey`

推送通知服務	預留的訊息屬性
	AWS.SNS.MOBILE.BAIDU.MessageType
	AWS.SNS.MOBILE.BAIDU.TTL
FCM	AWS.SNS.MOBILE.FCM.TTL
	AWS.SNS.MOBILE.GCM.TTL
macOS	AWS.SNS.MOBILE.MACOS_SANDBOX.TTL
	AWS.SNS.MOBILE.MACOS.TTL
MPNS	AWS.SNS.MOBILE.MPNS.NotificationClass
	AWS.SNS.MOBILE.MPNS.TTL
	AWS.SNS.MOBILE.MPNS.Type
WNS	AWS.SNS.MOBILE.WNS.CachePolicy
	AWS.SNS.MOBILE.WNS.Group
	AWS.SNS.MOBILE.WNS.Match
	AWS.SNS.MOBILE.WNS.SuppressPopup
	AWS.SNS.MOBILE.WNS.Tag
	AWS.SNS.MOBILE.WNS.TTL
	AWS.SNS.MOBILE.WNS.Type

[1] 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**
>
> 單一 PublishBatch API 請求中傳送的訊息彙總大小總計不得超過 262,144 位元組 (256 KB)。
> 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 com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
```

```
private static final int MAX_BATCH_SIZE = 10;

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

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

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

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

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

將 10 則訊息的批次發佈至 FIFO 主題

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

// Code
private static final int MAX_BATCH_SIZE = 10;

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

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

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

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

```
            System.out.println("SequenceNumber for successful message: " +
 publishBatchResultEntry.getSequenceNumber());
        });

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

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

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。如有任何訊息屬性或內文屬性相符，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)。

主題

- Amazon SNS 訂閱篩選政策範圍
- Amazon SNS 訂閱篩選政策
- 套用訂閱篩選政策
- 移除訂閱篩選政策

Amazon SNS 訂閱篩選政策範圍

`FilterPolicyScope` 訂閱屬性可讓您設定下列其中一個值來選擇篩選範圍：

- `MessageAttributes` - 篩選政策會套用至訊息屬性。此為預設值。
- `MessageBody` - 篩選政策會套用至訊息內文。

> **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

篩選政策範例

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"
   ]
}
```

篩選政策限制條件

為 Amazon SNS 訂閱建立篩選器政策時，請務必瞭解政策中金鑰的計算方式。要記住的關鍵方面是：

1. 父系金鑰 — 父項金鑰是篩選原則中的最上層索引鍵。這些是您為其指定值或約束的索引鍵。
2. 屬性名稱 — 父項索引鍵會被視為篩選策略中的屬性名稱。您為這些索引鍵指定的值或條件約束會套用至訊息承載中的對應屬性。
3. 有效值 — 為父系索引鍵指定的值必須是字串、字串陣列或數字。如果該值是物件 (例如 JSON 物件)，則該值不會計為篩選原則中的有效金鑰。

讓我們考慮以下示例過濾器策略：

```
{
  "state": ["SUCCESS"],
  "severity": [{ "exists": true }],
  "message": [{ "exists": true }],
```

```
  "finding": {
    "standard_control": [{ "exists": true }],
    "region": [{ "exists": true }],
    "account": [{ "exists": true }]
  }
}
```

在此範例中，下列機碼會計為篩選器原則的一部分：

- state
- severity
- message
- standard_control
- region
- account

索引鍵尋找不會計算在內，因為它包含 JSON 物件做為其值，而不是字串、字串陣列或數字。

另一個範例是：

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

在這種情況下，只有鍵key_d和key_f被計為篩選策略的一部分，因為它們已經為它們分配了一個字符串或字符串數組的值。父系索引鍵key_akey_b、和不會計算key_c在內，因為它們包含巢狀 JSON 物件做為其值。

主題

- 常見政策限制條件

- 以屬性為基礎的篩選政策限制條件
- 以承載為基礎的篩選政策限制條件

常見政策限制條件

- 字串比對 — 對於篩選策略中的字串比對，比較會區分大小寫。
- 數字匹配 — 對於數字匹配，值的範圍可以從 -10^9 到 10^9（-1 億到 10 億），小數點後有五位數的精確度。
- 過濾器策略複雜性 — 對於篩選策略的複雜性，值的總組合不得超過 150。若要計算總組合，請將篩選原則中每個陣列中的值數相乘。

請考慮下列範例原則：

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

在本政策中：

- 第一個數組具有 3 個值
- 第二個數組具有 1 個值
- 第三個數組有 2 個值

組合總計的計算如下：

- 三 x 1 x 2 = 6

篩選原則語法

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，您必須將篩選器政策的 JSON 作為有效的 UTF-8 字串傳遞。

篩選原則限制

- 篩選器原則的大小上限為 256 KB。
- 根據預設，每個主題最多可以有 200 個篩選策略，每個 AWS 帳戶最多可以有 10,000 個篩選策略。
- 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**
    >
    > 不建議在陣列中傳遞物件，由於其巢狀結果不支援屬性型篩選，因此可能會產生非預期結果。為巢狀政策使用以承載為基礎的篩選條件。

  - `Number`
- Amazon SNS 會忽略具有 `Binary` 資料類型的訊息屬性。
- 篩選政策最多可有 5 個屬性名稱。

以承載為基礎的篩選政策限制條件

Amazon SNS 接受以有效負載為基礎的篩選的巢狀篩選政策。若要計算篩選原則中值的總組合，請將每個巢狀陣列中的值數相乘。

請考慮下列範例原則：

```
{
"key_a": {
```

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

在本政策中：

- 第一個數組在三級嵌套密鑰中具有四個值。
- 第二個在兩層嵌套關鍵字中具有三個值。

組合總計的計算如下：

- 四 x 三 x 二 = 72

政策限制

篩選原則最多可以有五個上層金鑰 (最上層金鑰)。對於巢狀政策，只有上層索引鍵會計入五個金鑰限制。

數值範圍

對於篩選原則中的數值比對，值的範圍可以從 -10^9 到 10^9 (-1 億到 10 億)，小數點後有 5 位數的精確度。

切換到基於有效載入的篩選

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`。

AND/OR 邏輯

您可以使用包含 AND/OR 邏輯的操作來比對訊息屬性或訊息內文屬性。

主題

- AND 邏輯
- OR 邏輯

- OR 運算子

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" ],
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      {
        "metricType": [ "MetricType" ] ,
        "$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

您可以使用 exists 運算子在篩選原則中配對具有或不具有指定屬性的傳入訊息。exists 比對僅適用於葉節點。它不適用於中繼節點。

- 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

您可以比對數值與訊息屬性值或訊息內文屬性值，以篩選訊息。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。

主題

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

您可以比對字串值與訊息屬性值或訊息內文屬性值，以篩選訊息。在 JSON 政策中，字串值是以雙引號括住。您可以使用以下字串操作來比對訊息屬性或訊息內文。

主題

- 完全符合
- 除外相符
- 使用具 anything-but 運算子的字首
- 電子quals-ignore-case 匹配
- IP 地址比對

- 前綴相符
- 後綴相符

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。

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"}
```

它也符合以下訊息內文：

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

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

但不符合以下訊息屬性：

```
"customer_interests": {"Type": "String", "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。

舉例下列政策屬性：

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

它符合以下任一訊息屬性：

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

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

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"]
}
```

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"]
}
```

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"
}
```

電子quals-ignore-case 匹配

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"
}
```

套用訂閱篩選政策

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 開發套件。您也可以使用 AWS CloudFormation.

> **Important**
>
> AWS IAM 和 Amazon SNS 等服務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) 套用篩選原則，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。

如果您想要從以屬性為基礎的 (預設) 切換為以承載為基礎的郵件篩選，也可以使用命令。set-subscription-attributes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 開發套件

下列程式碼範例會示範如何使用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。有關如何安裝此類的說明，請參閱 GitHub 網站上的示例。

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SetSubscriptionAttributes中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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SetSubscriptionAttributes中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件SetSubscriptionAttributes中的 Python (博托 3) API 參考。

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

若要使用套用篩選原則 AWS CloudFormation，請使用 JSON 或 YAML 範本建立 AWS CloudFormation 堆疊。如需詳細資訊，請參閱《AWS CloudFormation 使用指南》中的`AWS::SNS::Subscription`資源`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. 在中 myHttpEndpoint，輸入要訂閱您主題的 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)。

移除訂閱篩選政策

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。

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。

AWS CLI

若要使用 AWS CLI 移除篩選政策，請使用 `set-subscription-attributes` 命令，並為 `--attribute-value` 引數提供空的 JSON 內文：

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

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。

訊息資料保護

主題

什麼是訊息資料保護？

訊息資料保護使用資料保護政策來稽核、遮罩、修訂或封鎖在應用程式或 AWS 服務之間移動的敏感資料，藉此保護發佈到 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發送到 Amazon CloudWatch，Amazon S3 或 Amazon 數據 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，則封鎖應用程式和 AWS 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。

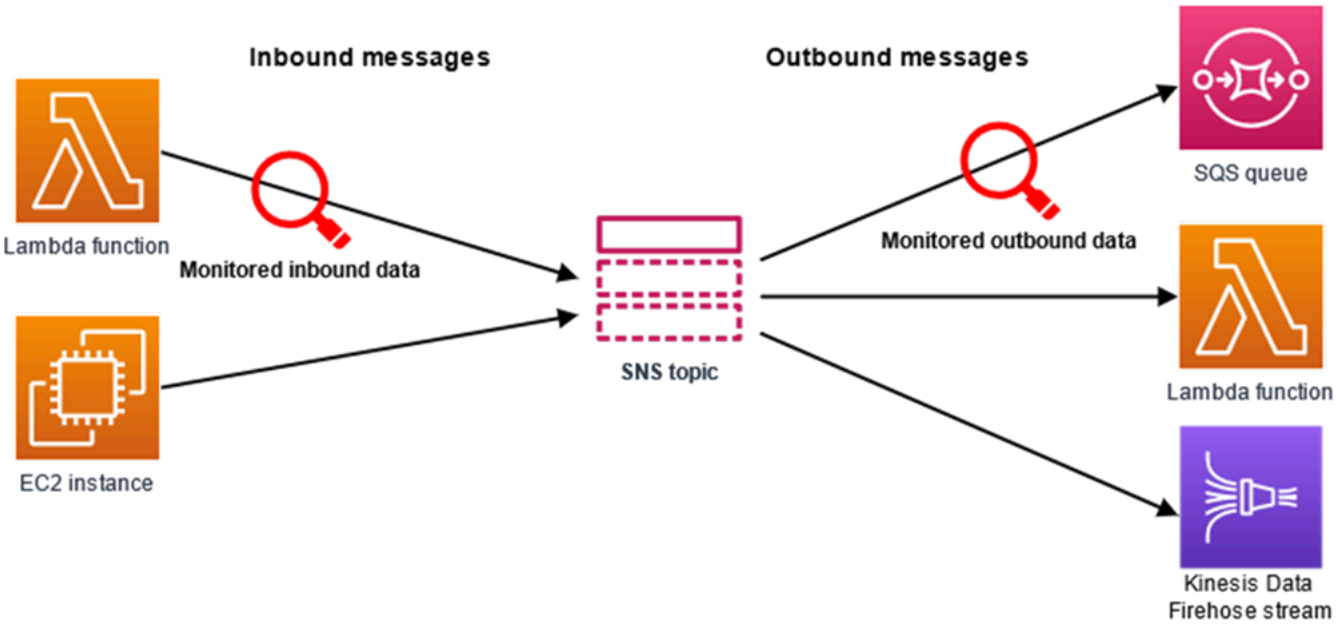
> **Note**
>
> Amazon SNS 僅支援 Amazon SNS 標準主題的訊息資料保護。

了解資料保護政策

主題

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。

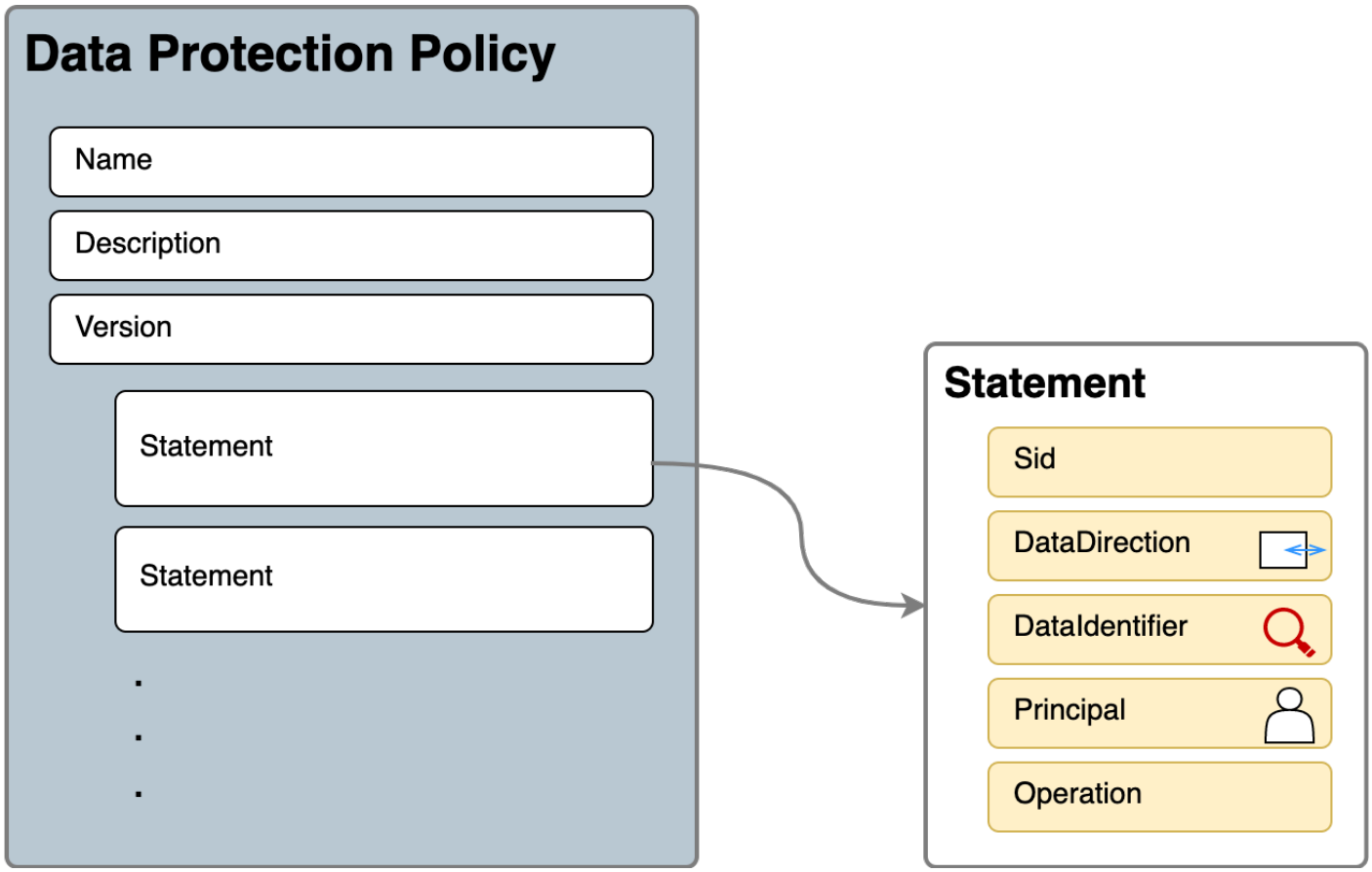


資料保護政策的結構如何？

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"
  }
}
```

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。

主題

- Audit 操作
- 去識別化操作
- Deny 操作

Audit 操作

Audit 操作會對主題傳入訊息進行取樣，並將敏感資料發現結果記錄在 AWS 目的地。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標。

2. NoFindingsDestination— Amazon SNS 主題在承載中找不到敏感資料時的記錄目標。

您可以在下列每個日誌目的地類型中使用下列 AWS 服務：

- Amazon CloudWatch 日誌 (選用) — LogGroup 必須位於主題區域中，且名稱必須以 /aw s/ vendedlogs/ 開頭。
- Amazon 數據 Firehose（可選）— DeliveryStream 必須位於主題區域，並以 Direct PUT 作為交付流的來源。如需其他詳細資訊，請參閱 Amazon 資料 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 儲存貯體作為日誌目的地，則可啟用採用 Amazon S3 受管金鑰 (SSE-S3) 的伺服器端加密或採用 AWS KMS keys (SSE-KMS) 的伺服器端加密，以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。如需與 SSE-KMS 搭配使用所需金鑰政策的詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的 Amazon S3 儲存貯體伺服器端加密。

稽核目的地日誌範例

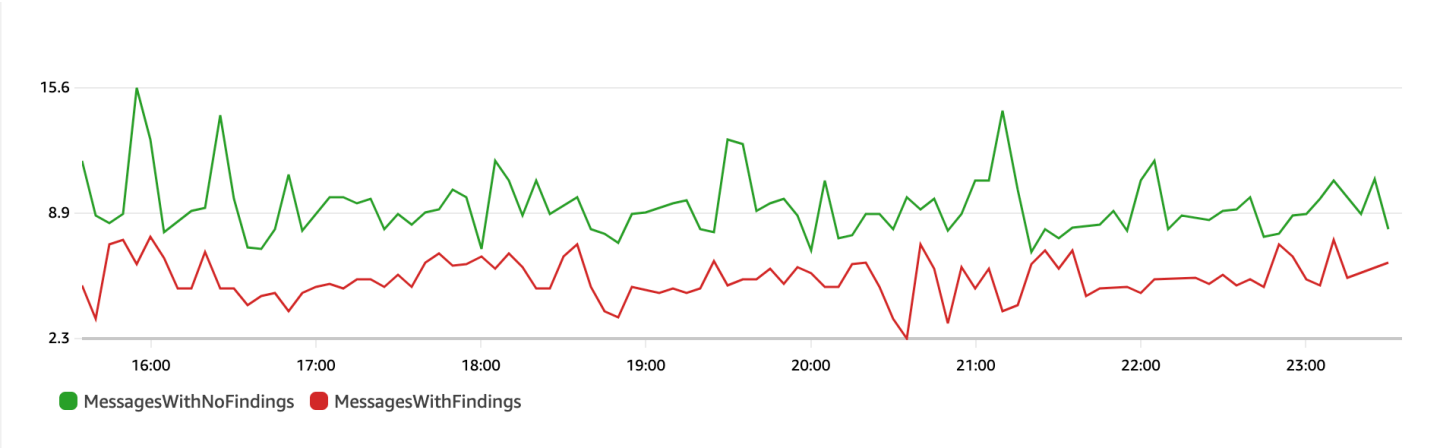
在以下範例中，`callerPrincipal` 用於識別敏感內容來源，`messageID` 用作檢查 `Publish` API 回應的參考。

```
{
  "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

當稽核作業指定了FindingsDestination或NoFindingsDestination屬性時，主題擁有者也會接收 CloudWatchMessagesWithFindings和MessagesWithNoFindings量度。



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
!	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

受支援的遮罩字元	名稱
_	底線
`	反引號
|	垂直分隔號
~	波浪符號

- RedactConfig— 通過完全刪除數據進行編輯。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"
}
```

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。

主題

- 稽核的範例政策
- 政策傳入去識別化遮罩陳述式範例
- 政策傳入去識別化修改陳述式範例
- 政策傳出去識別化遮罩陳述式範例
- 政策傳出去識別化修改陳述式範例
- 政策傳入拒絕陳述式範例
- 傳出拒絕陳述式的政策範例

稽核的範例政策

稽核政策可讓您稽核多達 99% 的輸入訊息，並將發現項目傳送至亞馬遜 CloudWatch、Amazon 資料 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，並將發現結果傳送至 CloudWatch 日誌、Firehose 和 Amazon S3，來稽核 99% 流經主題的訊息。

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

下列範例會防止 AWS 帳戶接收包含 `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": {
        "Deny": {}
      }
    }
  ]
}
```

輸出拒絕結果範例，已登入 Amazon CloudWatch：

```
{
  "notification": {
    "messageMD5Sum": "2e8f58ff2eeed723b56b15493fbfb5a5",
    "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
    "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
    "timestamp": "2022-09-08 15:40:57.144"
  },
  "delivery": {
```

```
    "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
    "destination": "arn:aws:sqs:us-east-2:664555388960:test",
    "providerResponse": "The topic's data protection policy prohibits this message from
 being delivered to <subscription arn>",
    "dwellTimeMs": 22,
    "attempts": 1,
    "statusCode": 403
  },
  "status": "FAILURE"
}
```

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

- 建立資料保護政策以保護訊息資料 (API)
- 建立資料保護政策以保護訊息資料 (CLI)
- 建立資料保護政策以保護訊息資料的安全 (CloudFormation)
- 建立資料保護政策以保護訊息資料 (主控台)
- 建立資料保護政策以保護訊息資料 (SDK)

建立資料保護政策以保護訊息資料 (API)

AWS 帳戶中的 Amazon SNS 資源數量和大小均有所限制。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 (AWS API)

您可以使用 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

建立資料保護政策以保護訊息資料 (CLI)

AWS 帳戶中的 Amazon SNS 資源數量和大小均有所限制。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 (AWS CLI)

您可以使用 AWS Command Line Interface 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 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

建立資料保護政策以保護訊息資料的安全 (CloudFormation)

AWS 帳戶中的 Amazon SNS 資源數量和大小均有所限制。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)

您可以使用 AWS CloudFormation 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 (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

建立資料保護政策以保護訊息資料 (主控台)

AWS 帳戶中的 Amazon SNS 資源數量和大小均有所限制。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. 輸入一般表示式 (RegEx) 做為自訂資料識別碼。 RegEx支援英數字元、 RegEx 保留字元和符號。 RegEx 最大長度為 200 個字元。如果過 RegEx 於複雜，Amazon SNS 將失敗 API 調用。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. 對於 Audit destination (稽核目的地)，選擇要將稽核發現結果傳送至哪一個 AWS 服務，並輸入要使用之每個 AWS 服務 的目的地名稱。您可以從以下 Amazon Web Services 中進行選擇：

- Amazon CloudWatch — CloudWatch 日誌是AWS標準的日誌記錄解決方案。使用 CloudWatch Logs，您可以使用 Logs Insights 執行日誌分析 (請參閱此處的範例)，並建立指標和警示。 CloudWatch 日誌是許多服務發布日誌的地方，這使得使用一個解決方案來彙總所有日誌變得更加容易。有關 Amazon 的信息 CloudWatch，請參閱 Amazon CloudWatch 用戶指南。
- 亞馬遜數據防火軟管 — Firehose 可以滿足對即時串流到 Splunk 和 Amazon Redshift 進行進一步日誌分析的需求。 OpenSearch有關 Amazon 數據 Firehose 的信息，請參閱 Amazon 數據 Firehose 用戶指南。
- 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 Define this de-identify statement for (定義此去識別化陳述式)，請選取要套用此去識別陳述式的 AWS 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 所有 AWS 帳戶，或使用帳戶 ID 或 IAM 實體 ARN 的 特定 AWS 帳戶 帳戶 或 IAM 實體 (例如：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`。

iv. (選用) 視需要繼續新增拒絕陳述式。

建立資料保護政策以保護訊息資料 (SDK)

AWS 帳戶中的 Amazon SNS 資源數量和大小均有所限制。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 (AWS SDK)

您可以使用 AWS SDK 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 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 AWS Management Console，刪除 Amazon SNS 資料保護政策。

如需 Amazon SNS 資料保護政策的一般資訊，請參閱 了解資料保護政策。

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。

主題

- 刪除資料保護政策 (主控台)
- 使用空的 JSON 字串刪除資料保護政策
- 使用 AWS CLI 刪除資料保護政策

刪除資料保護政策 (主控台)

刪除受管資料保護政策 (主控台)

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 AWS CLI 刪除資料保護政策

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 ""
```

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。

主題

- 在 Amazon SNS 中使用受管資料識別符
- 在 Amazon SNS 中使用自訂資料識別符

在 Amazon SNS 中使用受管資料識別符

主題

- 受管資料識別符是什麼？

- 敏感資料類型：憑證
- 敏感資料類型：裝置
- 敏感資料類型：財務
- 敏感資料類型：受保護醫療資訊 (PHI)
- 敏感資料類型：個人身分識別資訊 (PII)

受管資料識別符是什麼？

Amazon SNS 受管資料識別符的設計用途在於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：

- 憑證，例如私密金鑰或 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-美國。

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
IndividualTaxIdentificationNumber	個人	美國
InseeCode	個人	法國

識別符	類別	國家/地區/語言
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
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

敏感資料類型：憑證

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,	任何

偵測類型	受管資料識別符 ID	必要的關鍵字	國家和地區
		secret key, set-awscredential	
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: 資料保護:: aw: 資料識別碼/AwsSecretKey
arn: aws: 資料保護:: aw: 資料識別碼/OpenSshPrivateKey
arn: aws: 資料保護:: aw: 資料識別碼/PgpPrivateKey
arn: aws: 資料保護:: aw: 資料識別碼/PkcsPrivateKey
arn: aws: 資料保護:: aw: 資料識別碼/PuttyPrivateKey

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: 資料保護:: aw: 資料識別碼/IpAddress

敏感資料類型：財務

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、magstripe、stripe、swipe。	這包括第 1 軌和第 2 軌。	任何
信用卡號碼	CreditCardNumber	account number、american express、a	檢測要求數據是符合 Luhn 檢查公式的 13—19 位數序列，並為	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
		mex、bank card、card 、card num、card number、cc #、ccn、che ck card、cred it、credit card#、dan kort、debi t、debit card、dine rs club、disc over、elec tron、elo verification code、japanese card bureau、jc b、masterc ard、mc、pa n、payment account number、pa yment card number、pc n、union pay、visa	以下任何類型的信用卡使用標準卡號前綴：美國運通，Dankort，晚餐俱樂部，發現，電子，日本卡局（JCB）UnionPay，萬事達卡和 Visa（下面的上標鏈接 1）。	

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：

   122000000000003、2222405343248877、2222990905257051、2223007648726984、2223577120017 6 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, kartennum 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: 資料保護:: aws: 資料識別碼/-DE BankAccountNumber
arn: awn: 資料保護:: aws: 資料識別碼/-ES BankAccountNumber
arn: awn: 資料保護:: aws: 資料識別碼/-FR BankAccountNumber
arn: awn: 資料保護:: aws: 資料識別碼/-GB BankAccountNumber

財務資料識別符 ARN
ARN: aws: 資料保護:: aws: 資料識別碼/-IT BankAccountNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US BankAccountNumber
arn: aws: 資料保護:: aw: 資料識別碼/CreditCardExpiration
arn: aws: 資料保護:: aw: 資料識別碼/CreditCardNumber
arn: aws: 資料保護:: aw: 資料識別碼/CreditCardSecurityCode

敏感資料類型：受保護醫療資訊 (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
		surance card number、krankenversicherungskarte、krankenversicherungsnummer、medical account number、numero conto medico、numéro d’assurance maladie、numéro de carte d’assurance、numéro de compte medical、número de cuenta médica、número de seguro de salud、número de tarjeta de seguro、sairaanhoitokortin、sairausvakuutuskortti、sairausvakuutus numero、sjukförsäkring nummer、sjukförsäkringskort、suomi ehic-numero、tarjeta de salud、terveyskortti、tessera sanitaria assicurazione numero、versicherungsnummer	

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, npi	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, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
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, sairausvakuutusnum ero, sjukförsäkring nummer, sjukförsäkringskor t, 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: awn: 資料保護:: aws: 資料識別碼/-US DrugEnforcementAgencyNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US HealthcareProcedureCode
arn: aws: 資料保護:: aws: 資料識別碼/-EU HealthInsuranceCardNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US HealthInsuranceClaimNumber
arn: awn: 資料保護:: aws: 資料識別碼/-FR HealthInsuranceNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US MedicareBeneficiaryNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US NationalDrugCode
arn: awn: 資料保護:: aws: 資料識別碼/-GB NationalInsuranceNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US NationalProviderId
arn: awn: 資料保護:: aws: 資料識別碼/-GB NhsNumber

PHI 資料識別符 ARN
arn: awn: 資料保護:: aws: 資料識別碼/-CA PersonalHealthNumber

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	ElectoralRollNumber	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	IndividualTaxIdentificationNumber	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)	NationalI nsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationali nsurance# , nationali nsurancen umber, 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、cellular、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, socialsec urityno#, 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)；Steu eridentifikationsn ummer (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	Fahrgestellnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	Amazon SNS 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 prúkaz, 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, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
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, führerschäin
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 conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
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’identité, 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	passeport, passeport#, passport, passport#, passportno, passportno#
法國	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport 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 fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, 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, steueride ntifikationsnummer, 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: awn: 資料保護:: aws: 資料識別碼/-BR CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn: awn: 資料保護:: aws: 資料識別碼/-BR CpfCode
arn: aws: 資料保護:: aw: 資料識別碼/DateOfBirth
arn: awn: 資料保護:: aws: 資料識別碼/-AT DriversLicense

PII 資料識別符 ARN
arn: aws: 資料保護:: aws: 資料識別碼/-AU DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-BE DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-BG DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-CA DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-CY DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-CZ DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-DE DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-DK DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-EE DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-ES DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-FI DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-FR DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-GB DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-GR DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-HR DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-HU DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-IE DriversLicense
ARN: aws: 資料保護:: aws: 資料識別碼/-IT DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-LT DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-LU DriversLicense

PII 資料識別符 ARN
arn: awn: 資料保護:: aws: 資料識別碼/-LV DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-MT DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-NL DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-PL DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-PT DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-RO DriversLicense
arn: aws: 資料保護:: aws: 資料識別碼/-SE DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-SI DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-SK DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-US DriversLicense
arn: awn: 資料保護:: aws: 資料識別碼/-GB ElectoralRollNumber
arn: aws: 資料保護:: aw: 資料識別碼/EmailAddress
arn: awn: 資料保護:: aws: 資料識別碼/-US IndividualTaxIdentificationNumber
arn: awn: 資料保護:: aws: 資料識別碼/-FR InseeCode
arn: aws: 資料保護:: aw: 資料識別碼/LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn: aws: 資料保護:: aws: 資料識別碼/-DE NationalIdentificationNumber
arn: awn: 資料保護:: aws: 資料識別碼/-ES NationalIdentificationNumber
ARN: aws: 資料保護:: aws: 資料識別碼/-IT NationalIdentificationNumber
arn: awn: 資料保護:: aws: 資料識別碼/-ES NieNumber

PII 資料識別符 ARN
arn: awn: 資料保護:: aws: 資料識別碼/-ES NifNumber
arn: awn: 資料保護:: aws: 資料識別碼/-CA PassportNumber
arn: aws: 資料保護:: aws: 資料識別碼/-DE PassportNumber
arn: awn: 資料保護:: aws: 資料識別碼/-ES PassportNumber
arn: awn: 資料保護:: aws: 資料識別碼/-FR PassportNumber
arn: awn: 資料保護:: aws: 資料識別碼/-GB PassportNumber
ARN: aws: 資料保護:: aws: 資料識別碼/-IT PassportNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US PassportNumber
arn: awn: 資料保護:: aws: 資料識別碼/-CA PermanentResidenceNumber
arn: awn: 資料保護:: aws: 資料識別碼/-BR PhoneNumber
arn: aws: 資料保護:: aws: 資料識別碼/-DE PhoneNumber
arn: awn: 資料保護:: aws: 資料識別碼/-ES PhoneNumber
arn: awn: 資料保護:: aws: 資料識別碼/-FR PhoneNumber
arn: awn: 資料保護:: aws: 資料識別碼/-GB PhoneNumber
ARN: aws: 資料保護:: aws: 資料識別碼/-IT PhoneNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US PhoneNumber
arn: awn: 資料保護:: aws: 資料識別碼/-CA PostalCode
arn: awn: 資料保護:: aws: 資料識別碼/-BR RgNumber
arn: awn: 資料保護:: aws: 資料識別碼/-CA SocialInsuranceNumber
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: 資料保護:: aws: 資料識別碼/-DE TaxId
arn: awn: 資料保護:: aws: 資料識別碼/-ES TaxId
arn: awn: 資料保護:: aws: 資料識別碼/-FR TaxId
arn: awn: 資料保護:: aws: 資料識別碼/-GB TaxId
arn: aws: 資料保護:: aw: 資料識別碼/VehicleIdentificationNumber
arn: awn: 資料保護:: aws: 資料識別碼/-US ZipCode

在 Amazon SNS 中使用自訂資料識別符

主題

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

本節說明訊息傳遞的運作方式。

主題

Amazon SNS 原始訊息交付

為了避免亞馬遜數據 Firehose、Amazon SQS 和 HTTP/S 端點處理消息的 JSON 格式，Amazon SNS 允許原始消息交付：

- 當您為 Amazon 資料 Firehose 或 Amazon SQS 端點啟用原始訊息傳遞時，任何 Amazon SNS 中繼資料都會從已發佈的訊息中刪除，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、用户端 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 端點啟用原始郵件傳遞時，會傳遞郵件內文。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，選擇訂閱了 Firehose、Amazon SQS 或 HTTP/S 端點的主題。

4. 在***MyTopic***頁面的「訂閱」區段中，選擇訂閱，然後選擇 「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":
 "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/
zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm
+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy
+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/
LsDr0Iak4ZDpg8dXg==",
  "SigningCertURL": "https://sns.us-east-2.amazonaws.com/
SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
  "UnsubscribeURL": "https://sns.us-east-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
}
```

- 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 支援訊息屬性的交付，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、簽名和識別碼。對於已啟用原始訊息交付的 Amazon SQS 訂閱，最多可以傳送 10 個訊息屬性。若要傳送 10 個以上的郵件屬性，您必須停用原始郵件傳遞。不過，Amazon SNS 會捨棄具有超過 10 個訊息屬性的訊息，且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 (請參閱擴散到 Amazon SQS 佇列)。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。

主題

- 佇列擁有者建立訂閱
- 沒有擁有佇列的使用者建立訂閱
- 如何強制訂閱以要求對取消訂閱請求進行身分驗證？

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：使用AWS Management Console將 Amazon SQS 佇列訂閱新增到其他 AWS 帳戶中的主題

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。如需更多詳細資訊，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：使用AWS Management Console將 Amazon SQS 佇列訂閱新增到其他 AWS 帳戶中的主題

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 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. 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 AWS Management Console確認訂閱

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 訂閱命令必須在託管 Amazon SNS 之區域中的目標帳戶中執行。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
亞太 (大阪) 區域	ap-northeast-3
Europe (Milan) Region	eu-south-1
歐洲 (西班牙) 區域	eu-south-2

區域名稱	區域
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
中東 (阿拉伯聯合大公國) 區域	me-central-1

如需有關啟用選擇加入區域的資訊，請參閱中的Amazon Web Services 一般參考管理 AWS 區域。

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 `sns.<opt-in-region>.amazonaws.com`
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 `sns.<opt-in-region>.amazonaws.com`
選擇加入區域至選擇加入區域	不支援

  以下是存取政策陳述式的範例，該陳述式允許選擇加入區域 (距離南方 -1) 中的 Amazon SNS 主題傳遞至區域中的 Amazon SQS 佇列 (US-east-1)。enabled-by-default 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"
        }
      }
    },
    ...
  ]
}
```

- 若要將美國東部 (維吉尼亞北部) 的 AWS Lambda 函數訂閱亞太區域 (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 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 `sns.<opt-in-region>.amazonaws.com`	
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 使用下列 Amazon SNS 端點，提供將傳送至主題之通知訊息的交付狀態予以記錄的支援。

- HTTP
- Amazon 數據 Firehose
- AWS Lambda
- 平台應用程式端點
- Amazon Simple Queue Service

設定郵件傳遞狀態屬性之後，會針對傳送給主題訂閱者的郵件，將 CloudWatch 記錄項目傳送至記錄檔。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：

- 得知訊息是否已傳遞至 Amazon SNS 端點。
- 識別從 Amazon SNS 端點傳送至 Amazon SNS 的回應。
- 判定訊息暫留時間 (發布時間戳記到就在遞交至 Amazon SNS 端點前的時間)。

若要設定郵件傳遞狀態的主題屬性，您可以使用 AWS Management Console, AWS 軟體開發套件 (SDK)、查詢 API 或 AWS CloudFormation.

主題

- 使用 AWS Management Console設定交付狀態記錄日誌
- 使用 AWS SDK 設定傳送狀態記錄
- AWS 用於配置主題屬性的 SDK 示例
- 使用 AWS CloudFormation設定交付狀態記錄日誌

使用 AWS Management Console設定交付狀態記錄日誌

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. 在 [編輯] *MyTopic*頁面上，展開 [傳送狀態記錄] 區段。

5. 選擇您希望記錄交付狀態日誌的通訊協定，例如 AWS Lambda。
6. 輸入成功取樣率 (您要接收 CloudWatch 記錄檔的成功訊息百分比)。
7. 在 IAM roles (IAM 角色) 子區段中，執行以下其中一項作業：

   - 若要從您的帳戶選擇現有的服務角色，請選擇 Use existing service role (使用現有服務角色)，然後為成功和失敗交付指定 IAM 角色。

   - 若要在您的帳戶中建立新的服務角色，請選擇 Create new service role (建立新的服務角色)，選擇 Create new roles (建立新角色) 來在 IAM 主控台中為成功和失敗交付定義 IAM 角色。

     若要授予 Amazon SNS 寫入權限以代表您使用 CloudWatch 日誌，請選擇 [允許]。
8. 選擇儲存變更。

   您現在可以檢視和剖析包含郵件傳遞狀態的 CloudWatch 記錄檔。如需使用的詳細資訊 CloudWatch，請參閱CloudWatch文件。

使用 AWS SDK 設定傳送狀態記錄

AWS 開發套件提供多種語言的 API，可將訊息傳遞狀態屬性與 Amazon SNS 搭配使用。

主題屬性

您可以使用下列訊息傳遞狀態的主題屬性名稱值：

HTTP

- `HTTPSuccessFeedbackRoleArn` - 表示訂閱 HTTP 端點之 Amazon SNS 主題的成功訊息傳遞狀態。
- `HTTPSuccessFeedbackSampleRate` - 表示訂閱 HTTP 端點之 Amazon SNS 主題的訊息範例成功百分比。
- `HTTPFailureFeedbackRoleArn` - 表示訂閱 HTTP 端點之 Amazon SNS 主題的失敗訊息傳遞狀態。

Amazon 數據 Firehose

- `FirehoseSuccessFeedbackRoleArn` - 表示訂閱 Amazon Kinesis Data Firehose 端點之 Amazon SNS 主題的成功訊息傳遞狀態。

- `FirehoseSuccessFeedbackSampleRate` - 表示訂閱 Amazon Kinesis Data Firehose 端點之 Amazon SNS 主題的訊息範例成功百分比。
- `FirehoseFailureFeedbackRoleArn` - 表示訂閱 Amazon Kinesis Data Firehose 端點之 Amazon SNS 主題的失敗訊息傳遞狀態。

AWS Lambda

- `LambdaSuccessFeedbackRoleArn` - 表示訂閱 Lambda 端點之 Amazon SNS 主題的成功訊息傳遞狀態。
- `LambdaSuccessFeedbackSampleRate` - 表示訂閱 Lambda 端點之 Amazon SNS 主題的訊息範例成功百分比。
- `LambdaFailureFeedbackRoleArn` - 表示訂閱 Lambda 端點之 Amazon SNS 主題的失敗訊息傳遞狀態。

平台應用程式端點

- `ApplicationSuccessFeedbackRoleArn`— 指出訂閱 AWS 應用程式端點之 Amazon SNS 主題的成功訊息傳遞狀態。
- `ApplicationSuccessFeedbackSampleRate`— 針對訂閱 AWS 應用程式端點的 Amazon SNS 主題，指出要取樣的成功訊息百分比。
- `ApplicationFailureFeedbackRoleArn`— 指出訂閱 AWS 應用程式端點之 Amazon SNS 主題的失敗訊息傳遞狀態。

> **Note**
>
> 除了能夠設定傳送至 Amazon SNS 應用程式端點通知訊息的訊息交付狀態主題屬性，您也可以為傳送至推播通知服務的推播通知訊息，設定其交付狀態的應用程式屬性。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` - 表示訂閱 Amazon SQS 端點之 Amazon SNS 主題的成功訊息傳遞狀態。
- `SQSSuccessFeedbackSampleRate` - 表示訂閱 Amazon SQS 端點之 Amazon SNS 主題的訊息範例成功百分比。

- SQSFailureFeedbackRoleArn - 表示訂閱 Amazon SQS 端點之 Amazon SNS 主題的失敗訊息傳遞狀態。

> **Note**
>
> <ENDPOINT>SuccessFeedbackRoleArn和<ENDPOINT>FailureFeedbackRoleArn屬性用於授與 Amazon SNS 寫入存取權，以代表您使用 CloudWatch 日誌。<ENDPOINT>SuccessFeedbackSampleRate 屬性用於指定成功傳送訊息的取樣率百分比 (0-100)。設定<ENDPOINT>FailureFeedbackRoleArn屬性之後，所有失敗的郵件傳遞都會產生 CloudWatch 記錄檔。

AWS 用於配置主題屬性的 SDK 示例

下列程式碼範例會示範如何使用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SetTopicAttributes中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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SetTopicAttributes中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考SetTopicAttributes中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK SetTopicAttributes中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考SetTopicAttributes中的。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考SetTopicAttributes中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK SetTopicAttributes中的 SAP ABAP API 參考資料。

使用 AWS CloudFormation設定交付狀態記錄日誌

若要`DeliveryStatusLogging`使用進行設定 AWS CloudFormation，請使用 JSON 或 YAML 範本建立 AWS CloudFormation 堆疊。若要取得更多資訊，請參閱《 AWS CloudFormation 使用指南》中的`AWS::SNS::Topic`資源`DeliveryStatusLogging`屬性。以下是 JSON 和 YAML 中的 AWS CloudFormation 範本範例，可用來建立新主題或使用 Amazon SQS 通訊協定的所有`DeliveryStatusLogging`屬性更新現有主題。

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 (DLQ)。

主題

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 端點的策略重試時間總計不能超過 3,600 秒。這是硬性限制，無法再增加。

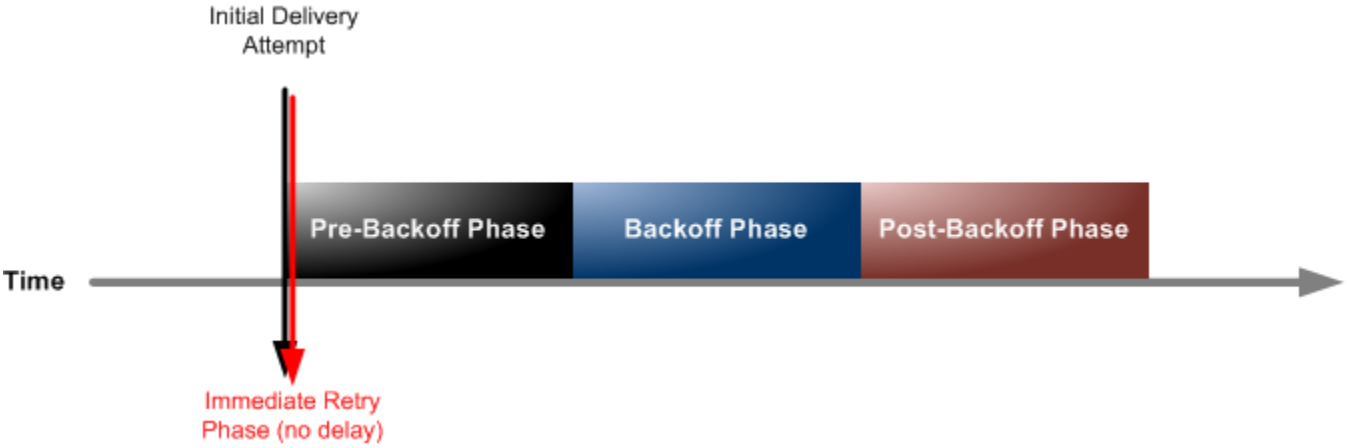
端點類型	傳遞通訊協定	立即重試 (無延遲) 階段	預先輪詢階段	輪詢階段	事後輪詢階段	嘗試總次數
AWS 受管理端點	Amazon 資料 Firehose [1]	3 次，毫無延遲	2 次，相隔 1 秒	10 次，含指數退避，從 1 秒到 20 秒	100,000 次，相隔 20 秒	100,015 次，在 23 天內
	AWS Lambda					
	Amazon SQS					
客戶管理的端點	SMTP	0 次，毫無延遲	2 次，相隔 10 秒	10 次，含指數退避，從 10 秒到	38 次，相隔 600 秒 (10 分鐘)	50 次嘗試，在 6 小時內
	SMS					

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
	行動推播			600 秒 (10 分鐘)		

[1] 針對 Firehose 通訊協定的節流錯誤，Amazon SNS 會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) - 此階段會在初始傳遞嘗試後立即發生。在此階段各次重試之間無延遲。
2. 預先輪詢階段 - 在立即重試階段之後。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. 輪詢階段 - 此階段會使用重試輪詢函數，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. 事後輪詢階段 - 此階段在輪詢階段之後。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

您可以使用交付政策及其四個階段，定義 Amazon SNS 如何重試將訊息傳遞至 HTTP/S 端點。舉例來說，當您想要根據 HTTP 伺服器容量自訂政策時，Amazon SNS 可讓您覆寫 HTTP 端點的預設重試政策。

您可以在訂閱或主題層級，將 HTTP/S 傳遞政策設為 JSON 物件。當您在主題層級定義政策時，它會套用到與該主題相關聯的所有 HTTP/S 訂閱。若要在訂閱層級設定傳送政策，您可以使用 `Subscribe` 或 `SetSubscriptionAttributes`API 動作。若要在主題層級設定傳送政策，您可以使用 `CreateTopic` 或 `SetTopicAttributes`API 動作。或者，您也可以在 AWS CloudFormation 模板中使用AWS::SNS::Subscription資源。

您應根據 HTTP/S 伺服器的容量自訂傳遞政策。您可以將政策設為主題屬性或訂閱屬性。如果您主題中的所有 HTTP/S 訂閱都將目標鎖定於同一個 HTTP/S 伺服器，建議您將傳遞政策設為主題屬性，這樣就能在該主題的所有 HTTP/S 訂閱中保持有效。否則，您必須根據政策鎖定目標之 HTTP/S 伺服器的容量，為您主題中的每個 HTTP/S 訂閱編寫傳遞政策。

您也可以設定請求政策中的 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定為 `text/plain; charset=UTF-8` 的 HTTP/S 端點。Amazon SNS 可讓您覆寫預設的請求政策。如需支援 `headerContentType` 與拘束，請參閱下列表格。

下列 JSON 物件代表傳遞政策，指示 Amazon SNS 重試失敗的 HTTP/S 傳遞嘗試，如下所示：

1. 3 次，在無延遲階段中立即執行
2. 2 次 (相隔 1 秒)，在預先輪詢階段中執行
3. 10 次 (含指數退避，從 1 秒到 60 秒)
4. 35 次 (相隔 60 秒)，在預先輪詢階段中執行

在此範例傳遞政策中，Amazon SNS 總共會嘗試 50 次，然後再捨棄訊息。若要在傳遞政策中指定的重試用盡後保留郵件，請設定訂閱以將無法傳遞的郵件移至無效字母佇列 (DLQ)。如需詳細資訊，請參閱Amazon SNS 無效字母佇列 (DLQ)。

> **Note**
>
> 此傳遞政策也會指示 Amazon SNS 利用 `maxReceivesPerSecond` 屬性將傳遞調節至每秒不超過 10 個。此自我限制速率可導致發佈的郵件 (輸入流量) 多於傳送 (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郵件待處理項目，這可能會導致郵件傳遞延遲過

高。在您的傳遞政策中，請務必為 maxReceivesPerSecond 指定一個數值，不會對您的工作負載造成負面影響。

> **Note**
>
> 此傳遞政策會將 HTTP/S 通知的預設內容類型覆寫為 application/json。

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

傳遞政策是由重試政策、限流政策和請求政策組成。一個傳遞政策中總共有 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

政策	描述	限制條件
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
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	每個訂閱的每秒最大傳遞次數。	1 或以上 預設值：無調節

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 • 應用程式/x-www-form-urlencoded • application/xhtml+xml • application/xml

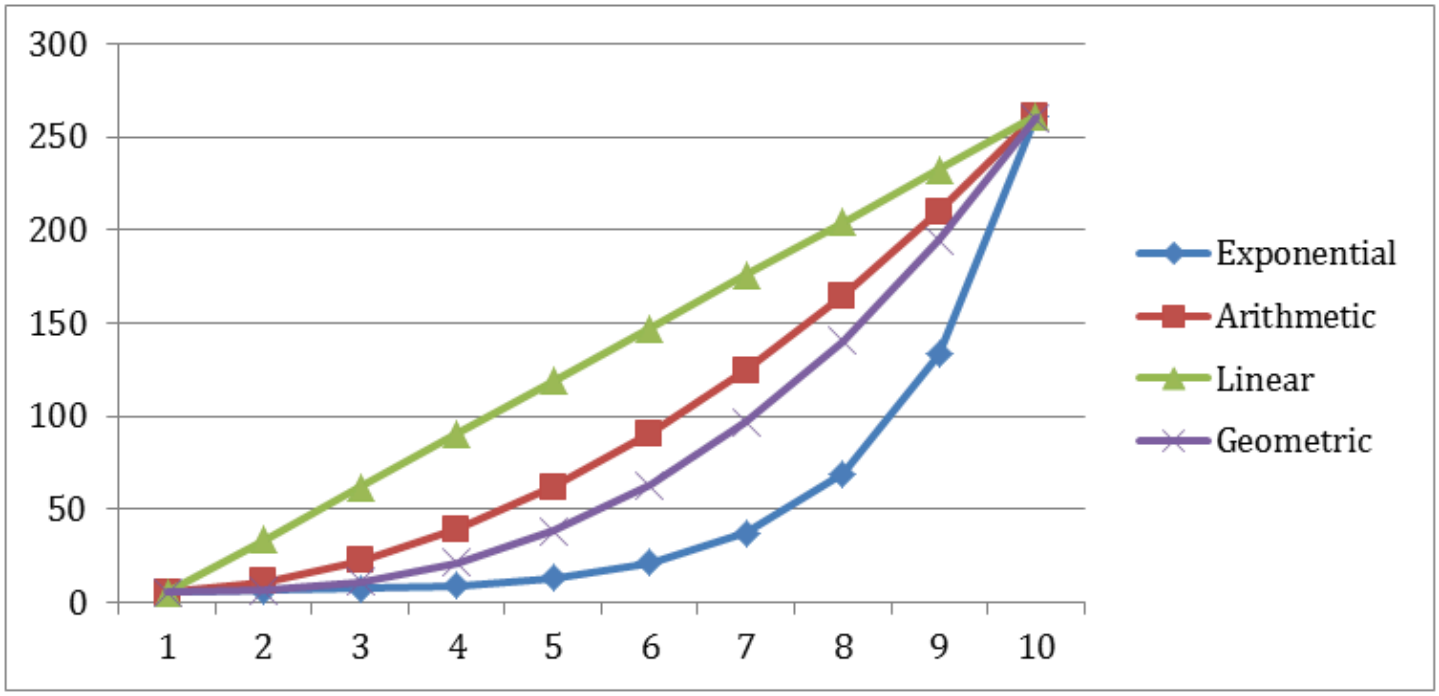
Amazon SNS 會使用下列公式計算輪詢階段中的重試次數：

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

您可以使用三個參數來控制輪詢階段中的重試頻率。

- `minDelayTarget` - 定義與輪詢階段中第一次重試嘗試相關聯的延遲。
- `maxDelayTarget` - 定義與輪詢階段中最後一次重試嘗試相關聯的延遲。
- `backoffFunction` - 定義 Amazon SNS 用來計算輪詢階段中第一次和最後一次重試間所有嘗試重試相關延遲的演算法。您可以使用四個重試輪詢函數中的其中一個函數。

下圖顯示每個重試輪詢函數如何影響與輪詢階段期間之重試相關聯的延遲：重試總次數設為 10、最小延遲設為 5 秒，且最大延遲設為 260 秒的傳遞政策。垂直軸代表 10 次重試的每一次相關的延遲 (以秒為單位)。水平軸代表重試次數，從第一次到第十次嘗試。



Amazon SNS 無效字母佇列 (DLQ)

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。如需更多詳細資訊，請參閱 設定訂閱的 Amazon SNS 無效字母佇列 及 Amazon SNS 訊息傳遞重試。

> **Note**
>
> - Amazon SNS 訂閱和 Amazon SQS 佇列必須位於相同 AWS 帳戶和區域。
> - 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，您必須將自訂 KMS 搭配金鑰政策，以授與 Amazon SNS 服務主體對 AWS KMS API 動作存取權限。如需詳細資訊，請參閱本指南中的 靜態加密 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。

主題

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。如需更多詳細資訊，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。對於以 Amazon SQS 或 AWS Lambda 為後端之 AWS 受管端點所導致的伺服器端錯誤，Amazon SNS 會在 23 天內最多重試傳遞 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 必須指向同一個 AWS 帳戶 中的 Amazon SQS 佇列和區域，做為 Amazon SNS 訂閱。如需更多詳細資訊，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。
- 編寫 Amazon SQS 消費者邏輯 - 使用 Amazon SQS API、AWS 開發套件或 AWS CLI 編寫自訂消費者邏輯，以在無效字母佇列中輪詢、處理及刪除訊息。

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 開發人員指南中的適用於 Amazon SQS 的 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 如需更多詳細資訊，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。例如，當您預期無效字母佇列一律為空白時，您可為 NumberOfMessagesSent 指標建立 CloudWatch 警示。您可以將警示閾值設為 0，並指定 Amazon SNS 主題在警示發出時接獲通知。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。如需更多詳細資訊，請參閱 Amazon SNS 訊息傳遞狀態。

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。如需更多詳細資訊，請參閱 Amazon SNS 無效字母佇列 (DLQ) 及 Amazon SNS 訊息傳遞重試。

本頁顯示如何使用 AWS Management Console AWS SDK 和設定 Amazon SNS 訂 AWS CloudFormation 閱的無效字母佇列。 AWS CLI

> **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。

必要條件

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. （略過 AWS CloudFormation）訂閱佇列以了解主題。
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。

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

主題

- 若要 Amazon SNS 用 AWS Management Console
- 使用開發套件為 Amazon SNS 訂閱設定無效字母佇列 AWS
- 若要 Amazon SNS 用 AWS CLI
- 若要使用以下方式設定 Amazon SNS 訂閱的無效字母佇列 AWS CloudFormation

若要 Amazon SNS 用 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. 選擇儲存變更。

您的訂閱已設為使用無效字母佇列。

使用開發套件為 Amazon SNS 訂閱設定無效字母佇列 AWS

執行此範例之前，請確定您已完成先決條件。

要使用 AWS SDK，您必須使用憑據對其進行配置。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。

下列程式碼範例會示範如何使用SetSubscriptionAttributesRedrivePolicy。

Java

適用於 Java 1.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// 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);
```

若要 Amazon SNS 用 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\"}"
```

若要使用以下方式設定 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 會開始建立`MyDeadLetterQueue`堆疊，並顯示「建立中 _ 進度」狀態。當程序完成時， AWS CloudFormation 會顯示「建立 _ 完成」狀態。

Amazon SNS 訊息封存、重播和分析

Amazon SNS 標準主題支援透過 Amazon 資料 Firehose 存檔訊息。您可以將通知散佈到 Firehose 交付串流，這可讓您將通知傳送到 Firehose 支援的儲存和分析目的地，包括亞馬遜簡單儲存服務 (Amazon S3)、亞馬遜 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 主題的訊息封存與重播功能。

功能	標準主題	FIFO 主題
訊息封存	扇出到 Firehose 交付流	FIFO 主題擁有者的訊息封存
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 主題訂閱用戶的訊息重播

使用 Amazon SNS 進行應用程式至應用程式 (A2A) 訊息

本節提供有關使用 Amazon SNS 與訂閱者進行應用程式對應用程式傳訊的資訊。

主題

扇出到 Firehose 交付流

您可以將 Amazon 資料 Firehose 交付串流訂閱到 Amazon SNS 主題，以便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，並依照 Firehose 中設定的方式傳送至目的地。訂閱擁有者最多可以為一個 Amazon SNS 主題訂閱五個 Firehose 交付串流。每個 Firehose 交付串流都有預設配額，可用於要求和每秒輸送量。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 通知傳送至 Amazon 簡單儲存服務 (Amazon S3)、亞馬遜 Redshift、亞馬遜 OpenSearch 服務 (服OpenSearch 務)，以及第三方服務提供者，例如資料多、新遺物、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。若要這麼做，請使用 S3 儲存貯體目的地建立 Firehose 交付串流，並訂閱該交付串流至 Amazon SNS 主題。另一個範例是，若要對傳送至 Amazon SNS 主題的訊息執行分析，請使用 OpenSearch 服務索引目的地建立交付串流。然後，您可以將 Firehose 交付串流訂閱至 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。如需更多詳細資訊，請參閱 Amazon SNS 訊息傳遞狀態。

主題

- 訂閱 Amazon SNS 主題的 Firehose 交付串流的先決條件
- 訂閱 Amazon SNS 主題的 Firehose 交付串流
- 使用交付串流目的地
- 訊息封存和分析的範例使用案例

訂閱 Amazon SNS 主題的 Firehose 交付串流的先決條件

若要訂閱某個 SNS 主題的 Amazon 資料 Firehose 交付串流，您AWS 帳戶必須具備以下條件：

- 標準 SNS 主題。如需詳細資訊，請參閱 建立 Amazon SNS 主題。
- 一個 Firehose 投遞流。如需詳細資訊，請參閱 Amazon 資料 Firehose 開發人員指南中的建立 Amazon 資料 Firehose 交付串流，並授予應用程式對 Firehose 資源的存取權。
- AWS Identity and Access Management (IAM) 角色，信任 Amazon SNS 服務主體並具有寫入交付串流的許可。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，這可讓 Amazon SNS 在 Firehose 交付串流中放置記錄。

  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 的完整權限，您也可以使用AWS受管理的原則AmazonKinesisFirehoseFullAccess。或者，若要提供更嚴格的使用 Firehose 權限，您可以建立自己的原則。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 Amazon SNS 主題的 Firehose 交付串流

若要將 Amazon SNS 通知傳送到 Amazon 資料 Firehose 交付串流，請先確定您已解決所有先決條件。如需支援的端點清單，請參閱中的 Amazon 資料 Firehose 端點和配額。Amazon Web Services 一般參考

若要訂閱某個主題的 Firehose 傳送串流

1. 登入 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Subscriptions (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. 針對 Topic ARN (ARN 主題)，選擇標準主題的 Amazon 資源名稱 (ARN)。
   b. 在「通訊協定」中，選擇「Firehose」

c. 對於端點，請選擇可接收來自 Amazon SNS 通知的 Firehose 交付串流的 ARN。

d. 對於訂閱角色 ARN，請指定您為寫入 Firehose 交付串流而建立之 AWS Identity and Access Management (IAM) 角色的 ARN。如需詳細資訊，請參閱 訂閱 Amazon SNS 主題的 Firehose 交付串流的先決條件。

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 (DLQ)。

7. 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

使用交付串流目的地

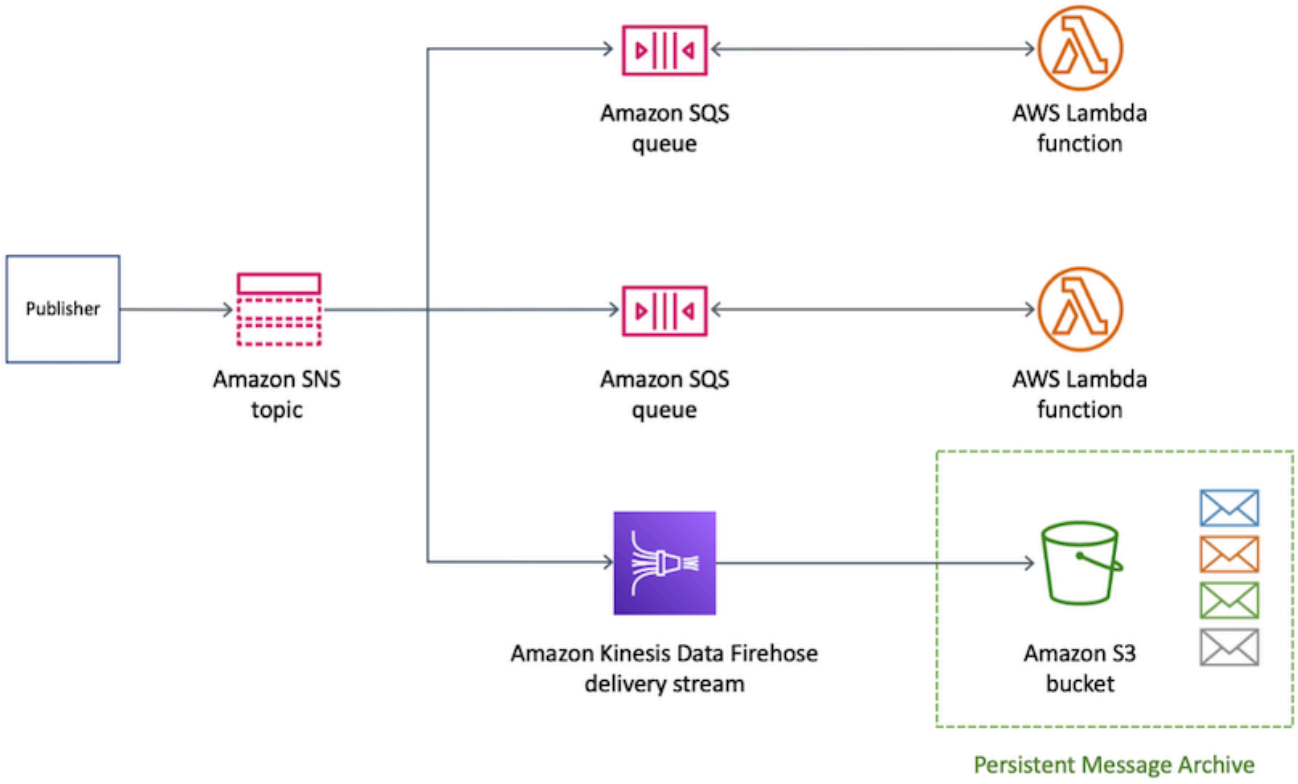
透過 Amazon 資料 Firehose 交付串流，您可以將訊息傳送到其他端點。本節說明如何使用支援的目的地。

主題

- Amazon S3 目的地
- OpenSearch 服務目的地
- Amazon Redshift 目的地
- HTTP 目的地

Amazon S3 目的地

本節提供將資料發佈到亞馬遜簡單儲存服務 (Amazon S3) 的 Amazon 資料 Firehose 交付串流的相關資訊。



主題

- Amazon S3 目的地的已封存訊息格式
- 分析 Amazon S3 目的地的訊息

Amazon S3 目的地的已封存訊息格式

以下範例顯示傳送到 Amazon Simple Storage Service (Amazon S3) 儲存貯體的 Amazon SNS 通知，使用縮排來提高可讀性。

> **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 資料 Firehose 交付串流傳送至相同 Amazon S3 儲存貯體的三個 SNS 訊息。緩衝被帶入帳戶內，換行分隔訊息。

```
{"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:s
east-1:333333333333:my-kinesis-test-topic","Subject":"My 3rd subject","Message":"My
 3rd body","Timestamp":"2020-11-27T00:31:39.755Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5"}
```

分析 Amazon S3 目的地的訊息

本頁說明如何分析透過 Amazon 資料 Firehose 交付串流傳送至亞馬遜簡單儲存服務 (Amazon S3) 目的地的 Amazon SNS 訊息。

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。如需指示，請參閱 Amazon 資料 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。

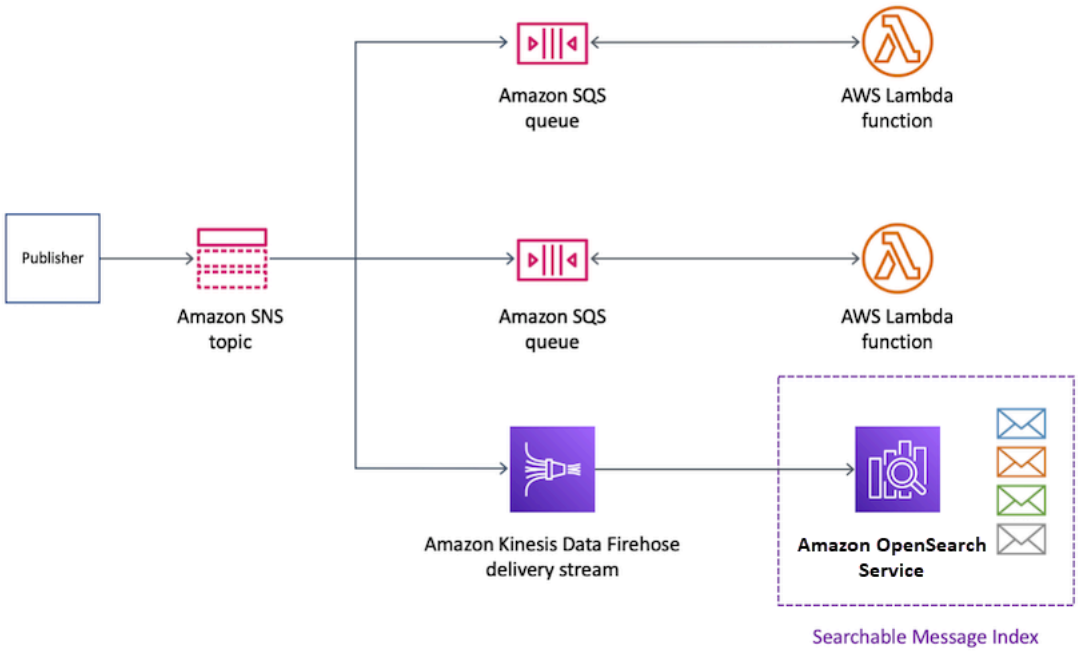
下列查詢會傳回在指定日期範圍內收到的所有 SNS 訊息：

```
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';
```

OpenSearch 服務目的地

本節提供將資料發佈到 Amazon OpenSearch 服務 (服OpenSearch 務) 的 Amazon 資料 Firehose 交付串流的相關資訊。



主題

- OpenSearch Service 索引中的封存訊息格式
- 分析 OpenSearch 服務目的地的訊息

OpenSearch Service 索引中的封存訊息格式

以下範例顯示 Amazon SNS 通知，傳送至命名為 `my-index` 的 Amazon OpenSearch Service (OpenSearch Service) 索引。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": "4961310096311132320325040540219328379477388655098593280 2.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
  },
  "sort": [
    1606948161189
```

```
  ]
}
```

分析 OpenSearch 服務目的地的訊息

本頁說明如何分析透過 Amazon 資料 Firehose 交付串流傳送至 Amazon OpenSearch 服務 (服 OpenSearch 務) 目的地的 Amazon SNS 訊息。

分析透過 Firehose 交付串流傳送至 OpenSearch 服務目的地的 SNS 訊息

1. 設定您的 OpenSearch 服務資源。如需指示，請參閱 Amazon OpenSearch 服務開發人員指南中的開始使用 Amazon OpenSearch 服務。
2. 設定交付串流。如需指示，請參閱 Amazon 資料 Firehose 開發人員指南中的為您的目的地選擇 OpenSearch 服務。
3. 使用 OpenSearch 服務查詢和 Kibana 執行查詢。如需詳細資訊，請參閱 Amazon OpenSearch 服務開發人員指南中的步驟 3：搜尋 OpenSearch 服務網域中的文件和 Kibana。

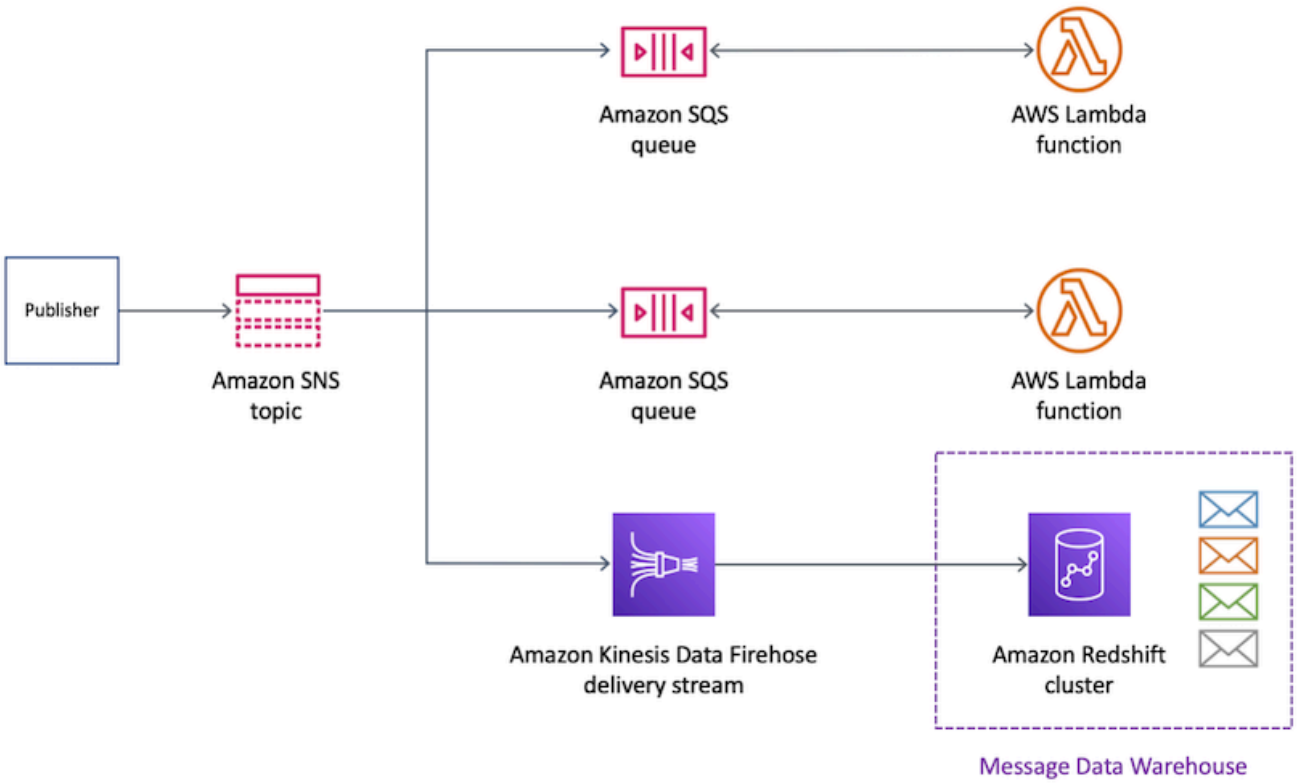
查詢範例

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"
            }
          }
        }
      ]
    }
  }
}
```

Amazon Redshift 目的地

本節說明如何將 Amazon SNS 通知散播至將資料發佈到 Amazon Redshift 的亞馬遜資料 Firehose 交付串流。使用此組態，您可以連線到 Amazon Redshift 資料庫，並使用 SQL 查詢工具來查詢資料庫中符合特定條件的 Amazon SNS 訊息。



主題

- 封存 Amazon Redshift 目的地的表格結構
- 分析 Amazon Redshift 目的地的訊息

封存 Amazon Redshift 目的地的表格結構

對於 Amazon Redshift 端點，已發佈的 Amazon SNS 訊息會以表格中的列形式封存。以下是範例。

> **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
Notification	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:my-topic:3	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						26deeeb-c bf4-45da-b92b-ca77 a247813b	
Notification	ab124832-a0d8-581d-9275-108 243c46114	arn:aws:sns:us-east-1:11111 1111111: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-1:11111 1111111:my-topic:326deeeb-c bf4-45da-b92b-ca77 a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
Notification	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 Amazon Redshift 目的地。

分析 Amazon Redshift 目的地的訊息

本頁說明如何分析透過 Amazon 資料 Firehose 交付串流傳送至亞馬遜 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。如需指示，請參閱 Amazon 資料 Firehose 開發人員指南中的「為目的地選擇亞馬遜紅移」。
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 `timestamptz` 的 `timestamp` 欄。

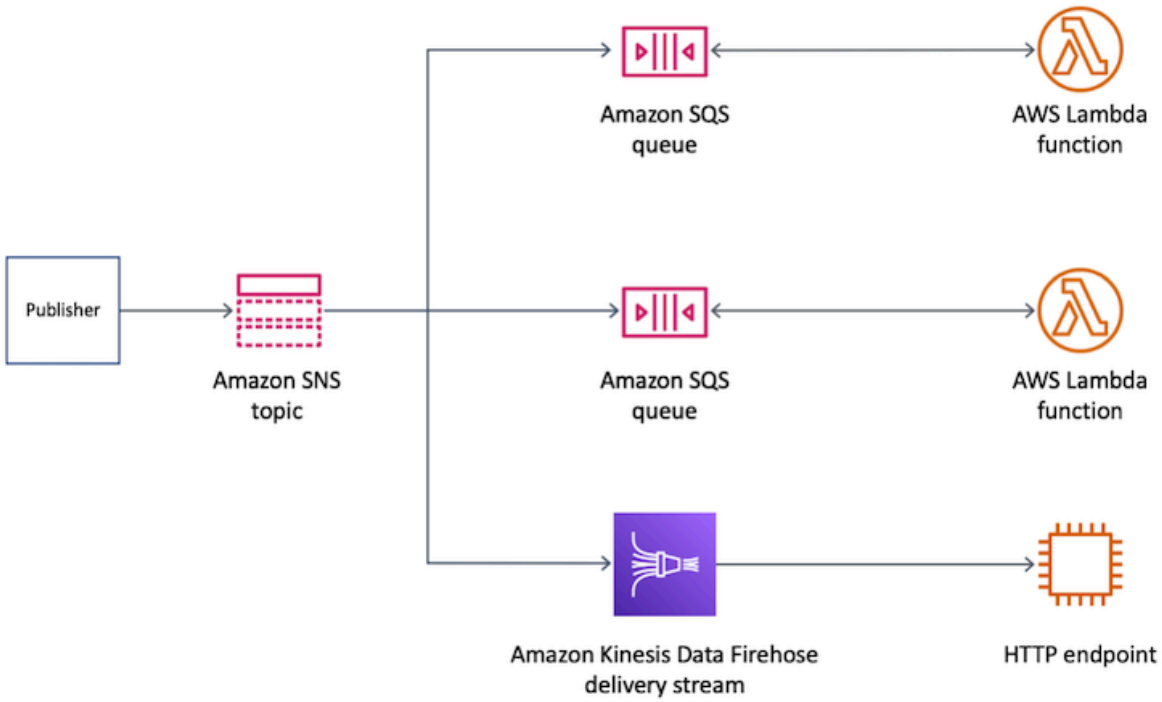
> **Note**
>
> 若要轉換 Amazon Redshift 端點的 JSON 中繼資料，您可以使用 SQL `COPY` 命令。如需詳細資訊，請參閱 Amazon Redshift 資料庫開發人員指南中的從 JSON 範例複製和使用「auto 忽略」選項從 JSON 資料載入。

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';
```

HTTP 目的地

本節提供將資料發佈到 HTTP 端點的 Amazon 資料 Firehose 交付串流的相關資訊。



主題

- 針對 HTTP 目的地傳遞的訊息格式

針對 HTTP 目的地傳遞的訊息格式

以下是來自 Amazon SNS 的 HTTP POST 請求主體範例，Amazon 資料 Firehose 交付串流可以傳送到 HTTP 端點。SNS 通知會在 `records` 屬性中編碼為 base64 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。如需原始交付的詳細資訊，請參閱 Amazon SNS 原始訊息交付。

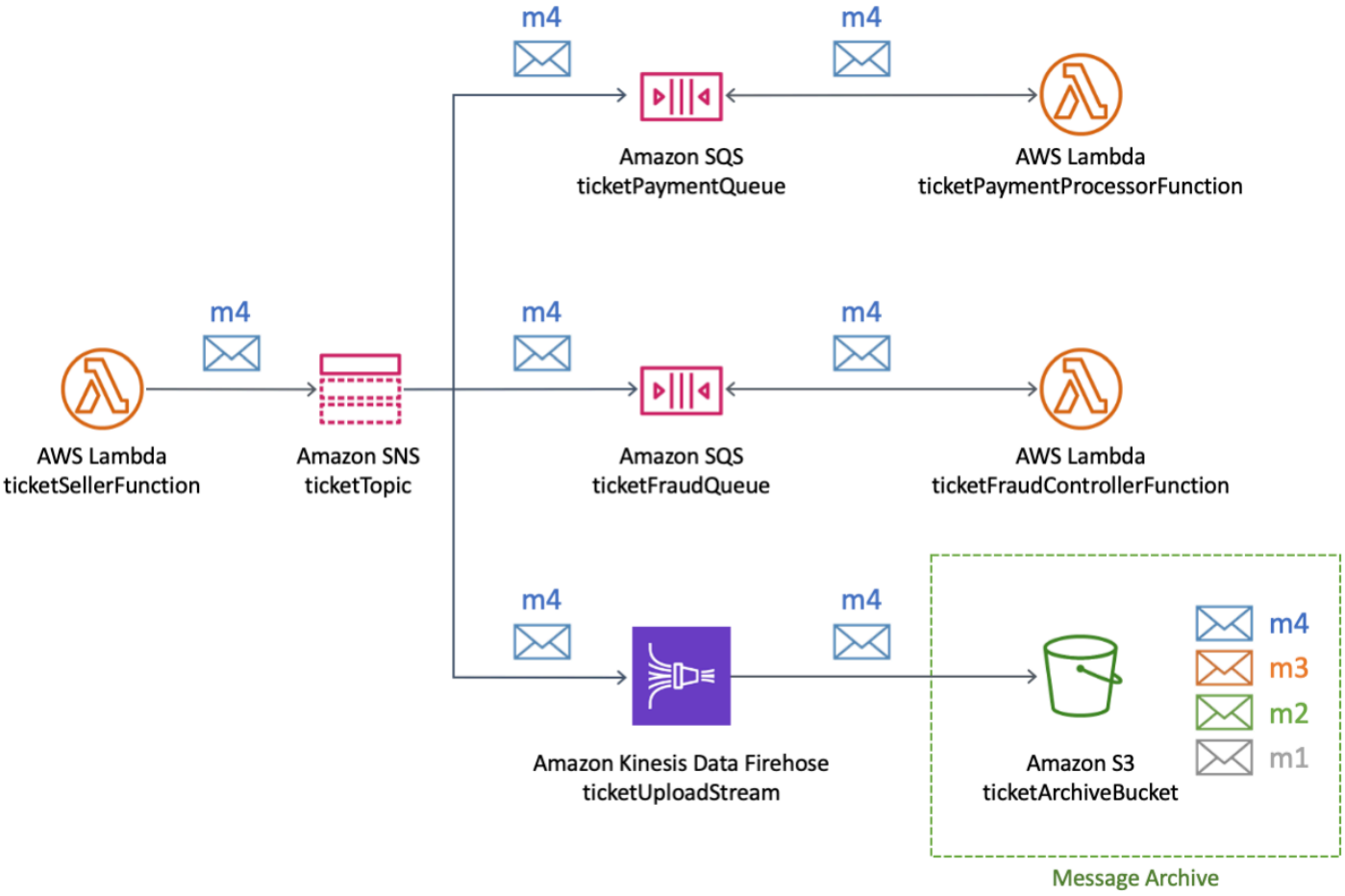
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
```

```
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```


訊息封存和分析的範例使用案例

本節提供封存和分析 Amazon SNS 訊息的常見使用案例教學課程。

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。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。為了達到資料保留的合規目標，該公司將 Amazon 資料 Firehose 交付串流訂閱至現有 SNS 主題。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。

建立此使用案例的 AWS 資源，您可以使用 AWS Management Console 或 AWS CloudFormation 範本。

主題

建立初始資源

本頁說明如何建立下列資源訊息封存和分析範例使用案例：

- 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 Create bucket (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 Create bucket (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 Create queue (建立佇列)。

c. 針對 Type (類型)，選擇 Standard (標準)。

d. 對於 Name (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 數字 (*123456789012*) 取代為自己的，並將 AWS 區域 (*us-east-1*) 隨之變更。

g. 選擇 Create queue (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 Create topic (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 Name (名稱)，輸入 **ticketTopic**。
e. 請選擇 Create topic (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 Create subscription (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 訂閱 Amazon SNS 主題。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 建立「Firehose」交付串流。

建立「Firehose」交付串流

本頁說明如何為訊息存檔和分析範例使用案例建立 Amazon Data Firehose 交付串流。

若要建立「Firehose」交付串流

1. 開啟 Amazon Kinesis 服務主控台。
2. 選擇 Firehose，然後選擇創建交付流。
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 Configure settings (組態設定) 頁面上的 Permissions (許可)，請選擇建立AWS Identity and Access Management(IAM) 角色，且會自動指派必要許可。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. 在 Kinesis Data Firehose 傳送串流頁面中，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。

如需有關建立交付串流的詳細資訊，請參閱 Amazon 資料 Firehose 開發人員指南中的建立 Amazon 資料 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。

您已建立具有必要權限的 Firehose 傳送串流。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

這個頁面會說明如何建立訊息封存和分析範例使用案例：

- 允許 Amazon SNS 訂閱將記錄放入 Amazon 資料 Firehose 交付串流的 AWS Identity and Access Management (IAM) 角色
- Firehose 傳送串流訂閱 SNS 主題

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 Next: Permissions (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*) 取代為自己的，並將 AWS 區域 (*us-east-1*) 隨之變更。

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將許可委派給 AWS 服務。

若要訂閱 SNS 主題的 Firehose 傳送串流

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「詳細資料」下，對於「通訊協定」，選擇 Amazon 資 Firehose
4. 在端點中，輸入您先前建立的ticketUploadStream交付串流的 Amazon 資源名稱 (ARN)。例如，輸入 **arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream**。
5. 對於訂閱角色 ARN，請輸入您先前建立的 ticketUploadStreamSubscriptionRoleIAM 角色的 ARN。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 測試和查詢組態。

測試和查詢組態

本頁描述了如何透過將訊息發佈至 Amazon SNS 主題，測試訊息封存和分析範例使用案例。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 Amazon SNS 訊息發布。

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 數據 Firehose 交付流
- AWS Identity and Access Management (IAM) 角色和政策

使用 AWS CloudFormation 範本

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
```

擴散到 Lambda 函數

Amazon SNS 和 AWS Lambda 已整合在一起，所以您可以使用 Amazon SNS 通知來叫用 Lambda 函數。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 函數接收訊息酬載做為輸入參數，且可以運用訊息中的資訊、將該訊息發佈到其他 SNS 主題或傳送到其他 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。如需更多詳細資訊，請參閱 Amazon SNS 訊息傳遞狀態。

Prerequisites

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 不支援從預設啟用區域到選擇加入區域的跨區域傳遞 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。

使用函數訂閱主題

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 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. 驗證所選擇的 Topic ARN (主題 ARN)。
   b. 針對 Protocol (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。如需如何使用 AWS Lambda 與 Amazon SNS (包含教學)，請參閱使用 AWS Lambda 與 Amazon SNS。

擴散到 Amazon SQS 佇列

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，請執行下列其中一項操作：

- 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。
- 請遵循下列步驟：
  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。

如需查看 AWS CloudFormation 範本，此範本會建立將訊息傳送至兩個佇列的主題，請參閱使用 AWS CloudFormation 範本建立會將訊息傳送至 Amazon SQS 佇列的主題。

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，並在 https://console.aws.amazon.com/sqs/ 開啟 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，並在 https://console.aws.amazon.com/sqs/ 開啟 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 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. 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

您應使用 AWS Identity and Access Management (IAM) 僅允許適當的使用者發佈到 Amazon SNS 主題，以及讀取/刪除來自 Amazon SQS 佇列的訊息。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。如果您想要提供主題或佇列的許可給其他 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 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**
>
> 當您提供其他 AWS 帳戶 存取您帳戶中資源的權限時，您同時也提供具有管理員層級權限 (萬用字元存取) 的 IAM 使用者該資源的許可。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 的登入資料或具有發佈至主題之許可的 IAM 使用者，在 https://console.aws.amazon.com/sns/ 登入 AWS Management Console 並開啟 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 帳戶的登入資料或具有檢視佇列中訊息許可的 IAM 使用者，在 https://console.aws.amazon.com/sqs/ 登入 AWS Management Console 並開啟 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 AWS CloudFormation 範本建立會將訊息傳送至 Amazon SQS 佇列的主題

AWS CloudFormation 可讓您使用範本檔案，並將一系列的 AWS 資源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。

如需有關使用 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 AWS 帳戶 內設定主題和佇列

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 Browse sample templates by AWS service (依照 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"]
            }
          ]
        ]
      }
    }
  }
}
```

擴散到 HTTP/S 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。

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
    Content-Length: 761
    Content-Type: text/plain; charset=UTF-8
    Host: ec2-50-17-44-49.compute-1.amazonaws.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" :
 "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

主題

- 讓 HTTP/S 端點訂閱主題
- 驗證 Amazon SNS 訊息的簽章
- 剖析訊息格式

讓 HTTP/S 端點訂閱主題

本節中的頁面說明如何訂閱 Amazon SNS 主題的 HTTP/S 端點。

主題

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.\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 SubscribeURLURL (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。連線將在 15 秒內逾時。如果您的端點在逾時前沒有回應，或者您的端點傳回 200–4xx 範圍外的狀態碼，Amazon SNS 會考慮傳遞訊息為失敗的嘗試。
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 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 Topics (主題)。
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 Close (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：確認訂閱

在您訂閱您的端點之後，Amazon SNS 會傳送訂閱確認訊息至端點。您應該已經將擁有執行步驟 1 中所述動作的程式碼部署至端點。特別是，在端點的程式碼必須擷取訂閱確定訊息的 `SubscribeURL`值，並且造訪 `SubscribeURL` 本身所指定的位置，或使其可供您使用，以便您可以手動造訪 `SubscribeURL`，例如使用 Web 瀏覽器。Amazon SNS 不會傳送訊息至端點，除非已經確認訂閱。當您造訪 `SubscribeURL` 時，回應將會包含 XML 文件，其中含有為訂閱指定 ARN 的元素 `SubscriptionArn`。您也可以使用 Amazon SNS 主控台來驗證訂閱已確認：Subscription ID (訂閱 ID) 會顯示訂閱的 ARN，而不是您首次新增訂閱時看到的 `PendingConfirmation` 值。

步驟 4：設定訂閱的傳遞政策 (選用)

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。您也可以在 `DeliveryPolicy` 中指定 HTTP/S 通知的內容類型。如需更多詳細資訊，請參閱 建立 HTTP/S 傳遞政策。

步驟 5：提供使用者發佈至主題的許可 (選用)

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，您應使用 AWS Identity and Access Management (IAM) 來提供發佈至主題的許可。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**
>
> 當您提供其他 AWS 帳戶 存取您帳戶中資源的權限時，您同時也提供具有管理員層級權限 (萬用字元存取) 的 IAM 使用者該資源的許可。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":[{
    "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：傳送訊息至 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 AWS 帳戶 的登入資料或具有發佈至主題之許可的 IAM 使用者，在 https://console.aws.amazon.com/sns/ 登入 AWS Management Console 並開啟 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

若要驗證 Amazon SNS 傳送至 HTTP 端點之訊息的真實性，您可以驗證訊息簽章。在兩種情況下，我們建議您驗證訊息的真實性。第一個，當 Amazon SNS 傳送訊息到您訂閱某個主題的 HTTP 端點時。

第二個，當 Amazon SNS 在執行 Subscribe 或 Unsubscribe API 動作時向您傳送確認訊息至您的 HTTP 端點。

您應在驗證 Amazon SNS 所傳送的訊息時執行以下動作：

- 從 Amazon SNS 取得憑證時始終使用 HTTPS。
- 驗證憑證的真偽。
- 驗證從 Amazon SNS 接收的憑證。
- 可行的話，為 Amazon SNS 使用其中一個支援的 AWS 開發套件，來驗證和確認訊息。
- 驗證 Amazon SNS 訊息是否已從您想要的 TopicArn 接收。

Amazon SNS 支援兩種訊息簽章版本：

- SignatureVersion1：Amazon SNS 根據訊息的 SHA1 雜湊值建立簽章。
- SignatureVersion 2：Amazon SNS 根據訊息的 SHA256 雜湊值建立簽章。

若要在 Amazon SNS 主題上設定訊息簽章版本

根據預設，Amazon SNS 主題會使用 SignatureVersion 1。若要在您的 Amazon SNS 主題上選擇雜湊演算法 SignatureVersion 1 (SHA1) 或 SignatureVersion 2 (SHA256)，您可以使用 SetTopicAttributes API 動作。

下列程式碼範例示範如何使用 AWS CLI 設定主題屬性 SignatureVersion：

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

使用 HTTP 查詢式的請求時，驗證 Amazon SNS 訊息的簽章

1. 從 Amazon SNS 傳送至您的端點的 HTTP POST 請求內文中的 JSON 文件，擷取名稱值組。您將使用一些名稱值組的值來建立要簽署的字串。當您要驗證 Amazon SNS 訊息的簽章時，這對您將逸出控制字元轉換為 Message 和 Subject 值中其原始字串表示法很重要。這些值在您將其用作要簽署之字串的一部分時必須為其原始形式。如需關於如何剖析 JSON 文件的資訊，請參閱 步驟 1：確保您的端點已就緒可處理 Amazon SNS 訊息。

SignatureVersion 告訴您 Amazon SNS 用來產生訊息簽章的簽章版本。從簽章版本，您可以判定如何產生簽章的要求。對於 Amazon SNS 通知，Amazon SNS 目前支援簽章版本 1 和 2。本節提供使用簽章版本驗證簽章的步驟。

2. 取得 Amazon SNS 用於簽署訊息的 X509 憑證。SigningCertURL 值會指向 X509 憑證的位置，該憑證用於建立訊息的數位簽章。從這個位置讀擷取憑證。
3. 從憑證擷取公開金鑰。從 SigningCertURL 所指定憑證的公開金鑰，是用來驗證訊息的真偽和完整性。
4. 判定訊息類型。要簽署字串的格式依訊息類型而定，而這由 Type 值指定。
5. 建立要簽署的字串。要簽署的字串是訊息的特定名稱值組的新行字元分隔清單。每個名稱值組是以後面先接新行字元、然後是值，再以新行字元結束的名稱表示。名稱值組必須以字元組排序的順序列出。

   依照訊息類型而定，要簽署的字串必需具有下列名稱值組。

   Notification

   通知訊息必須包含下列名稱值組。

   ```
   Message
   MessageId
   Subject (if included in the message)
   Timestamp
   TopicArn
   Type
   ```

   以下範例是 Notification 之要簽署的字串。

   ```
   Message
   My Test Message
   MessageId
   4d4dc071-ddbf-465d-bba8-08f81c89da64
   Subject
   My subject
   Timestamp
   2019-01-31T04:37:04.321Z
   TopicArn
   arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
   Type
   ```

```
Notification
```

SubscriptionConfirmation 和 UnsubscribeConfirmation

SubscriptionConfirmation 和 UnsubscribeConfirmation 訊息必須包含下列名稱值組：

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

以下範例是 SubscriptionConfirmation 之要簽署的字串。

```
Message
My Test Message
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
https://sns.us-east-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-2:123456789012:s4-
MySNSTopic-1G1WEFCOXTC0P&Token=233...
Timestamp
2019-01-31T19:25:13.719Z
Token
233...
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

6. 從 Base64 格式解碼 Signature 的值。訊息傳遞 Signature 值的簽章，這解碼為 Base64。在比較簽章值和您已計算的簽章之前，確保您從 Base64 解碼 Signature 值，以便您使用相同格式比較值。
7. 產生 Amazon SNS 訊息所衍生的雜湊值。提交典型格式的 Amazon SNS 訊息至用於產生簽章的相同雜湊演算法。

   a. 如果 SignatureVersion 是1，使用 SHA1 作為雜湊算法。

b. 如果 SignatureVersion 是 2，使用 SHA256 作為雜湊算法。

8. 產生 Amazon SNS 訊息所插入的雜湊值。插入的雜湊值是使用公開鍵值 (從步驟 3) 的結果，以解密使用 Amazon SNS 訊息傳遞的簽章。
9. 驗證 Amazon SNS 訊息的真偽和完整性。比較衍生的雜湊值 (從步驟 7) 和插入的雜湊值 (從步驟 8)。如果值相同，則接收者可確定訊息在傳輸時未被修改，並且訊息必須源自 Amazon SNS。如果值不相同，則接收者不應予以信任。

剖析訊息格式

Amazon SNS 使用下列格式。

主題

- HTTP/HTTPS 標頭
- HTTP/HTTPS 訂閱確認 JSON 格式
- HTTP/HTTPS 通知 JSON 格式
- HTTP/HTTPS 取消訂閱確認 JSON 格式
- SetSubscriptionAttributes 交付政策 JSON 格式
- SetTopicAttributes 交付政策 JSON 格式

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.\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.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

HTTP/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.\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.\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。

擴散至 AWS 事件分叉管道

對於事件存檔和分析，Amazon SNS 現在建議使用與 Amazon 資料 Firehose 的原生整合。您可以訂閱 Firehose 交付串流至 SNS 主題，以便將通知傳送到存檔和分析端點，例如 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體、Amazon Redshift 表格、亞馬遜 OpenSearch 服務 (OpenSearch 服務) 等。將 Amazon SNS 與 Firehose 交付串流搭配使用是完全受管且無程式碼的解決方案，不需要您使用功能。AWS Lambda如需詳細資訊，請參閱 扇出到 Firehose 交付流。

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。

若要加速事件驅動應用程式的開發，您可以訂閱事件處理管線 (由 AWS 事件分叉管線 — 適用於 Amazon SNS 主題。AWS事件分叉管道是一套開放原始碼巢狀應用程式，根據 AWS Serverless Application Model (AWS 無伺服器應用程式模型，SAM)，您可以將該管道直接從 AWS 事件分叉管道

套件 (選擇 Show apps that create custom IAM roles or resource policies (顯示建立自訂 IAM 角色或資源政策的應用程式)) 部署到您的 AWS 帳戶。

對於 AWS 事件分叉管道使用案例，請參閱部署和測試 AWS 事件分叉管道範例應用程式。

主題

- AWS 事件分叉管道的運作原理
- 部署 AWS 事件分叉管道
- 部署和測試 AWS 事件分叉管道範例應用程式
- 訂閱 AWS Amazon SNS 主題的事件分叉管道

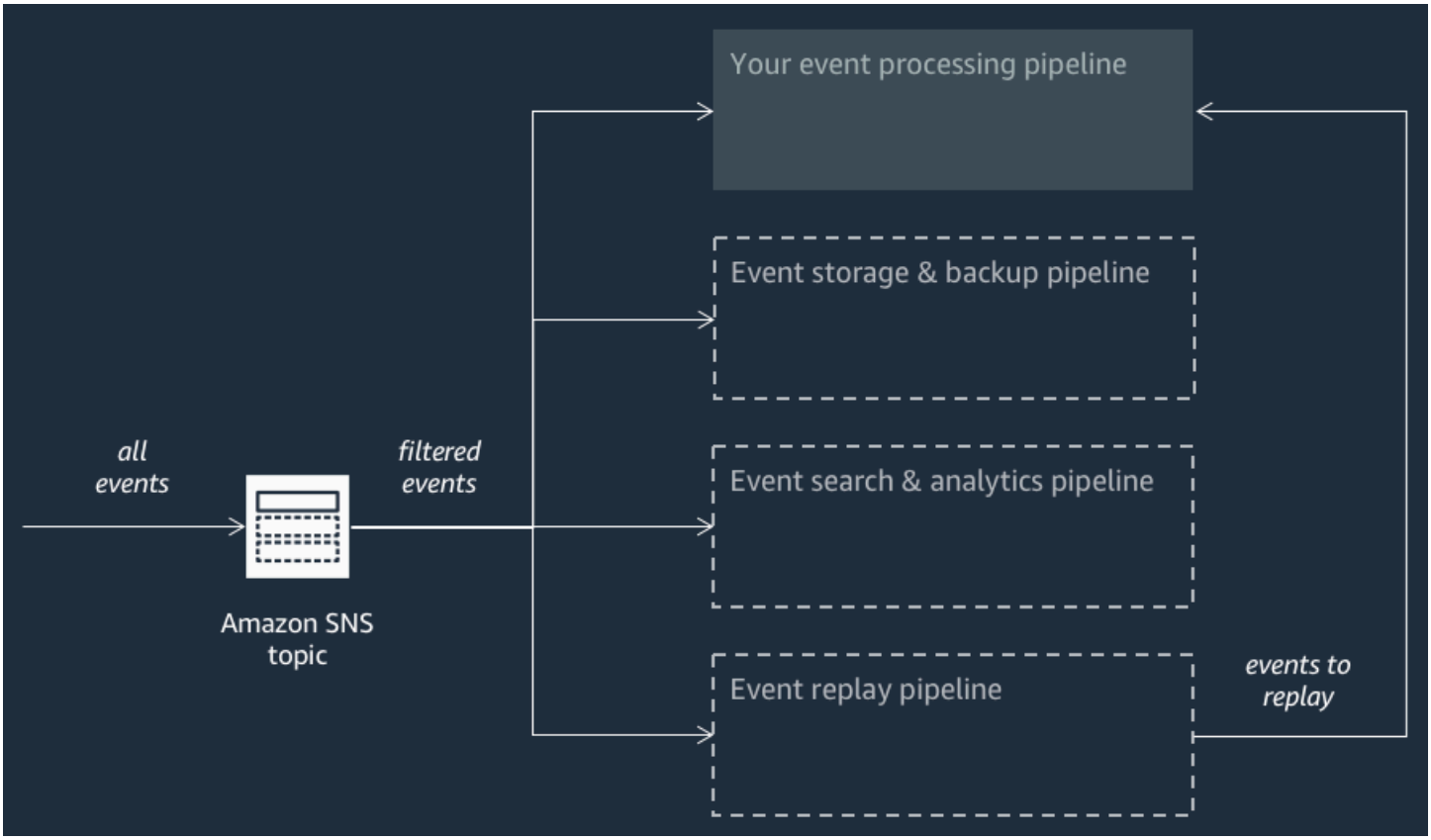
AWS 事件分叉管道的運作原理

AWS 事件分叉管線是無伺服器設計模式。不過，它也是一套根據 AWS SAM 的巢狀無伺服器應用程式 (可直接從 AWS Serverless Application Repository (AWS SAR) 部署到您的 AWS 帳戶 帳戶，以強化您的事件驅動型平台)。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的 AWS 應用程式。因應您的架構要求，您可以在 AWS SAR 上從 AWS 套件獨立部署任何管道。



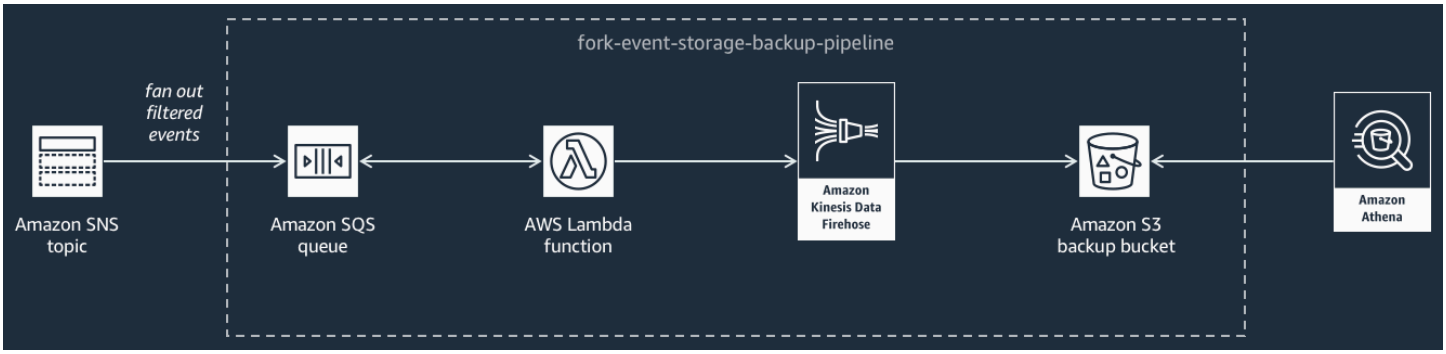
每個管道都訂閱同一個 Amazon SNS 主題，讓它本身可在事件發佈到主題時平行處理這些事件。每個管道各自獨立，而且可以設定自己的訂閱篩選政策。這可讓管道只處理它有感興趣的事件子集 (而不是發佈到該主題的所有事件)。

> **Note**
>
> 由於您沿著一般事件處理管道 (可能已經訂閱您的 Amazon SNS 主題) 設置三個 AWS 事件分叉管道，您不需要為了利用現有工作負載中的 AWS 事件分叉管道而變更目前訊息發佈者的任何部分。

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。

此管道由可緩衝 Amazon SNS 主題所交付事件的 Amazon SQS 佇列、自動輪詢佇列中的這些事件並將其推送至 Amazon Data Firehose 串流的AWS Lambda函數，以及可持久備份串流載入事件的 Amazon S3 儲存貯體。

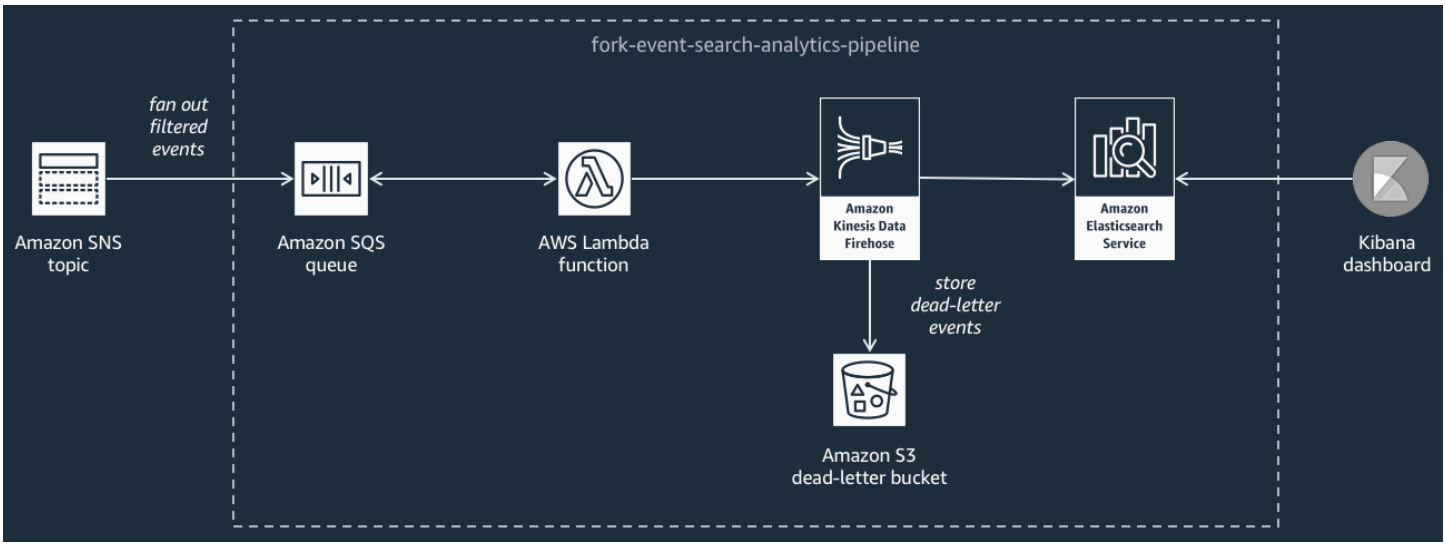


若要微調 Firehose 串流的行為，您可以將它設定為將事件載入到儲存貯體之前，先緩衝、轉換和壓縮事件。隨著事件載入，您可以透過 Amazon Athena 使用標準 SQL 查詢來查詢儲存貯體。您也可以將管道設定為重複使用現有的 Amazon S3 儲存貯體，或建立新的儲存貯體。

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。

此管道由可緩衝 Amazon SNS 主題所交付事件的 Amazon SQS 佇列、輪詢佇列中的事件並將事件推送至 Amazon Data Firehose 串流的AWS Lambda函數、用於為 Firehose 串流載入的事件編製索引的 Amazon OpenSearch 服務網域，以及儲存無法在搜尋網域中編製索引的無效字母事件的 Amazon S3 儲存貯體。



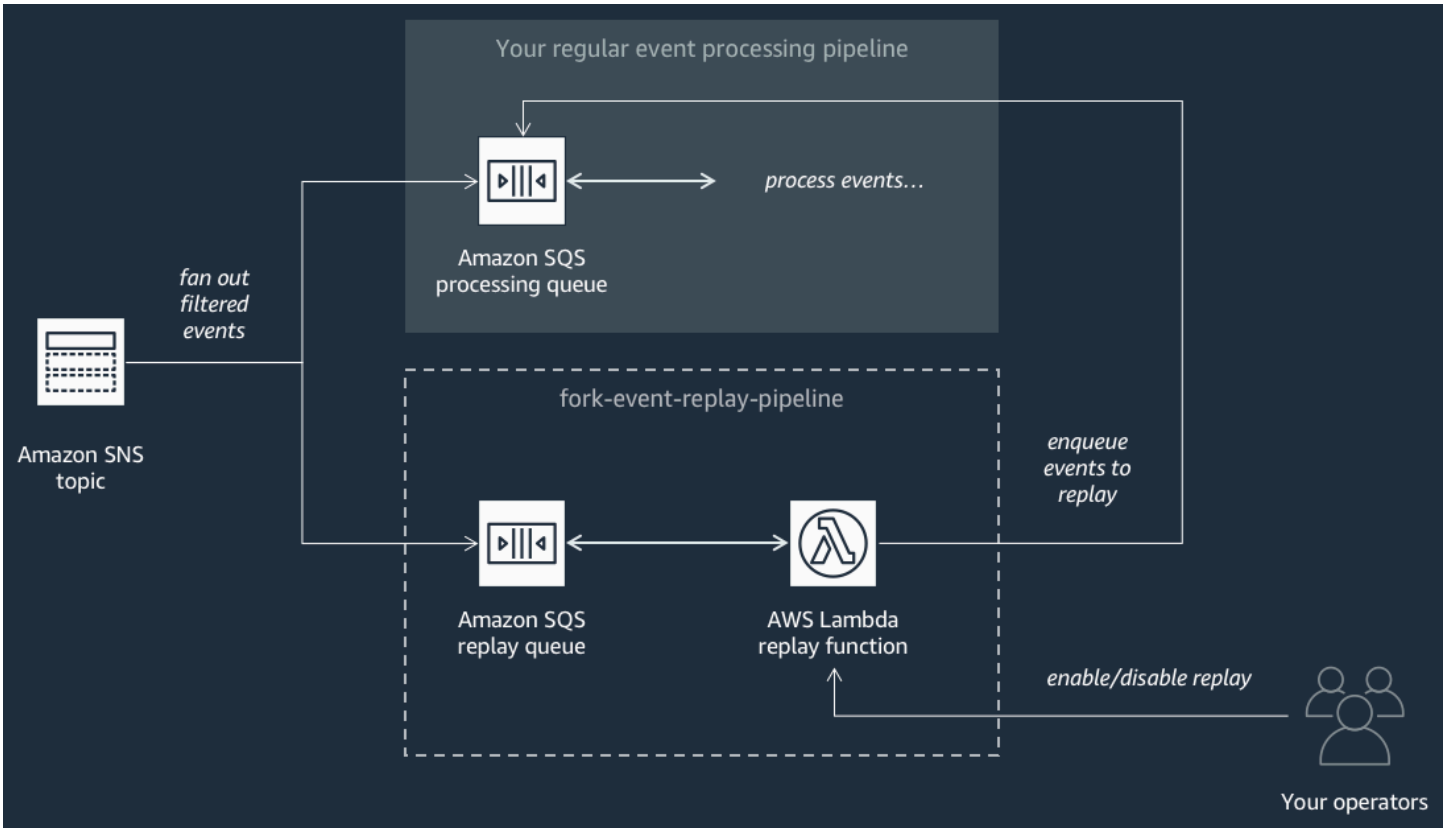
若要調整 Firehose 串流的事件緩衝、轉換和壓縮，您可以設定此管道。

您也可以設定管線是否應該重複使用您的 OpenSearch 網域中的現有網域，AWS 帳戶或為您建立新網域。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。

這個管道的組成包括緩衝 Amazon SNS 主題傳遞之事件的 Amazon SQS 佇列，還有輪詢佇列中的事件並將事件轉送到一般事件處理管道 (該管道也訂閱您的主題) 的 AWS Lambda 函數。



> **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 套件 (選擇 Show apps that create custom IAM roles or resource policies (顯示建立自訂 IAM 角色或資源政策的應用程式)) 是 AWS Serverless Application Repository 中一群可用的公有應用程式，您可以在其中使用 AWS Lambda 主控台來手動部署和進行測試。如需有關使用 AWS Lambda 主控台來部署管道的資訊，請參閱訂閱 AWS Amazon SNS 主題的事件分叉管道。

在生產情境下，建議您將 AWS 內嵌在整體應用程式的 AWS SAM 範本中。巢狀應用程式功能可讓您這樣做：請將資源 AWS::Serverless::Application 新增到 AWS SAM 範本，並參考巢狀應用程式的 AWS SAR ApplicationId 和 SemanticVersion。

例如，您可以將以下 YAML 片段新增到 AWS SAM 範本的 Resources 區段，就能使用事件儲存和備份管道當做巢狀應用程式。

```
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 片段中，TopicArn 參數參考 AWS::SNS::Topic 資源 MySNSTopic (定義在 AWS SAM 範本中的其他位置)。如需詳細資訊，請參閱AWS CloudFormation使用指南中的內建函數參考。

> **Note**
>
> AWS SAR 應用程式的 AWS Lambda 主控台頁面包含 Copy as SAM Resource (複製為 SAM 資源) 按鈕，可將 AWS SAR 應用程式巢狀化所需的 YAML 複製到剪貼簿。

部署和測試 AWS 事件分叉管道範例應用程式

若要加速事件驅動應用程式的開發，您可以訂閱事件處理管線 (由 AWS 事件分叉管線 — 適用於 Amazon SNS 主題。AWS事件分叉管道是一套開放原始碼巢狀應用程式，根據 AWS Serverless Application Model (AWS 無伺服器應用程式模型，SAM)，您可以將該管道直接從 AWS 事件分叉管道套件 (選擇 Show apps that create custom IAM roles or resource policies (顯示建立自訂 IAM 角色或資源政策的應用程式)) 部署到您的 AWS 帳戶。如需詳細資訊，請參閱 AWS 事件分叉管道的運作原理。

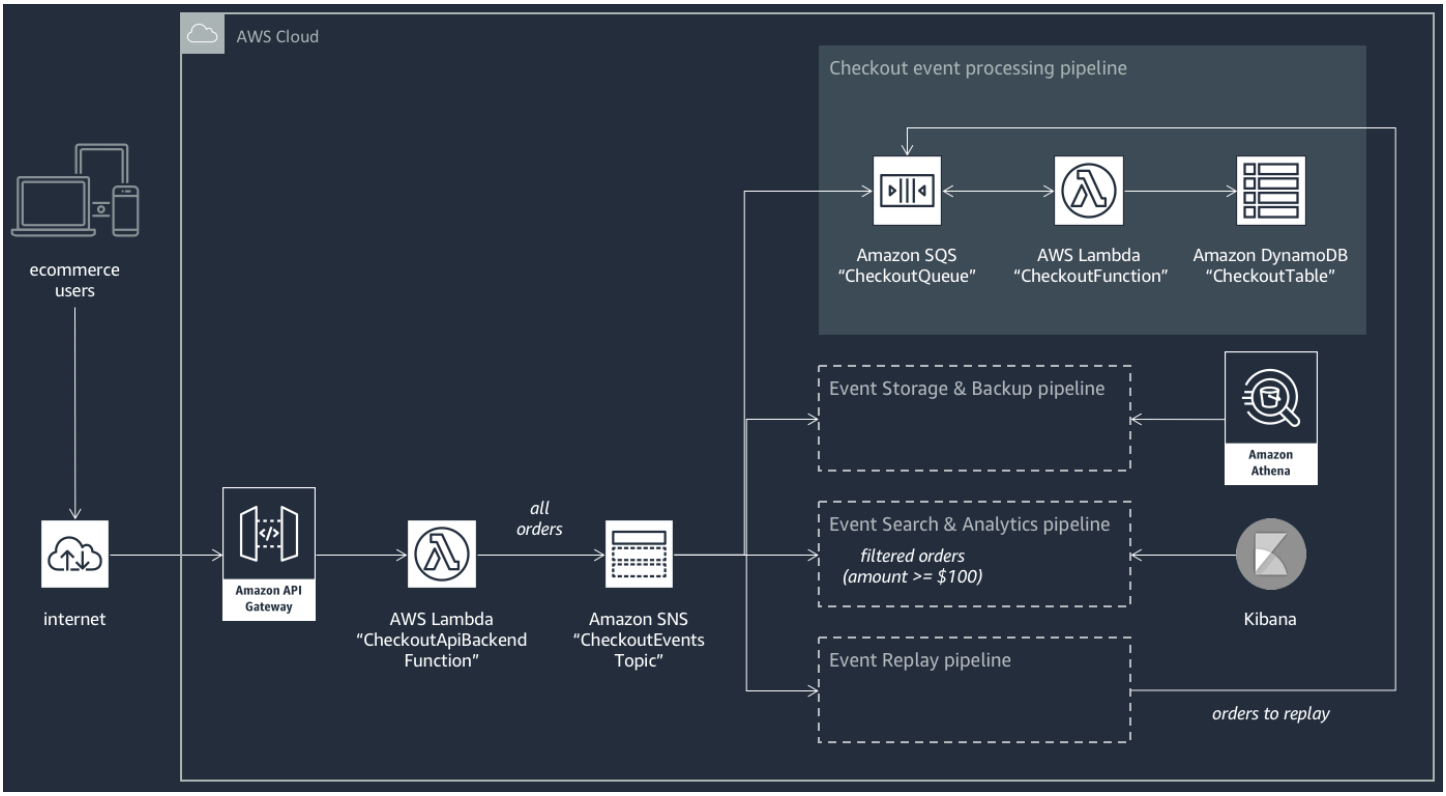
此頁面顯示如何使用 AWS Management Console 部署和測試 AWS 事件分叉管道範例應用程式。

> **Important**
>
> 為了避免在您完成部署 AWS 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「」刪除堆疊AWS CloudFormation主控台中的AWS CloudFormation使用者指南。

主題

範例 AWS 事件分叉管道使用案例

以下案例說明一個使用 AWS 的事件驅動型、無伺服器電子商務應用程式。您可以在中使用此示例電子商務應用程序，AWS Serverless Application Repository然後AWS 帳戶使用控制台將其部署到您的AWS Lambda控制台中，您可以在其中對其進行測試並檢查其源代碼 GitHub。



此電子商務應用程式會透過由 API Gateway 所託管和由 AWS Lambda 函數 `CheckoutApiBackendFunction` 所支援的 RESTful API，以取得買方的訂單。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。此管道有 Amazon SQS 佇列 `CheckoutQueue` 可緩衝所有收到的訂單、有名為 `CheckoutFunction` 的 AWS Lambda 函數可輪詢佇列來處理這些訂單，還有 DynamoDB 資料表 `CheckoutTable` 可安全地存放所有已提交的訂單。

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 在履行程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讓 AWS 事件分叉管道訂閱 `CheckoutEventsTopic` Amazon SNS 主題，而不需要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 AWS 管理，並且採用 AWS Key Management Service (AWS KMS) 技術。

  如需詳細資訊，請參閱 Amazon 資料 Firehose 開發人員指南中的「為目的地選擇 Amazon S3 」、「Amazon 資料 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 資料 Firehose 開發人員指南中的為您的目的地選擇 OpenSearch 服務。
- 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 ] }]
}
```

電子商務應用程式擁有者可以使用 AWS 模式，避免因為撰寫無差別邏輯來處理事件而產生的額外開發成本。她可以改為將 AWS 直接從 AWS Serverless Application Repository 部署到 AWS 帳戶 帳戶。

步驟 1：部署範例應用程式

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，部署只會更新先前已部署的 AWS CloudFormation 堆疊 (而不是建立新的堆疊)。

   b. (選擇性) 輸入下列其中一項LogLevel設定，以執行應用程式的 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 fork-example-ecommerce-*###############Lambda ########*式正在部署中的狀態。

在 Resources (資源) 區段中，AWS CloudFormation 會開始建立堆疊，並顯示每個資源為 CREATE_IN_PROGRESS 狀態。程序完成後，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：執行範例應用程式

1. 在 AWS Lambda 主控台的導覽面板上，選擇 Applications (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. 若要尋找類型為的資源 ApiGatewayRestApi，請依類型 (Type) 排序資源`ServerlessRestApi`，然後再展開資源。
   b. 會顯示「ApiGateway部署」和「ApiGateway階段」類型的兩個巢狀資源。
   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,
      "name": "John Doe",
      "email": "john.doe@example.com"
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
      "currency": "usd",
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. 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：驗證範例應用程式及其管道的執行情形

步驟 1：驗證範例結帳管道的執行情形

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情形

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情形

1. 登入服OpenSearch 務主控台。
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 serverl-analyt-***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情形

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 [我的應用程式 fork-example-ecommerce-******... 重播 ReplayQueue-*123ABCD4E5F6*] 對話方塊中，選擇 [郵件輪詢]。
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復原

步驟 1：啟用模擬的問題和傳送第二個 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-我的****-CheckoutFunction-*AB* CDEF... 頁面的 [環境變數] 區段中，將 BUG_ENABYED 變數設定為 true，然後選擇 [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,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
    "payment": {
       "id": 3311,
       "amount": 75.00,
       "currency": "usd",
       "method": "credit",
       "card-network": "mastercard",
       "card-number": "1234 5678 9012 3456",
       "card-expiry": "12/2025",
       "card-owner": "Marcia Oliveira",
       "card-cvv": "321"
```

```
    },
    "shipping": {
       "id": 9900,
       "time": 20,
       "unit": "days",
       "method": "plane"
    },
    "items": [{
       "id": 9993,
       "product": 3120,
       "name": "Hockey Stick",
       "quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. 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：驗證模擬的資料損毀

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，有些標記為 CORRUPTED! (毀損！)

步驟 3：停用模擬的問題

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-我的***####***-CheckoutFunction-***AB*** CDEF... 頁面的 [環境變數] 區段中，將 BUG_ENABYED 變數設定為 false，然後選擇 [儲存]。

步驟 4：啟用重播以從問題中復原

1. 在 AWS Lambda 主控台的導覽面板上，選擇 Functions (函數)。
2. 搜尋 `serverlessrepo-fork-example` 並選擇 `ReplayFunction`。
3. 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 `ReplayFunction` 的 Amazon SQS 事件來源觸發。

訂閱 AWS Amazon SNS 主題的事件分叉管道

若要加速事件驅動應用程式的開發，您可以訂閱事件處理管線 (由 AWS 事件分叉管線 — 適用於 Amazon SNS 主題。AWS事件分叉管道是一套開放原始碼巢狀應用程式，根據 AWS Serverless Application Model (AWS 無伺服器應用程式模型，SAM)，您可以將該管道直接從 AWS 事件分叉管道套件 (選擇 Show apps that create custom IAM roles or resource policies (顯示建立自訂 IAM 角色或資源政策的應用程式)) 部署到您的 AWS 帳戶。如需詳細資訊，請參閱 AWS 事件分叉管道的運作原理。

本節顯示如何使用 AWS Management Console來部署管道，然後訂閱 AWS Amazon SNS 主題的事件分叉管線。開始之前，請建立 Amazon SNS 主題。

若要刪除組成管線的資源，請在AWS Lambda主控台的 [應用程式] 頁面上尋找管線，展開 SAM 範本區段，選擇 [CloudFormation堆疊]，然後選擇 [其他動作] > [刪除堆疊]。

主題

- 部署及訂閱事件儲存和備份管道

- 部署及訂閱事件搜尋和分析管道
- 部署及訂閱事件重播管道

部署及訂閱事件儲存和備份管道

對於事件存檔和分析，Amazon SNS 現在建議使用與 Amazon 資料 Firehose 的原生整合。您可以訂閱 Firehose 交付串流至 SNS 主題，以便將通知傳送到存檔和分析端點，例如 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體、Amazon Redshift 表格、亞馬遜 OpenSearch 服務 (OpenSearch 服務) 等。將 Amazon SNS 與 Firehose 交付串流搭配使用是完全受管且無程式碼的解決方案，不需要您使用功能。AWS Lambda如需詳細資訊，請參閱 扇出到 Firehose 交付流。

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

如需將 S3 儲存貯體設定為目的地的串流的詳細資訊，請參閱 Amazon 資料 Firehose API 參考*S3DestinationConfiguration*中的。

如需有關轉換事件以及設定事件緩衝、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-管線頁面上，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，部署只會更新先前已部署的 AWS CloudFormation 堆疊 (而不是建立新的堆疊)。

   b. (選擇性) 在中 BucketArn，輸入要載入內送事件的 S3 儲存貯體的 ARN。如果您不輸入值，則您的 AWS 帳戶中會建立新的 S3 儲存貯體。

   c. (選擇性) 在中 DataTransformationFunctionArn，輸入 Lambda 函數的 ARN，以便透過該函數轉換傳入事件。如果您不輸入值，資料轉換會停用。

   d. (選擇性) 輸入下列其中一項LogLevel設定，以執行應用程式的 Lambda 函數：

      - `DEBUG`
      - `ERROR`
      - `INFO` (default)
      - `WARNING`

   e. 在中 TopicArn，輸入要訂閱此分叉管道執行個體的 Amazon SNS 主題的 ARN。

   f. (選擇性) 對於StreamBufferingIntervalInSeconds和 StreamBufferingSizeInMB，請輸入用於設定內送事件緩衝的值。如果您不輸入任何值，則會使用 300 秒和 5 MB。

   g. (選擇性) 輸入下列其中一個StreamCompressionFormat設定以壓縮傳入事件：

      - `GZIP`
      - `SNAPPY`
      - `UNCOMPRESSED` (default)
      - `ZIP`

   h. (選擇性) 對於 StreamPrefix，輸入字串前置詞，以命名 S3 備份儲存貯體中存放的檔案。如果您不輸入值，則不會使用字首。

   i. (選擇性) 在中 SubscriptionFilterPolicy，輸入 Amazon SNS 訂閱篩選政策 (採用 JSON 格式)，以用於篩選傳入事件。篩選原則會決定哪些事件會在 OpenSearch Service 索引中建立索引。如果您不輸入值，則不會使用篩選 (所有事件都編製索引)。

j. (選擇性) 在中 SubscriptionFilterPolicyScope，輸入字串，`MessageBody`或`MessageAttributes`啟用以承載為基礎或屬性型的訊息篩選。

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 Resources (資源) 區段中，AWS CloudFormation 會開始建立堆疊，並顯示每個資源為 CREATE_IN_PROGRESS 狀態。程序完成後，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 主題的訊息會自動存放在由事件儲存和備份管道佈建的 S3 備份儲存貯體。

部署及訂閱事件搜尋和分析管道

對於事件存檔和分析，Amazon SNS 現在建議使用與 Amazon 資料 Firehose 的原生整合。您可以訂閱 Firehose 交付串流至 SNS 主題，以便將通知傳送到存檔和分析端點，例如 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體、Amazon Redshift 表格、亞馬遜 OpenSearch 服務 (OpenSearch 服務) 等。將 Amazon SNS 與 Firehose 交付串流搭配使用是完全受管且無程式碼的解決方案，不需要您使用功能。AWS Lambda如需詳細資訊，請參閱 扇出到 Firehose 交付流。

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 服務域
- Amazon S3 無法投遞儲存貯體

如需有關將索引設定為目的地的串流的詳細資訊，請參閱 Amazon 資料 Firehose API 參考*ElasticsearchDestinationConfiguration*中的。

如需有關轉換事件以及設定事件緩衝、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-管線頁面上，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，部署只會更新先前已部署的 AWS CloudFormation 堆疊 (而不是建立新的堆疊)。

   b. (選擇性) 在中 DataTransformationFunctionArn，輸入用於轉換傳入事件之 Lambda 函數的 ARN。如果您不輸入值，資料轉換會停用。
   c. (選擇性) 輸入下列其中一項LogLevel設定，以執行應用程式的 Lambda 函數：
      - `DEBUG`
      - `ERROR`
      - `INFO` (default)
      - `WARNING`
   d. (選擇性) 在中 SearchDomainArn，輸入 OpenSearch Service 網域的 ARN，此叢集會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. 在中 TopicArn，輸入要訂閱此分叉管道執行個體的 Amazon SNS 主題的 ARN。
   f. 在中 SearchIndexName，輸入用於事件搜尋和分析的 OpenSearch 服務索引名稱。

> **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
> - 不能包含冒號（來自 OpenSearch 服務 7.0）

g. (選擇性) 針對「OpenSearch 服務」索引的循環期間輸入下列其中一個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. 在中 SearchTypeName，輸入用於組織索引中事件的 OpenSearch Service 類型名稱。

> **Note**
>
> - OpenSearch 服務類型名稱可以包含任何字元 (空位元組除外)，但不能以_.
> - 對於 OpenSearch 服務 6.x，每個索引只能有一個類型。如果您為已有其他類型的現有索引指定新類型，Firehose 會傳回執行階段錯誤。

i. (選擇性) 對於StreamBufferingIntervalInSeconds和 StreamBufferingSizeInMB，請輸入用於設定內送事件緩衝的值。如果您不輸入任何值，則會使用 300 秒和 5 MB。

j. (選擇性) 輸入下列其中一個StreamCompressionFormat設定以壓縮傳入事件：

- GZIP
- SNAPPY
- UNCOMPRESSED (default)
- ZIP

k. (選擇性) 對於 StreamPrefix，輸入字串前置詞，以命名 S3 無效字母儲存貯體中存放的檔案。如果您不輸入值，則不會使用字首。

l. (選擇性) 針對 Firehose 無法在 OpenSearch 服務索引中索引事件的情況 StreamRetryDurationInSecons，輸入重試持續時間。如果您不輸入值，則會使用 300 秒。

m. (選擇性) 在中 SubscriptionFilterPolicy，輸入 Amazon SNS 訂閱篩選政策 (採用 JSON 格式)，以用於篩選傳入事件。篩選原則會決定哪些事件會在 OpenSearch Service 索引中建立索引。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*my-app-search*頁面的部署狀態上，Lambda 會顯示您的應用程式正在部署中的狀態。

在 Resources (資源) 區段中，AWS CloudFormation 會開始建立堆疊，並顯示每個資源為 CREATE_IN_PROGRESS 狀態。程序完成後，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 主題的訊息會自動在事件搜尋和分析管道佈建的 OpenSearch 服務索引中編製索引。如果管道無法將事件編製索引，則會將事件存放在 S3 無法投遞儲存貯體中。

部署及訂閱事件重播管道

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 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，部署只會更新先前已部署的 AWS CloudFormation 堆疊 (而不是建立新的堆疊)。

   b. (選擇性) 輸入下列其中一項LogLevel設定，以執行應用程式的 Lambda 函數：
      - `DEBUG`
      - `ERROR`
      - `INFO` (default)
      - `WARNING`
   c. (選擇性) 對於 ReplayQueueRetentionPeriodInSeconds，輸入 Amazon SQS 重新顯示佇列保留訊息的時間長度 (以秒為單位)。如果您不輸入值，則會使用 1,209,600 秒 (14 天)。
   d. 在中 TopicArn，輸入要訂閱此分叉管道執行個體的 Amazon SNS 主題的 ARN。
   e. 在中 DestinationQueueName，輸入 Amazon SQS 佇列的名稱，Lambda 重新顯示函數會將訊息轉寄至該佇列。
   f. (選擇性) 在中 SubscriptionFilterPolicy，輸入 Amazon SNS 訂閱篩選政策 (採用 JSON 格式)，以用於篩選傳入事件。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*`my-app-replay`*頁面的部署狀態上，Lambda 會顯示您的應用程式正在部署中的狀態。

在 Resources (資源) 區段中，AWS CloudFormation 會開始建立堆疊，並顯示每個資源為 CREATE_IN_PROGRESS 狀態。程序完成後，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。

主題

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。如果您要使用 EventBridge 排程器 SDK、AWS CLI 或 AWS CloudFormation 建

立排程，您必須具備現有的執行角色，授與 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。	

頻率	執行此作業...	
	• 若要使用 Rate 運算式定義排程，請選擇 Rate 排程，然後輸入 Rate 運算式。 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。 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. 選擇下一步。
7. 在選取目標頁面上，選擇 EventBridge 排程器調用的 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。
將事件儲存在您建立排程所使用的同一個 AWS 帳戶	a. 選擇在目前的 AWS 帳戶選取 Amazon SQS 佇列作為 DLQ。 b. 選擇 Amazon SQS 佇列的 Amazon Resource Name (ARN)。
將事件儲存在建立排程所用帳戶以外的 AWS 帳戶	a. 選擇在其他 AWS 帳戶指定 Amazon SQS 佇列作為 DLQ。

無效字母佇列 (DLQ) 選項	執行此作業...	
	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

使用 Amazon SNS 進行應用程式至人員 (A2P) 訊息

本節提供對於行動應用程式、行動電話號碼和電子郵件地址之類的訂閱者使用使用者通知的 Amazon SNS 相關資訊。

主題

行動裝置簡訊 (SMS)

您可以使用 Amazon SNS 傳送文字訊息或簡訊至啟用簡訊功能的裝置。您可以直接傳送訊息至一組電話號碼，或一次傳送一則訊息至多組電話號碼，只要訂閱那些電話號碼到主題並且傳送您的訊息到該主題即可。

您可以為您的 AWS 帳戶設定簡訊喜好設定，為您的使用案例和預算量身打造您的簡訊傳遞。例如，您可以選擇是否針對成本或可靠的傳遞將訊息最佳化。您也可以指定個別訊息傳遞的費用配額，以及您的 AWS 帳戶 帳戶每月費用配額。

依當地法律和法規 (例如美國和加拿大) 的要求，簡訊收件人可以停止接收，表示他們選擇停止從您的 AWS 帳戶 帳戶接收簡訊。在收件人停止接收之後，您可以透過限制再次加入電話號碼，以便您可以繼續傳送訊息至該電話號碼。

Amazon SNS 支援幾個地區的簡訊，並且您可以傳送訊息至超過 200 個國家和地區。如需更多詳細資訊，請參閱 支援的國家和區域。

主題

- 支援的國家和區域
- 簡訊最佳實務

簡訊沙盒

當您開始使用 Amazon SNS 傳送簡訊時，AWS 帳戶位於簡訊沙盒。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，但有下列限制：

- 您只能傳送簡訊至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。
- 您必須在驗證或上次驗證後 24 小時以上，才能刪除目的地電話號碼。

當您的帳戶移出沙盒時，系統會移除這些限制，而且您可以傳送簡訊給任何收件者。

主題

- 在簡訊沙盒中新增和驗證電話號碼
- 從簡訊沙盒中刪除電話號碼
- 移出簡訊沙盒

在簡訊沙盒中新增和驗證電話號碼

若要在您的 AWS 帳戶位於 SMS 沙箱中時開始傳送 SMS 訊息，請建立起始身分、新增目標電話號碼，然後進行驗證。

> **Note**
>
> 如同不在簡訊沙盒中的帳戶一樣，需要來源身分才能傳送簡訊訊息給某些國家或區域的收件人。如需詳細資訊，請參閱 支援的國家和區域。
> 原始識別碼包括寄件者 ID 和不同類型的來源號碼。若要檢視您現有的原始編號，請在 Amazon SNS 主控台中，選擇來源號碼。目前，寄件者 ID 不會顯示在此清單中。

新增及驗證目的地電話號碼

1. 登入 Amazon SNS 主控台。

2. 建立電話號碼的來源身分。
3. 在主控台功能表中，選擇支援簡訊的AWS 區域。
4. 在導覽窗格中，選擇 Text messaging (SMS) (簡訊 (SMS))。
5. 在 Mobile message (SMS) (行動裝置簡訊 (SMS)) 頁面，在Sandbox destination phone numbers (沙盒目的地電話號碼) 中，選擇 Add phone number (新增電話號碼)。
6. 在 Destination details (目的地詳細資料)下方輸入國家/地區代碼和電話號碼、指定驗證訊息的使用語言，然後選擇 Add phone number (新增電話號碼)。

   Amazon SNS 會傳送一次性密碼 (OTP) 到目的地電話號碼。如果目的地電話號碼未能在 15 分鐘內收到一次性密碼，請選擇 Resend verification code (重新傳送驗證碼)。您可以每 24 小時將一次性密碼發送至同一個目的地電話號碼最多五次。
7. 在 Verification code (驗證碼) 方塊中，輸入傳送至目的地電話號碼的一次性密碼，然後選擇 Verify phone number (驗證電話號碼)。

   目的地電話號碼及其驗證狀態會顯示在 Sandbox destination phone numbers (沙盒目的地電話號碼) 區段。如果驗證狀態為 Pending (待定)，則驗證失敗。例如，如果您沒有在電話號碼中輸入國碼，就可能發生這種情況。您只能在驗證或上次驗證後超過 24 小時或更長時間後，刪除待處理或已驗證的目的地電話號碼。
8. 在您要使用此目的地電話號碼的每個區域重複這些步驟。

疑難排解未收到 OTP 文字

解決可能導致電話號碼無法接收 OTP 簡訊的常見問題。

- Amazon SNS SMS 消費限制：如果您 AWS 帳戶 已超過傳送簡訊的支出限制，則在增加限制或解決帳單問題之前，其他訊息 (包括 OTP 文字) 可能無法傳送。
- 未選擇接收 SMS 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短訊，短碼通常用於 OTP 簡訊。如果收件人的電話號碼沒有選擇加入，他們將不會收到 OTP 短信。
- 運營商限制或過濾：某些移動運營商可能具有限制或過濾機制，以防止某些類型的 SMS 消息（包括 OTP 文本）發送。這可能是由於運營商實施的安全策略或反垃圾郵件措施。
- 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，OTP 文本將不會發送。
- 網絡問題：臨時的網絡問題或中斷可能會阻止 SMS 消息（包括 OTP 文本）傳遞到收件人的手機。
- 延遲傳送：在某些情況下，SMS 訊息可能會因為網路擁塞或其他因素而導致傳送延遲。OTP 文本最終可能會發送，但可能會延遲超出預期的時間範圍。

- 帳戶暫停或終止：如果您的帳戶發生問題 AWS 帳戶，例如未付款或違反服務 AWS 條款，Amazon SNS 簡訊功能 (包括 OTP 文字) 可能會暫停或終止。

從簡訊沙盒中刪除電話號碼

您可以從簡訊沙盒刪除待處理或驗證的目的地電話號碼。

從簡訊沙盒安全執行程序中刪除目的地電話號碼

1. 等候 24 小時驗證電話號碼，或在上次驗證嘗試後 24 小時內完成。
2. 登入 Amazon SNS 主控台。
3. 在主控台功能表中，選擇 AWS 支援簡訊的區域，您已在其中新增目的地電話號碼。
4. 在導覽窗格中，選擇 Text messaging (SMS) (簡訊 (SMS))。
5. 在 Mobile text messaging (SMS) (行動裝置簡訊 (SMS)) 頁面，在 Sandbox destination phone numbers (沙盒目的地電話號碼) 下方選擇要刪除的電話號碼，然後選擇 Delete phone number (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)。

   如果在您驗證或嘗試驗證目的地電話號碼後已超過 24 小時或以上，則會刪除該號碼，Amazon SNS 會更新您的目的地電話號碼列表。
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。

移出簡訊沙盒

您必須先新增、驗證和測試目的地電話號碼，才能將您移 AWS 帳戶 出 SMS 沙箱。然後，您必須使用建立案例 AWS Support。

要求您的 AWS 帳戶移出 SMS 沙箱

1. 驗證電話號碼
   a. 當您在 AWS 帳戶 簡訊沙箱中時，開啟 Amazon SNS 主控台。
   b. 在功能窗格的 [行動裝置] 下，選擇 [簡訊 (SMS)]。
   c. 在沙箱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發送和接收消息。
2. 測試短信發布

- 確認您可以向至少一個已驗證的電話號碼傳送和接收訊息。如需如何發佈 SMS 訊息的詳細指示，請參閱發佈至行動電話。

3. 啟動沙箱編輯

   - 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 中心，並在選取「增加服務配額」選項的情況下自動建立支援案例。

4. 填寫表格

   - 在「增加服務配額」下的支援表單中，執行下列動作：

     i. 選擇選擇 SNS 簡訊作為服務。

     ii. 提供您要從中發送 SMS 消息的網站 URL 或應用程序名稱。

     iii. 指定您要傳送的訊息類型：「一次性密碼」、「促銷」或「交易」。

     iv. 選擇您將AWS 區域從中發送 SMS 消息的。

     v. 列出您計劃傳送簡訊的國家或地區。

     vi. 說明您的客戶如何選擇接收訊息。

     vii. 包括您打算使用的任何消息模板。

5. 指定配額和地區

   - 在 Requests (請求) 下，執行下列動作：

     i. 選擇您AWS 區域要移動的位置 AWS 帳戶。

     ii. 選擇資源類型的一般限制。

     iii. 選擇退出 SMS 沙箱以獲取配額。

     iv. (選擇性) 若要要求其他增加或其他調整，請選擇 [新增其他要求] 並指定必要的詳細資訊。

     v. 在新配額值中，輸入您要求的美元限制。

6. 其他細節

   a. 在案例說明中，提供與您的要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」，將您的請求傳送至 AWS Support。

該 AWS Support 團隊會在 24 小時內對您的請求進行初步響應。

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。

簡訊的來源身分

當您使用 Amazon SNS 傳送簡訊時，您可以使用下列類型的來源身分讓接收者了解您的身分：

- 寄件者 ID
- 來源號碼

> **Note**
>
> 在目前不支援 Amazon Pinpoint 的區域中，可使用 Amazon SNS 簡訊。如果您在歐洲 (斯德哥爾摩)、中東 (巴林)、歐洲 (巴黎)、南美洲 (聖保羅) 或美國西部 (加州北部)，請開啟美國東部 (維吉尼亞北部) 區域的 Amazon Pinpoint 主控台註冊您的 10DLC 公司和廣告活動，但是請勿要求一個 10DLC 號碼。請改用 AWSService Quotas 主控台來建立提高服務配額案例，同時要求該區域的 10DLC 號碼。若要進一步了解如何要求原始身分，請參閱 使用 Amazon SNS 請求簡訊支援。

寄件者 ID

寄件者 ID 是識別簡訊寄件者的字母名稱。當您使用寄件者 ID 傳送簡訊，且收件者位於支援寄件者 ID 驗證的區域時，您的寄件者 ID 會顯示在收件者的裝置上，而不是電話號碼。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。有關支援或需要寄件者 ID 的國家和區域的完整清單，請參閱支援的國家和區域。

使用寄件者 ID 無需額外費用。不過，寄件者 ID 驗證的支援和要求會有所不同。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。

> **Important**
>
> AWS 禁止簡訊詐騙，其中寄件者 ID 會被用來模擬其他人員、公司或產品。只能使用代表您擁有的品牌或商標的寄件者 ID。

優點

寄件者 ID 提供有關訊息寄件者的較多資訊給收件人。使用寄件者 ID 來建立品牌身分，比使用短碼或長碼更輕鬆。使用寄件者 ID 無需額外費用。

缺點

並非所有國家或區域都一致支援和要求寄件者 ID 身分驗證。有些主要市場 (包括加拿大、中國和美國) 不支援寄件者 ID。在某些區域，您的寄件者 ID 必須事先獲得監管機構核准，才能使用。

寄件者 ID 註冊 (依國家)

您必須開啟支援 AWS 的案例，才能註冊 SMS 訊息的寄件者識別碼。提出支援案例之後，AWS 將共用其他所需文件。您還必須為註冊寄件者識別碼的適當國家/地區提供以下資訊。

國名	訊息類型	格式限制和要求	註冊要求
澳洲 (AU)	交易與促銷	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元 • 寄件者 ID 必須是發送簡訊公司的品牌名稱	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司網址 (可連接到您的應用程式或公司網站)

國名	訊息類型	格式限制和要求	註冊要求

			• 預估每月數量
			• 使用案例的說明、訊息的目的
			• 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明
			• 公司的官方業務/貿易授權號碼或增值稅 #
			• 您計劃傳送的訊息範本
			• 商業登記證 其中包括但不限於：
			  • 澳洲企業編號 (ABN)
			  • 澳洲公司編號 (ACN)
			  • 澳洲註冊機構編號 (ARBN)
			  • 原住民公司編號 (ICN)
			• 授權書 (LOA)

國名	訊息類型	格式限制和要求	註冊要求
白俄羅斯 (BY)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元 • 寄件者 ID 必須是發送簡訊公司的品牌名稱	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 公司的官方業務/貿易授權號碼或增值稅 # • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
中國 (CN) 中國不需要寄件者 ID 註冊，但確實需要使用前面加上本文簽章的內容/範本註冊 (例如 [Amazon])。	不允許使用下列關鍵字： • 法輪功 • SB • 天安門廣場 以下類別的訊息： • 信用卡 • 數位支付 (包括加密貨幣) • 詐騙流量網址 (網路釣魚或垃圾郵件) • 賭博 • 不當內容 (成人、暴力、毒品、酒精) • 貸款 • 整形外科 • 政治 • 宗教 • 股票交易 • 虛擬貨幣 方括號內的簽章必須加在 SMS 訊息本文前面。若要順利傳送至中國，您必須在訊息內容範本中註冊訊息簽章。此訊息簽章必須放在每個要傳送的 SMS 的訊息內容前	• 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 公司名稱 • 公司地址 • 州/省 • Country • 公司電話號碼 • 公司網站 • 預估每月數量 • 訊息類型：促銷/交易 • 使用案例說明 • 您要註冊的範本 (發送的簡訊不得偏離提供的範本，以確保合規性和成功傳送)。例如，[CompanyName] (公司名稱) 您的一次性密碼為 {OTP}。此代碼將在 10 分鐘後過期。 • 確認您不會以交易訊息類型的形式傳送促銷內容 • 訊息簽章。請參閱簽章格式限制和要求。

國名	訊息類型	格式限制和要求	註冊要求
	面。如未在 SMS 訊息本文前面加上註冊簽章，可能會導致 SMS 遭到封鎖或過濾。簽章必須放在方括號內附加於 SMS 訊息本文前面。		

國名	訊息類型	格式限制和要求	註冊要求
埃及 (EG)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 貴公司在埃及有業務實體/辦公室嗎？還是非埃及公司傳送到埃及的訊息？ • 使用案例涵蓋此寄件者 ID 要傳送的所有訊息的書面確認 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係

國名	訊息類型	格式限制和要求	註冊要求
			• 您計劃傳送的訊息範本
印度 (IN)	僅傳送交易式訊息	• 英數字元 • 最多 6 個字元。 • 沒有空格 • 無特殊字元	請參閱印度的寄件者 ID 註冊要求。

國名	訊息類型	格式限制和要求	註冊要求
約旦 (JO)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 商業註冊憑證 • 您計劃發送的訊息類型 (例如 OTP，警示) • 使用案例說明此寄件者 ID 要傳送的所有訊息的書面確認 • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
科威特 (KW)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 您計劃發送的訊息類型 (例如 OTP，警示) • 使用案例說明此寄件者 ID 要傳送的所有訊息的書面確認 • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
菲律賓 (PH)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 您計劃發送的訊息類型 (例如 OTP，警示) • 使用案例說明此寄件者 ID 要傳送的所有訊息的書面確認 • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
卡達 (QA)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 發送的 SMS 類型 • (交易/促銷/OTP) 請注意，只有交易/OTP 訊息才能與目的地到卡達的寄件者 ID 一起使用，以便符合要求。 • 使用案例說明此寄件者 ID 要傳送的所有訊息的書面確認

國名	訊息類型	格式限制和要求	註冊要求
			• 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
俄羅斯 (RU)	僅傳送交易式訊息	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 稅務 ID 或證照號碼 • 商業註冊憑證 • 聯絡電子郵件地址 • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 確認此使用案例將套用至從此帳戶傳送至俄羅斯的所有訊息 • 確認非交易訊息必須使用個別的寄件者 ID 來傳送

國名	訊息類型	格式限制和要求	註冊要求
			• 260 USD/月費請確認您核准此重複月費：是/否 • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
沙烏地阿拉伯 (SA)	每個寄件者 ID 必須僅用於交易或促銷。單一寄件者 ID 無法同時用於這兩種類型的流量。如果要傳送 OTP 或 2FA 流量，寄件者 ID 必須僅用於此目的。促銷寄件人 ID 將受到沙烏地阿拉伯「請勿打擾」(DND) 清單的約束。	• 促銷寄件者 ID 長度：2 - 8 個字元，寄件者 ID 前面加上「-AD」 • 交易寄件者 ID 長度：2 - 11 個字元 • 寄件者 ID 必須代表寄件者的品牌身分 • 至少包含一個英文字母 • 請勿使用 ASCII 特殊字元 (例如，#、@) • 您可以包括大寫和小寫字母，以及數字 0 - 9	對沙烏地阿拉伯寄件人 ID 註冊的支援僅適用於國際公司。我們目前不支援沙烏地阿拉伯本地公司註冊寄件人 ID • 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 您計劃傳送的訊息範本。 • 此寄件者 ID 會用於交易或促銷內容嗎？

國名	訊息類型	格式限制和要求	註冊要求
			• 確認非交易訊息必須使用個別的寄件者 ID 來傳送 • 如果要傳送 2FA 或 OTP 流量，確認寄件者 ID 將僅用於此目的
新加坡 (SG)	僅傳送交易式訊息	• 最多 11 個字元。 • 沒有空格 • 無特殊字元	請參閱新加坡的寄件者 ID 註冊要求。

國名	訊息類型	格式限制和要求	註冊要求
斯里蘭卡 (LK)	沒有限制或特殊要求	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司類型 (本地/國際) • 公司網址 (可連接到您的應用程式或公司網站) • 使用案例的說明、訊息的目的 • 您計劃傳送的訊息範本 • 此寄件者 ID 會用於交易或促銷內容嗎？ • 確認非交易訊息必須使用個別的寄件者 ID 來傳送 • 如果要傳送 2FA 或 OTP 流量，確認寄件者 ID 將僅用於此目的

國名	訊息類型	格式限制和要求	註冊要求
泰國 (TH)	沒有限制或特殊要求	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 發送的 SMS 簡訊類型 (交易/促銷/OTP) • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
土耳其 (TR)	沒有限制或特殊要求	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司網址 (可連接到您的應用程式或公司網站) • 如果您的公司對土耳其而言是本地公司或國際公司 • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 發送的 SMS 簡訊類型 (交易/促銷/OTP) • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
烏克蘭 (UA)	沒有限制或特殊要求	• 英數字元 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司網址 (可連接到您的應用程式或公司網站) • 增值稅號碼 • 本地或國際公司 • 預估每月數量 • 使用案例的說明、訊息的目的 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 發送的 SMS 簡訊類型 (交易/促銷/OTP) • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
阿拉伯聯合大公國 (AE)	僅傳送交易式訊息	• 英數字元 • 不允許使用一般寄件者 ID，且必須識別品牌 • 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 公司名稱 • 公司地址 (包括公司城市、州/省、郵遞區號) • 公司國家 • 公司聯絡人電話號碼 • 公司網址 (可連接到您的應用程式或公司網站) • 預估每月數量 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 使用案例的說明、訊息的目的 • 公司的官方業務/貿易授權號碼或增值稅 # • 書面確認此寄件者 ID 發送的所有流量均由指定的使用案例進行說明 • 您計劃傳送的訊息範本

國名	訊息類型	格式限制和要求	註冊要求
越南 (VN)	僅傳送交易式訊息。不允許行銷和促銷訊息。禁止的內容包括： • 成人內容 • 慈善服務 • 加密貨幣 • 彩券 • 行動博弈/賭場 • 運動彩券 • 投票	• 最多 11 個字元。 • 沒有空格 • 無特殊字元	• 您要註冊的寄件者 ID • 使用者將呼叫 API/服務 的 AWS 區域 • 預估每月數量 • 如果不明顯，請提供公司名稱和寄件者 ID 之間的關係說明 • 使用案例的說明、訊息的目的 • 書面確認此寄件者 ID 發送的所有流量均由指定的使用案例進行說明

簽章格式限制和要求

若要順利傳送至中國，您必須在訊息內容範本中註冊訊息簽章。此訊息簽章必須放在每個要傳送的 SMS 的訊息內容前面。如未在 SMS 訊息本文前面加上註冊簽章，可能會導致 SMS 遭到封鎖或過濾。

- 簽章必須放在方括號內附加於 SMS 簡訊本文前面
- 標準文字必須包含在方括號內
- Unicode 文字必須使用方頭括號以包含簽章 – U+3010 左方頭括號和 U+3011 右方頭括號 – 範例：【注意】
- 必須介於 3 – 11 個字元之間
- 支援中文/英文字元

法國的寄件者 ID 需求

本指南提供了必要的步驟和指導方針，以建立法國行動電信業者將簡訊傳送至法國所需的專用寄件者 ID。

主題

設定法國的專用寄件者 ID

您可以使用下列其中一種方法來設定專用的寄件者 ID。針對使用 Publish API 發佈的簡訊，Amazon SNS 會代表您使用寄件者 ID。

- 您可以使用 Amazon SNS 主控台，來設定要用於所有已發佈之簡訊的預設寄件者 ID。若要進一步了解，請造訪 設定 SMS 訊息偏好設定 AWS Management Console。
- 請求 Amazon SNS 發佈簡訊時，您可以使用 Publish API，以利用 AWS.SNS.SMS.SenderID 訊息屬性設定寄件者 ID。若要進一步了解，請造訪 傳送訊息 (主控台)。

寄件者 ID 命名指導方針

- 寄件者 ID 名稱必須為英數字元，最多 11 個字元。
- 寄件者 ID 名稱不得包含特殊字元或空格。
- 我們建議您對寄件者 ID 和傳送簡訊之公司的品牌名稱使用相同的名稱。

印度的寄件者 ID 註冊要求

根據預設，當您傳送訊息給印度的收件人時，Amazon SNS 會使用國際長途營運商 (ILDO) 連線來傳輸這些訊息。當收件人者看到透過 ILDO 連線傳送的訊息時，它會顯示為由隨機數字 ID 傳送 (除非您購買專用的短碼)。

> **Note**
>
> 使用當地路由傳送訊息的價格會顯示在 Amazon SNS 全球簡訊定價頁面上。使用 ILDO 連線傳送訊息的價格，高於透過當地路由傳送訊息的價格。

如果您偏好針對簡訊使用字母寄件者 ID，則必須透過當地路由傳送這些訊息，而不是透過 ILDO 路由傳送。若要使用當地路由傳送訊息，您必須先向印度電信管理局 (TRAI) 透過分散式帳本 (DLT) 連接埠註冊您的使用案例和訊息範本。這些註冊要求旨在減少印度消費者收到的來路不明訊息數目，並保護消費者免受潛在有害訊息的威脅。這個註冊過程由 Vodafone India 透過其 Vilpower 服務管理。

主題

步驟 1：向 TRAI 註冊

您必須先向印度電信管理局 (TRAI) 註冊您的組織，才能傳送簡訊給印度的收件人。在註冊過程中，您必須提供下列資訊：

- 貴組織的永久帳戶號碼 (PAN)。
- 貴組織的扣稅帳戶編號 (TAN)。
- 貴組織的商品與服務稅務識別號碼 (GSTIN)。
- 貴組織的公司識別號碼 (CIN)。
- 授權您為貴組織進行註冊的授權書。

以下列出幾個分散式分類帳技術 (DLT) 註冊網站，您可以在這些網站上向 TRAI 註冊貴組織 (可能須支付費用)。每個網站的註冊流程會有所不同，如需協助，請聯絡各網站的專屬支援團隊。

- BSNL DLT - 免費註冊。
- Jio Trueconnect - 收取完成註冊程序的費用。
- Smart Enterprise Solutions - 收取完成註冊程序的費用。
- Vilpower - 提供可下載的範本，您可以依自身需求修改使用。Vilpower 會收取完成註冊程序的費用。

向 TRAI 註冊貴組織

以下詳細說明如何使用 Vilpower 向 TRAI 註冊貴組織。

1. 使用 Web 瀏覽器前往 Vilpower 網站：https://www.vilpower.in。
2. 選擇 Signup (註冊) 建立另一個帳戶。在註冊過程中，執行下列動作：
    - 對於要註冊為的實體類型，請選擇 As Enterprise (企業身分)。

- 對於電話行銷人員名稱，請使用 Infobip Private Limited - ALL (資訊私人有限公司 - 全部)。出現提示時，開始輸入 **`Infobip`**，然後在下拉式列表中選擇 Infobip Private Limited – ALL (資訊私人有限公司 - 全部)。
- 對於 Enter Telemarketer ID (輸入電話行銷人員識別碼)，輸入 **`110200001152`**。
- 當系統提示您提供標頭 ID 時，請輸入您要註冊的寄件者 ID。

> **Note**
>
> 印度要求寄件者 ID 的長度只有六個字元。

- 當系統提示您提供內容範本時，請輸入您打算傳送給收件人的訊息內容。包含您計劃傳送的每封訊息範本。

> **Note**
>
> DLT 註冊供應商網站並非由 Amazon Web Services 負責維護，網站上的註冊步驟可能會有所變更。

步驟 2：要求寄件者 ID

若要在印度請求寄件者 ID，您需要提交 AWS Support 請求。完成 請求寄件者 ID 中的步驟。請在您的請求中提供下列資訊：

- 此寄件者計劃傳送簡訊的來源 AWS 區域。
- 在 DLT 註冊過程中使用的公司名稱。
- 您在 DLT 實體註冊成功後收到的主要實體 ID (PEID)。
- 估計每月數量。
- 您的使用案例的解釋。
- 使用者輸入的流程說明。
- 確認已收集最終使用者的加入並完成註冊。

步驟 3：傳送簡訊

在向 TRAI 註冊貴組織後，您可以傳送簡訊給印度收件人。

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，選擇 Publish text message (發佈簡訊)。所以此 Publish SMS message (發佈簡訊) 視窗會開啟。
5. 對於 Message type (訊息類型)，選擇以下其中一項：

   - 促銷 - 非重要訊息，例如行銷訊息。

     使用數字寄件者 ID 時，請選擇此選項。
   - 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。

     使用字母或英數字元的寄件者 ID 時，請選擇此選項。

   此訊息級設定會覆寫您在 Text messaging preferences (簡訊喜好設定) 頁面設定的預設訊息類型。

   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。
6. 對於 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。

   將內容新增至簡訊時，請確定內容完全符合 DLT 註冊範本中的內容。如果訊息內容包含額外的字元換行、空格、標點符號或句子大小寫不符，電信業者就會封鎖 SMS 訊息。範本中的變數可以有 30 個或更少的字元。
8. 在來源身分區段中，針對寄件者 ID，輸入包含 3-11 個字元的自訂 ID。

   寄件者 ID 可以是促銷郵件的數字，或是交易郵件的字母或英數字元。寄件者 ID 會在接收的裝置上顯示為訊息寄件者。

   對於在印度註冊數字促銷寄件者 ID，請在 SMS 傳送要求中，指定寄件者ID 作為發起號碼參數。
9. 展開特定國家屬性區段，並指定傳送簡訊給印度收件人的下列必要屬性：

   - 實體 ID - 您從監管機構收到的實體 ID 或主體實體 (PE) ID，用於傳送簡訊給印度收件人。

     這是由 TRAI 提供的自訂字串，其中包含 1 至 50 個字元，可唯一識別您向 TRAI 註冊的實體。
   - 範本 ID - 您從監管機構收到的範本識別碼，用於傳送簡訊給印度收件人。

這是一個由 TRAI 提供的自訂字串，由 1 至 50 個字元組成，可唯一識別您向 TRAI 註冊的範本。範本 ID 必須與您在上一個步驟中指定的寄件者 ID 以及郵件內容相關聯。

10. 選擇 Publish message (發佈訊息)。

如需傳送簡訊訊給其他國家/地區收件人的詳細資訊，請參閱 發佈至行動電話。

傳送給印度收件人的簡訊疑難排解

電信業者可能會封鎖簡訊的部分原因如下：

- No template was found that matched the content sent. (找不到符合傳送內容的範本。)

  傳送的內容：**<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  相符的範本：無

  問題：沒有 DLT 範本在 DLT 註冊範本的開頭包含 <#> 或 {#var#}。

- The value of a variable exceeds 30 characters. (變數的值超過 30 個字元。)

  傳送的內容：**12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  符合的範本：**{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

  問題：傳送內容中「ABC 公司 - 印度私人有限公司」的價值超過單一 {#var#} 字元限制為 30。

- The message sentence case does not match the sentence case in the template. (訊息句子大小寫與範本中的句子大小寫不相符。)

  傳送的內容：**12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  符合的範本：**{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.**

  問題：附加至 DLT 相符範本的公司名稱會大寫，而傳送的內容已將名稱的部分變更為小寫 -「ABC Pvt. Ltd.」 與「ABC PVT. LTD.」

新加坡的寄件者 ID 註冊要求

Amazon SNS 客戶在新加坡可以使用已透過 Singapore SMS Sender ID Registry (SSIR) 註冊的寄件者 ID 傳送簡訊流量。SSIR 於 2022 年 3 月透過新加坡資訊通信媒體發展局 (IMDA) 擁有的新加坡網路資訊中心 (SGNIC) 啟動，讓組織能夠註冊其寄件者 ID，以便向新加坡行動電話傳送簡訊。

若要使用已註冊的新加坡寄件者 ID，您必須先取得唯一實體號碼 (UEN)，向 Amazon 提出請求，將您的帳戶加入寄件者 ID 的允許清單，最後透過 SSIR 完成註冊程序。

若您未在 2023 年 01 月 30 日 之前註冊 ID，任何使用寄件者 ID 傳送的訊息都會根據監管機構法規，將其 ID 變更為 LIKELY-SCAM。在此日期之後，監管機構將繼續自行決定篩選或封鎖未註冊的流量。

Important

若您要在 Amazon Pinpoint 區域 請求寄件者 ID，請使用 Amazon Pinpoint 主控台 註冊寄件者 ID。若要手動完成 Amazon Pinpoint 區域以外的註冊程序，請使用 新加坡寄件者 ID 註冊。
為了確保您能夠在新加坡傳送訊息，您必須在 2023 年 01 月 30 日之前完成註冊。
請務必按照以下順序完成註冊步驟。未按照順序執行這些步驟可能會導致您的寄件者 ID 被服務封鎖，或阻止您的寄件者 ID 保留在行動裝置上。
步驟 1。 註冊新加坡唯一實體編號 (UEN)
步驟 2. 若您要在 Amazon Pinpoint 區域 請求寄件者 ID，請遵照 Amazon Pinpoint 寄件者 ID 註冊 說明，來註冊寄件者 ID。

- 如要在帳戶 不 在 Amazon Pinpoint 區域 的情況下註冊寄件者 ID，請使用 新加坡寄件者 ID 註冊 說明手動註冊寄件者 ID。
- 代表其他公司傳送 SMS 文字時，需要公司發送授權書 (LOA)。
- 在提交您的 AWS 寄件者 ID 註冊後，請勿等待核准或變更狀態。立即前往步驟 3。

步驟 3。 向新加坡網路資訊中心 (SGNIC) 註冊寄件者 ID

主題

- 新加坡寄件者 ID 註冊狀態
- 編輯新加坡寄件者 ID 註冊
- 刪除新加坡寄件者 ID 註冊
- 新加坡註冊問題
- 新加坡寄件者 ID 註冊的常見問答集

註冊新加坡唯一實體編號 (UEN)

若要在 SSIR 開始註冊，您必須先取得新加坡唯一實體編號 (UEN)。UEN 是您向會計與企業管制局 (ACRA) 註冊公司時收到的唯一實體編號，如需更多詳細資訊，請參閱 Who Must Register with ACRA? (誰必須向 ACRA 申請註冊？)。程序時間可能因 ACRA 是否順利驗證您的請求而有所不同。

向 Amazon Pinpoint 註冊您的新加坡寄件者 ID

註冊新加坡唯一實體編號 (UEN) 後，您就可以在 Amazon Pinpoint 主控台完成寄件者 ID 註冊程序 (僅適用於 Amazon Pinpoint 區域)。註冊寄件者 ID 時，請確保訊息完整準確，否則您的註冊可能會遭到拒絕。

> **Important**
>
> 您透過 Amazon Pinpoint 主控台提交的資訊，將會傳送給我們的電信業者合作夥伴，以完成註冊程序。

註冊新加坡寄件者 ID

當帳戶位於 Amazon Pinpoint 區域 時，請使用這些步驟註冊寄件者 ID。如果您的帳戶不在 Amazon Pinpoint 區域，請參閱 手動註冊程序以完成新加坡寄件者 ID 註冊。

1. 登入 AWS 管理主控台，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 Sender ID registrations (寄件者 ID 註冊) 索引標籤上，選擇 Create registration (建立註冊)。
4. 選擇 Singapore (新加坡) 做為您的目的地國家。
5. 在 Company Information (公司資訊) 區段中，請輸入下列內容：
   - 在 Company Name (公司名稱) 中，輸入您公司在 UEN 註冊上的確切顯示名稱。

- 針對 Tax ID (稅務 ID)，輸入您從 ACRA 收到的 UEN 編號。
- 對於 Company Website (公司網站)，請輸入貴公司的網站 URL。
- 對於 Address 1 (地址 1)，請輸入您的公司總部街道地址。
- 針對 Address 2 (地址 2) - 選用，如果需要，請輸入公司總部的辦公室號碼。
- 針對 City (城市)，請輸入您的公司總部所在城市。
- 針對 State (州)，請輸入您的公司總部所在州別。
- 針對 Zip Code (郵遞區號)，請輸入您的公司總部的郵遞區號。
- 針對 Country (國家)，請輸入兩位數的 ISO 國家代碼。

6. 在 Contact Information (聯絡資訊) 區段中，請輸入下列內容：
   - 對於 First Name (名字)，請輸入您公司聯絡人員的名字。
   - 對於 Last Name (姓氏)，請輸入您公司聯絡人員的姓氏。
   - 對於 Support Email (支援電子郵件)，請輸入您公司聯絡人員的電子郵件地址。
   - 對於 Support Phone Number (支援電話號碼)，請輸入您公司聯絡人員的電話號碼。
7. 在 Sender ID Information (寄件者 ID 資訊) 中，請輸入下列內容：
   - 針對 Sender ID (寄件者 ID)，請輸入您要顯示於訊息的寄件者 ID。
   - 針對 Registering on behalf of another brand/entity? (代表其他品牌/實體註冊？)，如果是，則選擇「是」。如果您不是傳送訊息的最終使用者，您將被視為其他品牌/實體的「代表」。
   - 針對 Letter of authorization image - optional (授權書影像 - 選用)，您是否選取此選項，表示代表其他品牌/實體註冊？請上傳完整授權書 (LOA) 的影像。支援的檔案類型為 PNG，檔案大小上限為 400KB。為了方便起見，您可以下載 LOA 範本。
   - 針對 Sender ID connection - optional (寄件者 ID 連線 - 選用)，您可以新增更多詳細資訊，說明請求的 SenderID 和公司名稱之間的連線。
8. 在 Messaging Use Case (訊息使用案例) 區段，請執行下列動作：
   - 對於 Monthly SMS Volume (每月簡訊資料量)，請選擇每個月的簡訊數量。
   - 針對 Use Case Category (使用案例類別)，請從下列選取該號碼的使用案例類型：
     - Two-factor authentication (雙重身分驗證) – 用於傳送雙重身分驗證代碼。
     - One-time passwords (一次性密碼) – 用於向使用者傳送一次性密碼。
     - Notifications (通知) – 若您只打算向使用者傳送重要通知，請使用此選項。
     - Polling and surveys (輪詢和問卷) – 使用此選項可根據使用者偏好輪詢使用者。
     - Info on demand (隨需資訊) – 用於使用者傳送請求後傳送訊息。
     - Promotions and Marketing (促銷與行銷) – 若您只打算向使用者傳送行銷訊息，請使用此選項。

- Other (其他) – 其他如果您的使用案例不屬於任何其他類別，請使用此選項。請確保您已針對此選項填寫 Use Case Details (使用案例詳細資訊)。
- 填寫Use Case Details - optional (使用案例詳細資訊 - 選用)，以便為選取的 Use Case Category (使用案例類別) 提供其他內容。

9. 在 Message details(訊息詳細資訊) 區段中，請執行下列動作：
   - 對於 Message Sample 1 (訊息範例 1)，請輸入要傳送給終端使用者的簡訊內文之範例訊息。
   - 針對 Message Sample 2 (訊息範例 2) optional (選用) 和 Message Sample 3 (訊息範例 3) optional (選用)，如有需要，您可以輸入要傳送的 SMS 訊息內文範例。
   - 每個 Message Sample (訊息範例) 文字方塊的字元上限為 306 個字元。
10. 完成時請選擇 Submit registration (提交註冊)。

> **Important**
>
> 您可以按照 新加坡寄件者 ID 註冊狀態 中的說明檢查您的註冊狀態。
> 在提交您的寄件者 ID 註冊後，請勿等待核准或變更狀態。立即前往 向新加坡網路資訊中心 (SGNIC) 註冊寄件者 ID。

手動註冊程序以完成新加坡寄件者 ID 註冊

當帳戶 不在 Amazon Pinpoint 區域 時，請使用這些步驟註冊寄件者 ID。如果您的帳戶位於 Amazon Pinpoint 區域，請參閱 向 Amazon Pinpoint 註冊您的新加坡寄件者 ID。

1. 下載 Singapore_Sender_ID_Registration_LOA_Template.zip 並完成所需資訊。
2. 建立具有 AWS 支援的案例。
3. 在 Open support cases (開啟支援案例) 索引標籤中，選擇 Create case (建立案例)。
4. 選擇 尋找增加服務限制額度，對於限制類型，選擇 SNS 簡訊案例。
5. 針對 Resource Type (資源類型)，選擇 Sender ID Registration (寄件者 ID 註冊)。
6. 附上 LOA 文件並 submit (提交) 請求。

向新加坡網路資訊中心 (SGNIC) 註冊寄件者 ID

> **Warning**
>
> 未按照順序執行這些步驟可能會導致您的寄件者 ID 被服務封鎖，或阻止您的寄件者 ID 保留在行動裝置上。

1. 您必須先使用 AWS (Amazon Pinpoint 主控台，或在非 Amazon Pinpoint 區域手動註冊)，為帳戶註冊新加坡 (SG) 寄件者 ID。完成此步驟後，您可以繼續進行下一個步驟。
2. 與 SGNIC 合作，透過 SGNIC 簡訊寄件者 ID 註冊的程序，註冊寄件者 ID。
   - 完成後，請務必將下列所有項目列為您的「參與彙總工具」：
     - AMCS SG Private Limited (Amazon Media Communications Services)
     - Nexmo PTE LTD
     - Sinch Singapore PTE LTD
     - Telesign Singapore PTE LTD
     - Twilio Singapore PTD LTD

> **Note**
>
> 您必須從需使用寄件者 ID 的每個個別 AWS 帳戶提交寄件者 ID 註冊。

新加坡寄件者 ID 註冊狀態

在 Amazon SNS 註冊新加坡寄件者 ID 時，您的註冊可能為下列五種狀態：

- 已建立 - 已建立您的註冊，但尚未提交。
- 已提交 - 已提交您的註冊並進行驗證中。
- Reviewing (檢閱中) – 已接受您的註冊並進行檢閱中。可能需要 1 至 3 週的時間，在部份情況下，可能需要更長的時間才能完成檢閱。
- 完成 – 已核准您的註冊，您可以開始使用該寄件者 ID。
- 需要更新 - 您必須修改註冊並重新提交。如需詳細資訊，請參閱 編輯新加坡寄件者 ID 註冊。需要更新的欄位會顯示警告圖示和問題的簡短說明。

針對 Amazon Pinpoint 區域以外的所有區域，AWS 支援會在註冊時傳送電子郵件確認，或透過 AWS 支援建立案例。

- 在 Open support cases (開啟支援案例) 索引標籤中，選擇 Create case (建立案例)。
- 選擇 Service limit increase (提高服務限制)。
- 針對資源類型，選擇 Sender ID Registration (寄件者 ID 註冊)，並針對限制選擇 General Inquiry (一般查詢)。

Check your registration status (檢查您的註冊狀態)

1. 登入 AWS 管理主控台，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 Sender ID registrations (寄件者 ID 註冊) 索引標籤上，選擇 SenderID。
4. 您可以檢視每個 SenderID 的註冊狀態。

編輯新加坡寄件者 ID 註冊

以 Amazon Pinpoint 提交註冊後，如果註冊有問題，registration status (註冊狀態) 將為 Requires Updates (需要更新)。在此狀態下，註冊表單是可編輯的。需要更新的欄位會顯示警告圖示和問題的簡短說明。

編輯寄件者 ID

1. 開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 SenderID Registration (寄件者 ID 註冊) 索引標籤上，選擇您要編輯的號碼，然後選取 Registration ID (註冊 ID)。
4. 選擇 Update registration (更新註冊) 以編輯表單並更正具有警告圖示的欄位。
5. 如果您代表其他品牌/實體註冊，則需重新上傳針對 Letter of authorization image - optional (授權書影像 - 選用) 提供的檔案。
6.

> **Important**
>
> 請重新檢查所有欄位，以確保欄位內容正確無誤。

7. 完成時，請選擇 提交註冊 以重新提交。

刪除新加坡寄件者 ID 註冊

如果您不想繼續註冊新加坡寄件者 ID，您可以刪除註冊。只有在狀態為 已建立 或 需要更新 時才能刪除註冊。

To delete a registration (若要刪除註冊)

1. 開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 Sender ID (寄件者 ID) 索引標籤上，選擇您要刪除的註冊 ID，然後選擇 Delete Registration (刪除註冊)。

新加坡註冊問題

若 Amazon Pinpoint 不接受您的新加坡寄件者 ID，您會收到一則訊息，解釋遭拒原因。若您對此拒絕有問題，且未在最佳實踐得到解答，您可以向我們的支援團隊提交請求。

針對遭拒的新加坡寄件者 ID 提出請求

1. 開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 選擇 Support (支援)，接著選擇 Support Center (支援中心)。
3. 在支援頁面上，選擇 Create Case (建立案例)。
4. 針對 Case type (案例類型) 選擇 Service limit increase (提升服務配額)。
5. 針對 Limit type (限制類型)，選擇 Pinpoint SMS。
6. 在 Requests (請求) 區段，執行下列動作：
   - 針對 Resource Type (資源類型)，請選擇 Sender ID Registation (寄件者 ID 註冊)。
   - 針對 Limit (限制)，請選擇 Registration Rejection Query (註冊拒絕查詢)。
7. 在 Use case description (使用案例說明) 中，輸入遭拒的新加坡寄件者 ID，並提供拒絕原因。
8. 在 Contact options (聯絡人選項) 下，針對 Preferred contact language (偏好的聯絡語言)，選擇您與 AWS 支援團隊通訊時偏好使用的語言。
9. 針對 Contact method (聯絡方法)，選擇您與 AWS 支援團隊聯絡的慣用方法。
10. 選擇 Submit (提交)。

AWS Support 團隊將在 AWS Support 案例中提供相關資訊，說明您的寄件者 ID 註冊遭到拒絕的原因。

新加坡寄件者 ID 註冊的常見問答集

Amazon Pinpoint 的新加坡寄件者 ID 號碼註冊程序的常見問答集。

我目前有新加坡寄件者 ID 嗎？

檢查您是否擁有新加坡寄件者 ID

1. 開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 SenderID Registration (寄件者 ID 註冊) 索引標籤上，選擇您要檢視的寄件者 ID，然後選取 Registration ID (註冊 ID)。

完成註冊需要多久時間？

一般審查需要 1 至 3 週，但在部份情況下，可能需 5 週或更長時間才能向政府機構驗證您的資訊。

什麼是唯一實體編號 (UEN)？我該如何取得？

UEN 是由會計和公司監管局 (ACRA) 簽發的新加坡企業 ID。新加坡的本地企業可以向 ACRA 申請獲得 UEN。一旦您通過註冊和標準註冊程序，即可獲得 UEN。您可以透過 Bizfile 向 ACRA 申請 UEN。

我必須註冊新加坡寄件者 ID 嗎？

是。若您未在 2023 年 01 月 30 日之前註冊 ID，任何使用寄件者 ID 傳送的訊息 ID 都會變更為 LIKELY-SCAM。

如何向 Amazon Pinpoint 註冊我的新加坡寄件者 ID？

請依照以 Amazon Pinpoint 註冊新加坡寄件者 ID 的說明，來註冊寄件者 ID。

我的寄件者 ID 註冊狀態為何？該狀態代表什麼？

按照新加坡寄件者 ID 註冊狀態的指示，檢查您的註冊和狀態。

我需要提供哪些資訊？

您需要提供貴公司的地址、公司聯絡人和使用案例。您可以在以 Amazon Pinpoint 註冊新加坡寄件者 ID 中找到所需資訊。

如果我的新加坡寄件者 ID 註冊遭到拒絕怎麼辦？

如果您的註冊遭到拒絕，註冊狀態將變更為「需要更新」，您可以按照編輯新加坡寄件者 ID 註冊的指示，以進行更新。

我需要哪些許可？

您必須以 *"sms-voice:*"* 許可，啟用造訪 Amazon Pinpoint 主控台的 IAM 使用者/角色。

來源號碼

同時來源號碼是識別簡訊寄件者電話號碼的數字字串。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。

> **Tip**
>
> 若要檢視 AWS 帳戶中所有現有來源號碼的清單，請在 Amazon SNS 主控台的導覽窗格中，選擇 Origination numbers (來源號碼)。

當地法律要求使用 寄件者 ID 而不是來源號碼。

主題

- 10DLC
- 免付費電話號碼
- 短代碼
- 人對人 (P2P) 長代碼
- 美國產品編號比較

10DLC

美國電信業者不再支援透過本地、未註冊的長碼使用應用程式至人員 (A2P) 簡訊。對於高容量 A2P 簡訊，美國電信業者提供一種新型的長碼，稱為 10 位數長碼 (10DLC)。

> **Important**
>
> 自 2023 年 1 月 26 日起，Amazon SNS 的簡訊供應商在 10DLC 行銷活動上推出了新的手動審查程序，以解決美國電信業者提出的垃圾郵件問題。您可以使用短碼和免費電話號碼作為 10DLC 的替代方案，在美國傳送簡訊。
>
> 目前，我們的 SMS 廠商尚未針對 10DLC 促銷活動審查需要多長時間提供服務等級目標。一旦數字與 10DLC 促銷活動相關聯，就會觸發評論。評論所花費的時間超過 Amazon SNS 先前傳達的 14 天估計時間。

Amazon SNS 每天都與短信供應商合作，以確保：

- 供應商盡快完成任何待處理的 10DLC 促銷活動評論
- 供應商在待辦項目中優先處理 AWS 請求

您可以按照 10DLC 促銷活動中的指示，查看 10DLC 促銷活動的狀態。如果需要額外資訊才能核准 10DLC 促銷活動，AWS 支援團隊會通知您。
您可以註冊美國免費電話號碼，這比獲得 10DLC 號碼更快。如需美國免付費電話號碼和註冊程序的詳細資訊，請參閱 免付費電話號碼註冊要求和程序。

什麼是 10DLC？

10DLC 是一種在電信業者註冊的長碼，以支援使用 10 位數的電話號碼格式的高容量 A2P 短訊訊息。Amazon SNS 不再提供本地長碼作為簡訊產品，而是提供 10DLC。如果您只使用短碼和免費電話號碼，10DLC 不會影響您。

10DLC 是僅在美國使用的 10 位數電話號碼。從 10DLC 傳送給收件者的郵件會顯示 10 位數字作為寄件者。與免費電話號碼不同，10DLC 同時支援交易和行銷活動訊息，並可包含任何美國區碼。

如果您有現有的本地長碼，您可以要求為 10DLC 啟用其本地長碼。若要這麼做，請完成 10DLC 註冊程序，然後提交支援票證。如果啟用 10DLC 的長程式碼發生問題，系統會通知您並指示您透過 Amazon Pinpoint (非 Amazon SNS) 主控台要求新的 10DLC。如需如何提交支援票證以轉換長程式碼的相關資訊，請參閱 將長代碼與 10DLC 廣告活動建立關聯。

若要使用 10DLC 號碼，請先註冊您的公司，並使用 Amazon Pinpoint (而非 Amazon SNS) 主控台建立 10DLC 廣告活動。AWS 與活動註冊處分享此資訊，該第三方根據資訊核准或拒絕您的註冊。在某些情況下，會立即登錄。例如，如果您先前曾向「活動登錄」註冊，他們可能已經有您的資訊。不過，某些廣告活動可能需要一週或更長的時間才能核准。在您的公司和 10DLC 活動獲得核准後，您可以購買 10DLC 號碼，並將其與您的活動建立關聯。申請 10DLC 最多可能需要一週的時間才能核准。雖然您可以將多個 10DLC 與單一廣告活動建立關聯，但您無法在多個廣告活動中使用相同的 10DLC。對於您建立的每個廣告活動，您都需要有一個獨特的 10DLC。

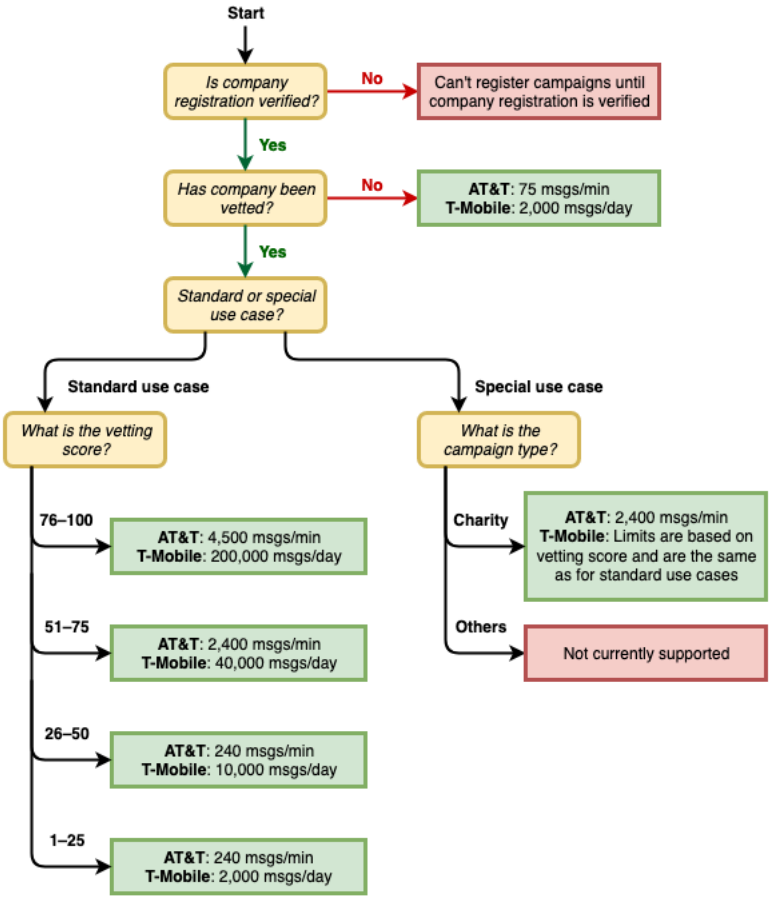
10DLC 功能

10DLC 電話號碼的輸送量取決於收件人使用的行動電信業者。AT&T 會根據每場行銷活動每分鐘可傳送的訊息部分數目，設定輸送量上限。T-Mobile 對可傳送給每間公司的訊息數量設定了每日上限，但並未限制每分鐘可傳送的訊息部分數量。Verizon 尚未公佈輸送量上限，但會使用 10DLC 篩選系統移除垃圾郵件、來路不明的訊息和濫用內容，較少關注實際訊息輸送量。

與未經審核的公司相關聯的新 10DLC 行銷活動，每分鐘可以向使用 AT&T 的收件人發送 75 個訊息部分，每天可以向使用 T-Mobile 的收件人發送 2,000 則訊息。公司上限由所有 10DLC 行銷活動共用。舉例來說，如果您註冊了一間公司和兩場行銷活動，那麼這兩場活動每天總共可以向 T-Mobile 客戶發送 2,000 則訊息；同樣地，如果您在多個 AWS 帳戶中註冊了同一間公司，每日分配量也將由這些帳戶共用。

如果您的輸送量需求超過這些上限，您可以申請公司註冊資訊審核。您將公司註冊資訊送交審核後，會由第三方驗證供應商分析貴公司的詳細資訊。審核結束後，驗證供應商會提供審核分數，用於決定您的 10DLC 行銷活動輸送量。每次申請審核服務，皆須支付一筆一次性費用。如需更多詳細資訊，請參閱審核您的 Amazon SNS 10DLC 註冊資訊。

您的實際輸送率會因各種因素而有所不同，例如貴公司是否經過審核、您的行銷活動類型，以及您的審核分數。下方的流程圖顯示了各種情況下的輸送率。



10DLC 的輸送率由美國行動電信業者與活動登記處共同決定，Amazon SNS 或任何其他簡訊傳送服務都不能將 10DLC 輸送量提高到規定標準以上。如果您需要所有美國電信業者的高輸送率和高交付率，建議您使用簡短代碼。如需取得簡短代碼的詳細資訊，請參閱針對使用 Amazon SNS 的簡訊請求專用的短碼。

10DLC 入門

請使用 Amazon Pinpoint 主控台 (而非 Amazon SNS) 申請 10DLC。請依照這些步驟設定 10DLC，以搭配您的 10DLC 廣告活動使用。

1. 登錄您的公司。

   您的公司必須先向活動登記處登錄，才能申請 10DLC；詳情請參閱 登錄公司。除非活動登記處需要更多資訊，否則登錄通常會立即完成。登錄頁面會顯示註冊貴公司的一次性登錄費用。此一次性費用將與您每月的活動費用和 10DLC 分開支付。

   > **Note**
   >
   > 在目前不支援 Amazon Pinpoint 的區域中，可使用 Amazon SNS 簡訊。有兩種不同的情況：
   >
   > a. 如果您使用的是商業雲端帳戶，則必須開啟美國東部 (維吉尼亞北部) 區域的 Amazon Pinpoint 主控台，以註冊您的 10DLC 公司和廣告活動。請勿請求 10DLC 號碼。
   >
   > b. 請使用 AWS Service Quotas 主控台來建立服務限制提升案例，同時請求該區域的 10DLC 號碼。如需 Amazon Pinpoint 可在其中使用之區域的詳細資訊，請參閱 AWS 一般參考 中的 Amazon Pinpoint 端點和配額。
   >
   > c. 如果您使用的是 AWS GovCloud (US) 帳戶，請在美國西部區域開啟 Amazon Pinpoint 主控台，以註冊您的 10DLC 公司和行銷活動。請勿請求 10DLC 號碼。請改用 AWSService Quotas 主控台來建立提高服務配額案例，同時要求該區域的 10DLC 號碼。如需 Amazon Pinpoint 可在其中使用之區域的詳細資訊，請參閱 AWS 一般參考 中的 Amazon Pinpoint 端點和配額。

2. (非必要但建議採取的步驟) 申請審核

   如果貴公司註冊成功，您就可以開始建立少量混合使用的 10DLC 行銷活動，這類行銷活動每分鐘可以向使用 AT&T 的收件人發送 75 則訊息，您的註冊公司則可每天向使用 T-Mobile 的收件人發送 2,000 則訊息。如果您的使用案例需要的輸送率超過這些值，您可以申請公司註冊資訊審核。將公司註冊資訊送交審核或許能提高公司和行銷活動的輸送率，但不保證一定可行。如需進一步瞭解如何審核，請參閱審核您的 Amazon SNS 10DLC 註冊資訊。

3. 註冊您的廣告活動。

   您的公司登錄後，建立 10DLC 廣告活動，並將其與您登錄的公司之一建立關聯。此活動已提交至活動登記處核准。在大多數情況下，除非活動登記處需要更多資訊，否則 10DLC 活動會即時核准。如需更多詳細資訊，請參閱 登錄 10DLC 廣告活動。

4. 索取您的 10DLC 號碼。

   在您的 10DLC 廣告活動獲得核准後，您可以申請 10DLC，並將該號碼與核准的廣告活動建立關聯。您的 10DLC 廣告活動只能使用核准的號碼。請參閱 透過 Amazon SNS 要求 10DLC 號碼、免付費電話號碼和 P2P 長代碼以進行簡訊傳送。

10DLC 登錄及月費

有使用 10DLC 相關的註冊費和月費，例如登錄您的公司和 10DLC 廣告活動。這些有別於 AWS 收取的任何其他月費。如需詳細資訊，請參閱 Amazon SNS Worldwide SMS Pricing 頁面。

登錄公司

在您申請 10DLC 之前，您必須先向活動登記處登錄您的公司。

> **Note**
>
> 在目前不支援 Amazon Pinpoint 的區域中，可使用 Amazon SNS 簡訊。在這些情況下，請開啟美國東部 (維吉尼亞北部) 區域的 Amazon Pinpoint 主控台，登錄您的 10DLC 公司和廣告活動，但是請勿要求一個 10DLC 號碼。請改用 AWSService Quotas 主控台來建立提高服務配額案例，同時要求該區域的 10DLC 號碼。如需 Amazon Pinpoint 可在其中使用之區域的詳細資訊，請參閱 AWS 一般參考 中的 Amazon Pinpoint 端點和配額。

10DLC 公司註冊狀態

您註冊公司或品牌後，系統會傳回下列其中一種狀態：Unverified (未驗證) 或 Verified (已驗證)。如果您的公司註冊狀態為 Unverified (未驗證)，代表註冊過程出現問題，例如您提供的註冊公司名稱可能與您提供的公司註冊名稱相關稅務 ID 不完全符合。如果您發現公司註冊詳細資訊有問題，可加以更正。如需進一步瞭解如何修改公司註冊詳細資訊，請參閱編輯或刪除註冊公司。

如果公司註冊的狀態顯示為「Verified」(已驗證)，則表示您提供的註冊詳細資訊準確無誤，可以開始建立 10DLC 行銷活動。

註冊您的公司或品牌

您只需登錄您的公司一次。註冊完畢後，您可以編輯您的公司和聯絡資訊。如要刪除已註冊的公司，請使用 AWS Support 建立案例。如需進一步瞭解如何編輯或刪除公司詳細資訊，請參閱編輯或刪除註冊公司。

登錄公司

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 10DLC campaigns (10DLC 行銷活動) 索引標籤上，選擇 Register company (註冊公司)。

> **Note**
>
> Register your company (註冊您的公司) 頁面上會顯示 Registration fee (註冊費)，這是與登錄您的公司相關的一次性費用。這筆費用獨立於任何其他每月費用或支出，當您註冊公司或修改現有公司的註冊詳細資訊時，便須支付這筆費用。

4. 在 Company info (公司資訊) 區段中，執行下列操作：

   - 在 Legal company name (法定公司名稱) 一欄輸入公司註冊的名稱。您輸入的名稱必須與您提供的稅務 ID 關聯公司名稱完全相同。

> **Important**
>
> 請務必使用貴公司的確切法定名稱，這項資訊一經提交便無法再行變更。如您提交的資訊不正確或不完整，可能導致註冊程序延遲或遭到拒絕。

   - 在 What type of legal form is this organization (這個組織的法律形式類型) 一欄，選擇對貴公司而言最適合的選項。

> **Note**
>
> 唯有要為設於美國的組織進行註冊時，才能選擇 US government (美國政府) 與 Not-for-profit (非營利) 選項。如果您的組織位於美國以外的國家/地區，那麼無論貴組織實際的法律形式為何，都必須註冊為 Private for-profit (私人營利) 組織。

   - 如果您在上一步選擇了 Public for profit (公共營利)，請輸入公司的股票代號和其上市的證券交易所。

- 在 Country of registration (註冊國家/地區) 列表中選擇公司進行註冊的國家/地區。
- 在 Doing Business As (DBA) or brand name (經營別稱 (DBA) 或品牌名稱)一欄，輸入貴公司用於執行業務的任何其他名稱。
- 在 Tax ID (稅務 ID) 一欄輸入貴公司的稅務 ID。您輸入的 ID 取決於貴公司進行註冊的國家/地區。
  - 如果您想註冊的是具有 IRS 雇主身分識別碼 (EIN) 的美國或非美國企業，請輸入您的 9 位數 EIN。您輸入的法定公司名稱、EIN 和實體地址必須與您在 IRS 註冊的公司資訊完全相同。
  - 如果您要註冊的是加拿大企業，請輸入您的聯邦或省級公司編號，而非 CRA 提供的企業號碼 (BN)。您輸入的法定公司名稱、公司編號和實體地址必須與在您加拿大公司局 (Corporations Canada) 註冊的公司資訊完全相同。
  - 如果您要註冊的企業位於其他國家/地區，請輸入您所在國家/地區的主要稅務 ID；在許多國家/地區，這都是指您 VAT ID 的數字部分。
- 在 Vertical (產業) 一欄選擇對您要註冊的公司而言最合適的類別。

5. 在 Contact info (聯絡資訊) 區段中，執行下列操作：

- 在 Address/Street (地址/街道) 一欄，輸入與貴公司關聯的實體街道地址。
- 在 City (城市) 一欄，輸入實體地址所在的城市。
- 在 State or region (州或區域) 一欄，輸入實體地址所在的州或區域。
- 在 Zip Code/Postal Code (郵遞區號) 一欄,輸入實體地址的郵遞區號。
- 在 Company website (公司網站) 一欄，輸入貴公司的完整網站 URL，地址開頭須包含「http://」或「https://」。
- 在 Support email (支援電子郵件) 一欄，輸入電子郵件地址。
- 在 Support Phone number (支援電話號碼) 一欄，輸入附帶國家/地區代碼的電話號碼。

> **Note**
>
> 為了與貴公司代表確認註冊資訊，活動登記處需要公司的聯絡電子郵件地址和電話號碼。

6. 當您完成時，請選擇 Create (建立)。您的公司註冊資料會提交至活動登記處。在大多數情況下，系統會立即接受您的註冊並提供註冊狀態。

如果貴公司的註冊狀態為 Verified (已驗證)，您就可以開始建立少量混合使用的 10DLC 行銷活動。透過這類行銷活動，您每分鐘最多可向使用 AT&T 的收件人發送 75 則訊息，您的註冊公司每天則可向使

用 T-Mobile 的收件人發送 2,000 則訊息。您也可以向使用 Verizon、US Cellular 等其他美國電信業者的收件人發送訊息，這些電信業者不會嚴格執行輸送量限制，但會嚴密監控 10DLC 訊息是否有發送垃圾郵件和濫用的跡象。

如果您的使用案例所需的輸送率超過這些值，您可以申請對公司註冊資訊進行額外審核。如需更多審核品牌註冊的詳細資訊，請參閱審核您的 Amazon SNS 10DLC 註冊資訊。

如果貴公司的註冊狀態為 Unverified (未驗證)，代表您提供的資訊有問題。請檢查您提供的資訊，確認所有欄位輸入的資訊都正確無誤。您可以在 Amazon Pinpoint 主控台中變更部分公司註冊資訊。如需進一步瞭解如何修改公司註冊詳細資訊，請參閱編輯 10DLC 公司註冊資訊。

審核您的 Amazon SNS 10DLC 註冊資訊

如果貴公司已註冊成功，但您想註冊需要更高輸送量的行銷活動，貴公司的註冊資訊便須接受審核。

審核註冊資訊時，會由第三方組織會分析您提供的公司詳細資訊，然後傳回審核分數；假如審核獲得高分，您的 10DLC 公司及其相關行銷活動便可使用較高的輸送率。不過，並非只要進行審核就一定能夠提高輸送量。

審核分數不會溯及以往；換句話說，如果您在建立 10DLC 行銷活動後才將貴公司的註冊資訊送交審核，您獲得的審核分數不會自動套用至現有的行銷活動。因此，建立 10DLC 行銷活動之前，請務必先將貴公司或品牌送交審核。

> **Note**
>
> 將公司或品牌送交審核需支付 40 USD，這筆費用不會退還。

如何將公司註冊資訊送交審核

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 10DLC 行銷活動索引標籤上，選擇想送交審核的 10DLC 公司。
4. 在公司詳細資訊頁面上，選擇底部的 Apply for vetting (申請審核)。
5. 在 Apply for additional vetting (申請額外審核) 視窗中選擇 Submit (提交)。

   對位於美國的公司來說，審核流程通常只需約一分鐘就能完成，但美國境外的公司所需的審核時間可能會多出許多，取決於該國家/地區立即可用的資料傳輸量。

提交審核申請後，您將返回公司詳細資訊頁面，Company vetting results (公司審核結果) 區段會顯示審核申請的狀態和結果。審核流程完成後，此資料表的 Score (分數) 欄位會顯示審核分數，您的 10DLC 輸送量就取決於這個審核分數。您的輸送量會因您建立的行銷活動類型而異；如果您建立的是混合使用或與行銷相關的 10DLC 行銷活動，那麼比起其他行銷活動類型，您必須獲得更高的審核分數才能達到高輸送率。如想進一步瞭解 10DLC 電話號碼的輸送量，請參閱10DLC 功能。

如果您在完成審核流程後變更了公司的註冊詳細資訊，可以申請重新審核。假如您只更改了公司註冊資訊的「產業」，審核分數不會改變，但如果您更改了「產業」以外的任何詳細資訊，則審核結果可能會有所變動。無論是哪一種情況，您都必須再支付一筆一次性審核費用。

編輯或刪除註冊公司

您可以直接在 Amazon Pinpoint 主控台中為貴公司編輯部分 10DLC 註冊資訊，也可以在 AWS Support Center 建立案例，藉此刪除 10DLC 公司註冊資訊。

編輯 10DLC 公司註冊資訊

完成公司的 10DLC 註冊流程後，您便可以編輯註冊詳細資訊。

如果您在編輯公司註冊詳細資訊後看到錯誤訊息，代表您的註冊資訊可能有其他問題，您可以向 Sup AWS port 人員開立票證以要求更多資訊。

若要編輯公司註冊

1. 請在以下位置開啟 AWS SMS 主控台。 https://console.aws.amazon.com/sms-voice/
2. 請依照 Amazon Pinpoint 位簡訊使用者指南中的編輯註冊說明進行操作。

刪除 10DLC 公司註冊資訊

刪除公司註冊

1. 請在以下位置開啟 AWS SMS 主控台。 https://console.aws.amazon.com/sms-voice/
2. 按照 Amazon Pinpoint 位簡訊使用者指南中刪除註冊的說明進行操作。

登錄 10DLC 廣告活動

註冊 10DLC 行銷活動時，您必須提供使用案例描述以及您計劃使用的訊息範本，且貴公司必須先註冊，才能建立和註冊 10DLC 行銷活動。如需進一步瞭解如何註冊公司，請參閱登錄公司。

> **Note**
>
> 註冊公司後，Amazon Pinpoint 會顯示下列其中一種註冊狀態：Verified (已驗證) 或 Unverified (未驗證)；貴公司的註冊狀態必須為 Verified (已驗證)，您才能完成 10DLC 行銷活動註冊流程，並建立少量混合使用行銷活動。
>
> 如果狀態為 Unverified (未驗證)，通常代表您在註冊公司時提供的某些資料不正確；當貴公司處於此狀態時，您無法建立任何 10DLC 行銷活動。您可以修改貴公司的註冊資訊，以嘗試修正公司註冊問題。如需進一步瞭解如何修改 10DLC 公司註冊資訊，請參閱編輯或刪除註冊公司。

在此頁面上，您會先提供您要建立 10DLC 廣告活動的公司詳細資料，然後提供廣告活動本身的使用案例詳細資料。此頁面上的資料將提供予活動登記處核准。

在本區段中，您將選擇您要建立 10DLC 廣告活動的公司，並提供其他詳細資料。

如何註冊 10DLC 行銷活動

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在 SMS and voice (簡訊和語音)下方選擇 Phone numbers (電話號碼)。
3. 在 10DLC campaigns (10DLC 行銷活動) 索引標籤上，選擇 Create a 10DLC campaign (建立 10DLC 行銷活動)。
4. 在 Create a 10DLC campaign (建立 10DLC 行銷活動) 頁面上的Campaign info (行銷活動資訊) 區段，執行下列動作：
    a. 在 Company name (公司名稱)，請選擇您要為其建立此廣告活動的公司。您必須先登錄該公司。如需進一步瞭解如何註冊公司，請參閱登錄公司。
    b. 在 10DLC Campaign name (10DLC 行銷活動名稱) 一欄，輸入行銷活動的名稱。
    c. 在 Vertical (產業) 一欄，選擇對貴公司而言最合適的選項。
    d. 在 Help message (協助訊息) 一欄，輸入客戶將關鍵字「HELP」發送至您的 10DLC 電話號碼後會收到的訊息。
    e. 在 Stop message (停止訊息) 一欄，輸入客戶將關鍵字「STOP」發送至您的 10DLC 電話號碼後會收到的訊息。

> **Tip**
>
> 您的客户可以用「HELP」一詞回覆您的訊息，藉此進一步瞭解您傳送給他們的訊息。他們也可以回覆「STOP」，選擇不接收您的訊息。美國行動電信業者要求您對這兩個關鍵字提供回應。
>
> 以下是符合美國行動電信業者要求的 HELP 回應示例：
>
> **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
>
> 以下是合規的 STOP 回應示例：
>
> **`You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.`**
>
> 您對這些關鍵字的回應最多不得超過 160 個字元。

5. 在 Campaign use case (行銷活動使用案例) 區段執行下列動作：

   a. 針對 Use case type (使用案例類型)，如果您有與慈善相關的使用案例，請選擇 Special (特殊)，否則請選擇 Standard (標準)。

   b. 在 Use case (使用案例)，請從預設使用案例清單中選擇最接近廣告活動的使用案例。每個使用案例的月費會顯示在使用案例名稱旁邊。

   > **Note**
   >
   > 註冊 10DLC 行銷活動的月費會顯示在每個使用案例類型旁邊，大多數 10DLC 行銷活動類型的月費都相同。少量混合使用案例的註冊費用低於其他使用案例類型，但是少量混合行銷活動所支援的輸送率也比其他行銷活動類型更低。

   c. 輸入至少一個簡訊範例。這是您計劃傳送給客戶的訊息範例。如果您計劃在這場 10DLC 行銷活動中使用多個訊息範本，請將所有範本都加入。

   > **Important**
   >
   > 請勿在訊息範例中使用預留位置文字，您提供的訊息範例應盡可能與您計劃發送的實際訊息內容相符。

6. Campaign and content attributes (行銷活動和內容屬性) 區段包含一系列與行銷活動特定功能相關的是非題。某些屬性是強制性的，因此您無法變更預設值。

請務必為您的行銷活動選擇的正確的屬性。

請指明下列各個項目是否適用於您正在註冊的行銷活動：

- 訂閱者加入 - 訂閱者可以選擇加入以接收有關此活動的訊息。
- 訂閱者退出 - 訂閱者可以選擇不要接收此行銷活動的訊息。
- 訂閱者說明 - 訂閱者可以在傳送 HELP 關鍵字之後連絡郵件寄件者。
- 數字集區 - 這個 10DLC 活動使用超過 50 個電話號碼。
- 直接貸款或貸款安排 - 活動包括有關直接貸款或其他貸款安排的資訊。
- 內嵌連結 - 10DLC 廣告活動包含內嵌連結。請勿使用 Tinyurl 或 Bitly 等常見 URL 縮短網站產生的短連結，但您可以使用提供自訂網域的 URL 縮短程式。
- Embedded phone number (內嵌電話號碼) - 行銷活動包含並非客戶支援號碼的內嵌電話號碼。
- 聯盟行銷 - 10DLC 活動包括來自聯盟行銷的資訊。
- 年齡限制內容 - 10DLC 活動包括電信業者和行動電信和網際網路協會 (CTIA) 規範所定義的年齡門禁內容。

7. 選擇 Create (建立)。

   您提交行銷活動的註冊詳細資訊後，SMS 和語音頁面會隨即開啟。此時會出現一則訊息，指出您的廣告活動已送出且正在審核中。您可以在 10DLC 廣告活動索引標籤上檢視請求狀態。您可以在 10DLC 索引標籤上確認登錄狀態，此標籤會是下列其中一項：

   - 作用中 - 您的 10DLC 廣告活動已獲得核准。您可以申請一組 10DLC 電話號碼，然後將該號碼與您的行銷活動建立關聯。如需更多詳細資訊，請參閱 透過 Amazon SNS 要求 10DLC 號碼、免付費電話號碼和 P2P 長代碼以進行簡訊傳送。
   - Pending (待定) – 您的 10DLC 行銷活動尚未獲得核准。在某些情況下，核准可能需要一週或更長時間。如果狀態改變，Amazon Pinpoint 主控台顯示的資訊也會隨之改變。我們不會通知您狀態變更。
   - Rejected (已拒絕) – 您的 10DLC 行銷活動已遭到拒絕。若要取得更多資訊，請提交支援要求，其中包含已拒絕之廣告活動的行銷活動 ID。
   - 已暫停 - 一或多家電信業者暫停您的 10DLC 廣告活動。若要取得更多資訊，請提交支援要求，其中包含已暫停之廣告活動的廣告活動 ID。Amazon Pinpoint 不會在主控台上加入暫停原因，而且如果您的廣告活動遭到暫停，我們也不會通知您。

8. 如果您的 10DLC 獲得核准，您可以要求一個 10DLC 號碼與該活動建立關聯。如需有關請求 10DLC 號碼的詳細資訊，請參閱 透過 Amazon SNS 要求 10DLC 號碼、免付費電話號碼和 P2P 長代碼以進行簡訊傳送。

在多個 AWS 區域使用 10DLC 行銷活動

您註冊公司後，可透過所有 AWS 區域 的 AWS 帳戶使用該公司，但 10DLC 行銷活動並非如此，10DLC 行銷活動只能在進行註冊的 AWS 區域使用，

因此如果您計劃在多個 AWS 區域使用 10DLC，就必須在每個區域分別註冊不同的 10DLC 行銷活動。為了遵守電信業者的要求，您必須執行這個步驟，而且即使您註冊的每場行銷活動使用案例都完全相同，您依然必須支付每場行銷活動的費用。

註冊多場行銷活動另有一項優點，就是針對使用 AT&T 行動電信業者的收件人，能夠提高發送訊息的輸送率，因為 AT&T 會分別為每場行銷活動提供 10DLC 輸送率。您可以比較 AT&T 與 T-Mobile 處理 10DLC 輸送量的方式，後者是計算每間公司每日訊息的分配量 (無論有多少場行銷活動)。

編輯或刪除 10DLC 廣告活動

您可以使用 Amazon Pinpoint 主控台編輯 10DLC 行銷活動的 HELP 回應、STOP 回應和範例訊息，也可以使用主控台刪除 10DLC 行銷活動。

編輯 10DLC 行銷活動

您的行銷活動獲得核准後，您就可以修改 HELP、STOP 與範例訊息，也可以新增其他範例訊息。變更這些欄位不需經過活動登記處或電信業者再次核准。一旦 10DLC 行銷活動獲得核准，您便無法再修改任何其他欄位。

您最多可以設定五則範例訊息，但無法減少最初註冊的範例訊息數量。舉例來說，如果您註冊行銷活動時使用了三則範例簡訊，便無法將範例簡訊的數量減少到少於三則。

> **Note**
>
> 如要修改 HELP、STOP、範例訊息以外的任何欄位，您必須先刪除 10DLC 行銷活動，再重新建立行銷活動，才能加入更新資訊。

如何編輯 10DLC 行銷活動

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 10DLC Campaigns (10DLC 行銷活動) 索引標籤上，選擇您要編輯的 10DLC 行銷活動。
4. 在行銷活動詳細資訊頁面的 Campaign messages (行銷活動訊息) 區段，選擇 Edit (編輯)。

5. 更新以下任何欄位：

   - Help 訊息
   - Stop 訊息
   - 範例簡訊

   您無法刪除先前加入的範例訊息，也不能刪除範例訊息的內容讓欄位留白。如果您刪除了訊息內容但沒有改換成新內容，更新時將使用原始訊息。
6. 選擇 Update (更新)。畫面上會隨即顯示確認橫幅，通知您行銷活動訊息已更新。

刪除 10DLC 行銷活動

您可以使用 Amazon Pinpoint 主控台刪除 10DLC 行銷活動。刪除 10DLC 行銷活動之前，您必須先移除所有與該行銷活動相關聯的電話號碼。

> **Important**
>
> 行銷活動中的 10DLC 號碼一經移除，便無法再供您存取。此外，已刪除的 10DLC 行銷活動無法還原。

如何刪除 10DLC 行銷活動

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於 SMS and voice (簡訊和語音) 下方選擇 Phone numbers (電話號碼)。
3. 在 10DLC Campaigns (10DLC 行銷活動) 索引標籤上，選擇您要編輯的 10DLC 行銷活動。
4. 記下 Phone numbers (電話號碼)區段中與行銷活動相關聯的電話號碼。
5. 在 Phone numbers (電話號碼) 索引標籤上，依序選擇您要移除的 10DLC 號碼與 Remove phone number (移除電話號碼)。

   > **Note**
   >
   > 唯有在您有多組與行銷活動相關聯的 10DLC 電話號碼時，才需要執行此步驟。如果您只有一組與 10DLC 行銷活動相關聯的電話號碼，該號碼會顯示在 10DLC campaigns (10DLC 行銷活動) 索引標籤上。請注意顯示在該索引標籤上的號碼。

6. 在確認方塊中輸入 **delete**，然後選擇 Confirm (確認)。成功訊息會顯示在「SMS and voice」(簡訊和語音) 頁面頂端。
7. 對每組與行銷活動相關聯的 10DLC 號碼重複執行前兩個步驟。
8. 移除所有與 10DLC 行銷活動相關聯的號碼後，選擇 10DLC campaigns (10DLC 行銷活動) 索引標籤。
9. 選擇您想要刪除的 10DLC 行銷活動。
10. 在 10DLC campaign details (10DLC 行銷活動詳細資訊) 頁面右上角，選擇 Delete (刪除)。
11. 在確認方塊中輸入 **delete**，然後選擇 Confirm (確認)。成功訊息會顯示在「SMS and voice」(簡訊和語音) 頁面頂端。

將長代碼與 10DLC 廣告活動建立關聯

如果您有現有的長代碼，您可以提交支援要求，將該長代碼與目前的 10DLC 廣告活動建立關聯。您與 10DLC 廣告活動相關聯的長代碼只能與該廣告活動搭配使用，不能用於任何其他 10DLC 廣告活動。當您的長代碼正在移轉到 10DLC 時，您仍然可以使用它。然而，在獲得核准之前，您無法使用它進行任何 10DLC 廣告活動。

提交請求時，您需要：

- 與 10DLC 廣告活動相關聯的長代碼
- 10DLC 廣告活動 ID 與長代碼相關聯

> **Note**
>
> 您必須先登錄 10DLC 廣告活動，才能將任何長代碼與廣告活動建立關聯。如果您尚未建立並登錄 10DLC 廣告活動，請參閱 登錄 10DLC 廣告活動。

若要將長代碼指派給 10DLC

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在 Settings (設定) 下方，然後在 SMS and voice (簡訊和語音) 下方，選擇 Phone numbers (電話號碼) 索引標籤。
3. 選擇您要轉換為 10DLC 的長代碼。

4. 若要開啟支援中心，請選擇 Assign to 10DLC campaign (指派給 10DLC 廣告活動)。
5. 對於案例類型，請選擇Service limit increase (提升服務限額)。
6. 針對 Limit type (限制類型)，選擇 Pinpoint。
7. 在 Requests (請求) 區段中選擇 Region (區域)，Limit (限制) 則選擇 10 DLC - Associate existing US long code to 10DLC campaign (10 DLC - 將現有美國長碼與 10DLC 行銷活動建立關聯)。
8. 在案例描述的使用案例說明，請務必包含 10DLC 廣告活動 ID，以及您要建立關聯該廣告活動的長代碼數字。您可以在要求中加入多個長代碼，但您應該只加入一個廣告活動 ID。
9. 在 Contact options (聯絡人選項) 下，針對 Preferred contact language (偏好的聯絡語言)，選擇您與 AWS 支援團隊通訊時偏好使用的語言。
10. 針對 Contact method (聯絡方法)，選擇您與 AWS 支援團隊聯絡的慣用方法。
11. 選擇 Submit (提交)。

10DLC 跨帳戶存取

每組 10DLC 電話號碼都與單一 AWS 區域 中的單個帳戶相關聯。如想透過多個帳戶或在多個區域使用同一組 10DLC 電話號碼，可採取下列兩種做法：

1. 為每個 AWS 帳户註冊相同的公司和行銷活動；每一項註冊都要分別管理及收費。如果您為多個 AWS 帳號註冊同一間公司，您每日可傳送給 T-Mobile 客户的訊息數量將由所有帳號共用。
2. 在一個 AWS 帳戶中完成 10DLC 註冊流程，然後使用 AWS Identity and Access Management (IAM) 授予其他帳戶透過您的 10DLC 號碼發送訊息的許可。

> **Note**
>
> 這項做法能允許您真正跨帳戶存取 10DLC 電話號碼，不過請注意，系統會將您次要帳戶發送的訊息視為您主要帳戶發送的訊息，因此會根據主要帳戶計算配額和帳單金額，而非次要帳戶。

使用 IAM 政策設定跨帳戶存取

您可以使用 IAM 角色將其他帳戶與您的主帳戶建立關聯，然後在主要帳戶中將 10DLC 號碼存取權授予次要帳戶，亦即將主要帳戶的存取許可委派給次要帳戶。

如何在主要帳戶中授予 10DLC 號碼的存取權

1. 如果您尚未在主要帳戶中完成 10DLC 註冊流程，請先進行註冊，整個流程包含三個步驟：

- 登錄您的公司。如需詳細資訊，請參閱註冊您的公司或品牌以便使用 10DLC。
- 註冊您的 10DLC 行銷活動 (使用案例)。如需更多詳細資訊，請參閱 登錄 10DLC 廣告活動。
- 將電話號碼與您的 10DLC 行銷活動建立關聯。如需更多詳細資訊，請參閱 將長代碼與 10DLC 廣告活動建立關聯。

2. 在主要帳戶中建立 IAM 角色，用於允許其他帳戶對您的 10DLC 電話號碼呼叫 `Publish` API 操作。如需進一步瞭解如何建立角色，請參閱 IAM 使用者指南中的建立 IAM 角色。
3. 從主要帳戶使用 IAM 角色對其他所有需要使用 10DLC 號碼的帳戶委派存取許可並進行測試，例如從生產帳戶將存取許可委派給開發人員帳戶。如需進一步瞭解如何委派及測試許可，請參閱 IAM 使用者指南中的使用 IAM 角色委派跨 AWS 帳戶存取權。
4. 使用新角色，利用主帳號的 10DLC 號碼傳送訊息。如需進一步瞭解如何使用角色，請參閱《IAM 使用者指南》中的使用 IAM 角色。

取得有關 10DLC 註冊問題的資訊

在某些情況下，當您嘗試註冊公司或 10DLC 行銷活動時，可能會收到錯誤訊息。

公司註冊問題

您註冊公司時，會看到下列其中一種註冊狀態：Verified (已驗證) 或 Unverified (未驗證)。如果公司的註冊狀態為 Verified (已驗證)，代表貴公司已註冊成功，您可以開始建立 10DLC 行銷活動。

如果貴公司的註冊狀態為 Unverified (未驗證)，表示您提供的資訊有問題。Amazon Pinpoint 主控台會提供相關資訊，說明貴公司顯示為此註冊狀態的原因。

如何查看 10DLC 公司註冊的問題

1. 登入 AWS Management Console，開啟位於 https://console.aws.amazon.com/pinpoint/ 的 Amazon Pinpoint 主控台。
2. 在導覽窗格中，於簡訊下方選擇電話號碼。
3. 在 10DLC campaigns (10DLC 行銷活動) 索引標籤的行銷活動清單中，選擇您想要查找詳細資訊的公司名稱。
4. 公司詳細資訊頁面會列出相關資訊，說明在您的註冊申請中發現的問題。如果 Company info (區段) 中有任何欄位顯示警告符號，即表示註冊問題與該欄位中的資訊有關。

   請檢查您提供的資訊，確認所有欄位輸入的資訊都正確無誤。您可以在 Amazon Pinpoint 主控台編輯您的公司註冊資訊。如需進一步瞭解如何修改公司註冊詳細資訊，請參閱編輯或刪除註冊公司。

行銷活動註冊問題

註冊 10DLC 行銷活動時，在某些情況下可能會看到錯誤訊息。

如果您找不到註冊問題，可以前往 AWS Support 中心建立案例，請求提供更多資訊。使用下列程序建立 AWS Support 案例。AWS Support 團隊將提供您的 10DLC 行銷活動註冊遭到拒絕的原因相關資訊。

如何針對遭拒 10DLC 行銷活動提交資訊提供請求

1. 前往 https://console.aws.amazon.com/ 登入 AWS Management Console。
2. 在 Support (支援) 功能表上，選擇 Support Center (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 Pinpoint SMS。
5. 在 Requests (請求) 下，填寫以下部分：
   - 針對區域，選擇您嘗試註冊行銷活動的 AWS 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 Resource Type (資源類型)，選擇 10DLC Registration (10DLC 註冊)。
   - 在限制，選擇拒絕公司或 10DLC 行銷活動註冊。
6. 在新增限制值中，選擇限制類型的限制增加。通常此值是 **1**。
7. 在案例說明一欄，輸入遭拒的 10DLC 行銷活動 ID。
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 使用 Amazon SNS 請求簡訊支援中的其他小節。
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 Submit (提交)。

免付費電話號碼

免付費電話號碼 (TFN) 是以下列其中一個區碼開頭的 10 位數號碼：800、888、877、866、855、844 或 833。您可以使用 TFN，僅傳送交易式訊息。

Important

美國行動電信業者最近已變更其規定，要求所有免付費電話號碼 (TFN) 必須在 2022 年 9 月 30 日前向監管機構完成註冊程序。請前往 the section called “免付費電話號碼註冊狀態” 查看您的 TFN。如需進一步瞭解如何註冊公司，請參閱 the section called “註冊免付費電話號碼”。
提交註冊後，最長需要 15 個工作天才能處理完畢。
2023 年 3 月 3 日更新：自 2023 年 4 月 1 日起，行動電信業者將針對透過任何未註冊免付費電話號碼傳送的訊息套用下列產業範圍的閾值：

- 每日限制：每日 5000 則訊息，於上午 12 時重設
- 每週限制：每週 1,000 則訊息，於星期日上午 12 時重設
- 每月限制：2,000 則訊息，於日曆月底於太平洋時間中午 12:00 重設

2022 年 9 月 19 日更新：自 2022 年 10 月 1 日起，行動電信業者將針對透過任何未註冊免付費電話號碼傳送的訊息套用下列產業範圍的閾值：

- 每日限額：2,000 則訊息
- 每週限額：12,000 則訊息
- 每月限額：25,000 則訊息

我們強烈建議您盡快完成註冊。透過未註冊 TFN 的訊息會盡可能傳送所有訊息。隨著電信業者持續限制未註冊的流量，透過未註冊 TFN 傳送的訊息可能隨著時間遭受更多篩選和封鎖。

主題

- 免付費電話號碼常見問答集
- 免付費電話號碼的優點和缺點

免付費號碼的使用指引

TFN 通常用於交易式訊息，例如註冊確認或傳送一次性密碼，僅於美國境內使用。免付費電話號碼可用於語音訊息傳送和簡訊。平均輸送量是每秒三個訊息部分 (MPS)。不過，此輸送量會受到字元編碼的影響。如需字元編碼如何影響訊息部分的詳細資訊，請參閱 Amazon SNS 中的簡訊字元限制。如需進一步瞭解如何註冊 TFN，請參閱 免付費電話號碼註冊要求和程序。

每個客戶帳戶最多可有 5 個 TFN。如果您每秒傳送超過 15 則但少於 100 則訊息，我們建議您註冊一或多個 10DLC 原始 ID。如果您的使用案例需要每秒傳送超過 100 則訊息，我們建議您購買並註冊一或多個短代碼。

使用 TFN 作為起始號碼時，請遵循下列準則：

- 請勿使用從第三方 URL 縮短程式建立的縮短 URL，因為這些郵件更有可能被篩選為垃圾郵件。

  如果您需要使用縮短的 URL，請考慮使用 10DLC 號碼或短代碼。使用短代碼和 10DLC 需要您登錄訊息範本，您可以在其中指定縮短的 URL。
- 請注意，關鍵字退出 (STOP) 和選擇加入 (UNSTOP) 回應是在電信業者層級設定。您無法修改這些關鍵字或其他任何關鍵字。您也無法修改當使用者使用 STOP 和 UNSTOP 回覆時所傳送的訊息。
- 請勿使用多個 TFN 傳送相同或類似的訊息內容。營運商稱這種做法雪鞋或號碼集區並鎖定這些郵件以進行篩選。
- 與以下行業相關的任何消息都可能被視為受限制，並且受到重度過濾或直接阻止。這可能包括與受限類別相關的服務的一次性密碼 （ OTP) 和多重要素驗證 (MFA)。

  如果您因為不合規的使用案例而遭到拒絕註冊，且您覺得此指定不正確，您可以透過支援服務提交請求。如需詳細作法，請參閱註冊問題。

  下表描述這些受限制內容的類型：

類別	範例
賭博	• 應用程式/網站 • 賭場 • 抽獎活動

類別	範例
高風險金融服務	• 汽車貸款 • 加密貨幣 • 收集債務 • 發薪日貸款 • 短期高息貸款 • 抵押貸款 • 學生貸款 • 股票提醒
債務寬免	• 合併債務 • 減少債務 • 修復信用計畫
G et-rich-quick 方案	• W ork-from-home 程序 • 風險投資機會 • 金字塔或多層次行銷計畫
禁止/受管制物品	• 大麻/CBD
網路釣魚	• 試圖讓使用者透露個人資訊或網站登入資訊
S.H.A.F.T.	• 性 • 仇恨 • 酒精 • 槍械 • 菸草/電子菸

購買免付費電話號碼

要購買 TFN，請使用 Amazon Pinpoint 控制台 https://console.aws.amazon.com/sms-voice/。如需詳細資訊，請參閱 免付費電話號碼註冊要求和程序。

目前，Amazon Pinpoint 簡訊同時支援語音和簡訊的免付費電話號碼。Amazon SNS 僅支援簡訊。

免付費電話號碼註冊要求和程序

> **Important**
>
> 如果 TFN 用於指定使用案例以外的任何事項，則該 TFN 可能會遭到撤銷。

免付費電話號碼禁止使用案例

Amazon SNS 在訊息遭到封鎖的狀況下傳送訊息的能力有限 (例如，與受控物質或網路釣魚相關的使用案例)，或預期進行高度篩選 (例如，高風險財務訊息)。您可能無法註冊與定義於 免付費號碼的使用指引 中之受限內容使用案例相關聯的 TFN。

註冊免付費電話號碼

購買 TFN 後，您必須註冊該號碼。如需有關如何執行此操作的指示，請參閱 Amazon Pinpoint 位 SMS 使用者指南中的免付費號碼註冊程序。

Amazon Pinpoint 簡訊區域中的免付費電話號碼自助註冊

如果您已在 Amazon 精確定位簡訊區域中要求 TFN，請使用 Amazon 精確點簡訊使用者指南中的美國免付費電話號碼註冊表單中的指示，直接在 AmazonPinpoint SMS 主控台中完成公司註冊程序。

註冊 TFN 時，請確保資訊完整準確，否則您的註冊可能會遭到拒絕。您輸入的資訊應與您公司的企業總部完全相符。

針對 Amazon 精確 SMS 區域以外地區的免付費電話號碼進行手動表單註冊程序

1. 下載此 US_TFN_Registration.zip 並使用示例註冊表格（AWS 美國免費電話註冊表-企業-Final.docx）填寫 TFN 註冊 CSV 文件中的所需信息（BulkustFN-Final.csv）。

   每個註冊請求或使用案例最多只可有五個 TFN。如果您認為自己符合此規則的豁免資格，請提供詳細說明以供考慮。列出與註冊或使用案例相關聯的所有電話號碼。
2. 建立具有 AWS 支援的案例。將您填寫的 CSV 檔案附加至案例中，並提交 TFN 註冊請求。
3. 選擇建立案例，然後選擇尋找提高服務限制？
4. 針對限制類型，選擇 SNS 簡訊。
5. 對於 資源類型，選擇 10DLC or Toll-free number registration (10DLC 或免付費電話號碼註冊)。
6. 附加 US_TFN_registration 文件並 提交 請求。

須知事項

1. 提交所有必要資訊後，註冊最多可能需要兩週的時間進行處理。若資訊遺失或不完整，則註冊過程將遭到延遲。若您的註冊遭到拒絕，我們將會協助您找出遭拒的原因，並建議改善行銷活動的方法，以便進行註冊。
2. TFN 適用於交易性使用案例，如需要有限輸送量的多重要素驗證 (MFA)。每個 TFN 每秒最多可傳送三個文字訊息部分，每個客戶帳戶最多可有五個 TFN。如果您每秒傳送超過 15 則但少於 100 則訊息，我們建議您註冊一或多個 10DLC 原始 ID。如果您的使用案例需要每秒傳送超過 100 則訊息，我們建議您購買並註冊一或多個短代碼。如需詳細資訊，請參閱免付費號碼的使用指引。

免付費電話號碼註冊狀態

若要檢查您的註冊狀態，請參閱 Amazon Pinpoint 位簡訊使用者指南中的檢查註冊狀態。

編輯、捨棄及刪除您的註冊

使用 Amazon Pinpoint 簡訊使用者指南來執行下列任務：

- 編輯您的註冊
- 放棄您的註冊
- 刪除您的註冊
- 檢視您的註冊資源

註冊問題

如果您的免費電話號碼註冊不被接受，您將看到一條消息，說明該號碼被拒絕的原因。

如何針對遭拒免付費號碼資訊提出請求

1. 請登入電子郵件 AWS Management Console 至 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 Pinpoint SMS。
5. 在請求下，填寫以下部分：
    - 您嘗試註冊行銷活動的區域。

> **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。

- 對於資源類型，選擇 10DLC 註冊。
- 在限制，選擇拒絕公司或行銷活動註冊。

6. 在新增限制值中，選擇限制類型的限制增加。通常此值是 **1**。
7. (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。如需所需的資訊，請參閱 使用 Amazon SNS 請求簡訊支援中的其他小節。
8. 在案例說明，輸入遭拒的免付費電話號碼。
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。

免付費電話號碼常見問答集

關於 TFN 註冊程序的常見問答集。

我目前是否擁有免付費電話號碼？

To check if you own a toll-free number (檢查您是否擁有免費電話號碼)

- 打開 Amazon Pinpoint 位短信控制台 https://console.aws.amazon.com/sms-voice/.
- 在導覽窗格中，於簡訊下方選擇電話號碼。
- 將 TFN type (類型) 列為 toll-free (免付費)。

我必須註冊我的免付費電話號碼嗎？

是。如要繼續使用您目前擁有的 TFN，您必須在 2022 年 9 月 30 日前進行註冊。如果您在 2022 年 9 月 30 日之後購買新的 TFN，則必須先進行註冊才能傳送訊息。

如何購買免付費電話號碼？

請依照使用 Amazon 精確簡訊主控台要求電話號碼中的指示購買 TFN。

如何註冊我的免付費電話號碼？

依照 the section called “註冊免付費電話號碼” 的指示，註冊一個 TFN。

我的免付費電話號碼註冊狀態為何？該狀態代表什麼？

按照 the section called “免付費電話號碼註冊狀態” 指示檢查您的註冊和狀態。

我需要提供哪些資訊？

您需要提供貴公司的地址、公司聯絡人和 TFN 的使用案例。您可以在 the section called “註冊免付費電話號碼” 找到所需資訊。

如果我的註冊遭到拒絕怎麼辦？

若您的註冊遭到拒絕，則狀態會變更為 Requires Updates (需要更新)。若要進行更新，請參閱 the section called “編輯、捨棄及刪除您的註冊”。

我需要哪些許可？

您用來造訪 Amazon Pinpoint SMS 主控台的 IAM 使用者/角色必須具有 ######*# 權限，否則您會收到存取遭拒的錯誤訊息。

免付費電話號碼的優點和缺點

優點

免付費發送者在長代碼中具有更高的 MPS 以及良好的交付能力。

缺點

由於這些在電信業者層級進行管理，因此無法控制選擇退出和選擇加入。

請勿在訊息中包含縮短的 URL，也請勿使用該號碼傳送促銷訊息。請改用 10DLC 號碼或短代碼。當您使用短代碼或 10DLC 號碼時，您需要註冊訊息範本，其中可以包含縮短的 URL，且可為促銷消息。如需短代碼的相關詳細資訊，請參閱 短代碼。如需 10DLC 的相關詳細資訊，請參閱 10DLC。

短代碼

短代碼是比一般電話號碼更短的數值序列。例如，在美國和加拿大，標準電話號碼 (長代碼) 包含 11 位數，而短代碼包含 5 或 6 位數。Amazon SNS 支援專用的短代碼。

專用短代碼

如果您將大量簡訊傳送給美國或加拿大的收件人，您可以購買專用短代碼。與共用集區中的短代碼不同，專用短代碼保留供您專用。

優點

使用記得住的短代碼有助於建立信任。如果您需要傳送敏感資訊，例如一次性密碼，最好使用短代碼來傳送，讓您的客戶可以快速判斷訊息是否確實來自於您。

如果您在進行開拓新客戶活動，您可以邀請潛在客戶將關鍵字傳送到您的短碼 (例如，以簡訊傳送 FOOTBALL 到 10987 以接收足球新聞和資訊)。短代碼比長代碼更容易記住，客戶在其裝置中輸入短代碼也較為輕鬆。只要降低客戶註冊您的行銷活動時所遇到的困難，就能提高行銷活動的成效。

由於行動電信業者必須先核准新的短代碼才能讓短代碼生效，所以他們不太可能將從短代碼傳送的訊息標示為來路不明。

當您使用短代碼來傳送簡訊時，您在每 24 小時期間內可傳送的訊息量會高於使用其他類型的來源身分。換言之，您有很高的傳送份額。此外，您每秒可以傳送更大量的訊息。也就是說，您有很高的傳送速率。

缺點

取得短代碼需要額外的成本，而且需要較長的時間來實作。例如，在美國，每個短代碼的一次性設定費為 650.00 USD，另外每個短代碼每月還有經常性費用 995.00 USD。可能需要 8-12 週，短代碼才會在所有電信業者網路上生效。若要尋找其他國家或區域的價格和佈建時間，請完成 針對使用 Amazon SNS 的簡訊請求專用的短碼 中所述的程序。

人對人 (P2P) 長代碼

> **Important**
>
> 以下規定自 2023 年 8 月 31 日起生效：須有專用號碼 (如 10DLC 號碼或免付費電話才能傳送簡訊至美國及其領土 (波多黎各、關島、美屬薩摩亞群島和 US 維京群島)。如果您使用美國做為這些地區的位置，您的長碼請求將會被拒絕。

> **Important**
>
> 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。反之，您必須使用另一種類型的起始 ID 做為這些訊息。如需更多詳細資訊，請參閱 10DLC。

P2P 長代碼是採用收件人所在國家或區域的數字格式的電話號碼。P2P 長代碼也稱為完整號碼或虛擬手機號碼。例如，在美國和加拿大，P2P 長代碼包含 11 位數：數字 1 (國碼)、三位數區碼，以及七位數電話號碼。

如需有關請求 P2P 長代碼的詳細資訊，請參閱 透過 Amazon SNS 要求 10DLC 號碼、免付費電話號碼和 P2P 長代碼以進行簡訊傳送。

優點

專用 P2P 長代碼保留僅供 Amazon SNS 帳戶使用，不與其他使用者共用。當您使用專用 P2P 長代碼時，您可以指定傳送每則訊息時要使用的 P2P 長代碼。如果您傳送多則訊息給相同的客戶，您可以確保每個訊息都顯示為從相同的電話號碼傳送。因此，專用 P2P 長代碼有助於建立您的品牌或身分。

缺點

以 US 為目的地的 A2P 通訊不支援 P2P 長碼。

如果您每天從專用 P2P 長代碼傳送數百則訊息，行動電信業者可能會認為您的號碼傳送來路不明的訊息。如果您的 P2P 長代碼被打上標記，則您的訊息可能不會傳遞給收件人。

P2P 長代碼的輸送量也受限。最高傳送速率會因國家或區域而不同。如需詳細資訊，請聯絡 AWS Support。如果您打算傳送大量簡訊，或以大於每秒一則訊息的速率傳送，您應該購買專用短代碼。

部分電信業者不允許您使用 P2P 長代碼傳送 A2P 短訊，包括美國地區。A2P 簡訊是當客戶將其手機號碼提交到應用程式時，傳送到客戶行動裝置的訊息。A2P 訊息是單向交談，例如行銷訊息、一次性密碼和約會提醒。如果您打算傳送 A2P 訊息，您應該購買專用短代碼 (如果您的客戶位於美國或加拿大)，或使用寄件者 ID (如果您的收件人位於支援寄件者 ID 的國家或區域)。

10DLC 號碼僅用於在美國境內傳送訊息。使用 10DLC 號碼時，您必須註冊公司品牌以及要與該號碼建立關聯的行銷活動。一旦受核准，您可以在 Amazon Pinpoint 主控台 https://console.aws.amazon.com/pinpoint/ 的簡訊和語音頁面上申請 10DLC 電話號碼。一旦提出申請，需 7-10 天才會獲得批准。此號碼無法用於任何其他行銷活動。

美國產品編號比較

此表格顯示美國電話號碼類型的支援比較。

產品功能	短代碼	免付費號碼	10DLC
數字格式	5-6 位數字	10 位數字	10 位數字
通道支援	簡訊	簡訊	簡訊
簡訊流量類型	促銷與交易	交易	促銷與交易
需要審核	是	否	是
預估佈建時間	12 週[1]	15 個工作日	1 週
簡訊傳輸量 (每秒的簡訊數量)[2]	每秒 100 個訊息部分；需額外付費提供更高的輸送量。	每秒 3 個訊息部分	根據您的 10DLC 註冊而有所不同。每秒支援多達 100 個訊息部分。
必要的關鍵字	選擇加入、選擇退出和說明	停止，取消停止。這些都是由網路管理的。您無法變更選擇退出，並在訊息中選擇退出。	選擇加入、選擇退出和說明

[1]佈建估計不包括核准時間。

[2]如需有關簡訊大小上限的詳細資訊，請參閱 發佈至行動電話。

使用 Amazon SNS 請求簡訊支援

Amazon SNS 的某些簡訊選項在您聯絡 AWS Support 前不適用於您的 AWS 帳戶。在 AWS Support 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。如需更多詳細資訊，請參閱 移出簡訊沙盒。
- 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。如需更多詳細資訊，請參閱 支援的國家和區域。

在 AWS Support 中心建立您的案例時，請包括您要提交的請求類型所需的所有資訊。否則，AWS Support 會聯絡您取得此資訊，之後才會繼續。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。

主題

- 針對使用 Amazon SNS 的簡訊請求專用的短碼
- 透過 Amazon SNS 要求 10DLC 號碼、免付費電話號碼和 P2P 長代碼以進行簡訊傳送
- 使用 Amazon SNS 請求簡訊寄件者 ID
- 為 Amazon SNS 請求提升您的每月簡訊花費配額

針對使用 Amazon SNS 的簡訊請求專用的短碼

短碼是可用於傳送大量簡訊的號碼。簡短代碼經常被用於應用程式至人員 (A2P) 簡訊、雙重身分驗證 (2FA) 和行銷。依據所在國家/地區而定，短碼通常包含三至七位數。

如果使用短碼，您只能將訊息傳送至短碼所在國家/地區的收件人。如果您的使用案例需要您在多個國家/地區使用短碼，您必須為您的收件人所在的每個國家/地區請求個別的短碼。

如需短碼定價的資訊，請參閱 Amazon SNS 定價。

> **Important**
>
> 如果您是第一次使用 Amazon SNS 處理簡訊，應請求符合您簡訊使用案例預期需求的每月簡訊花費閾值。在預設情況下，您的每月花費閾值為 1.00 USD (美元)。您可以在包含請求短碼的相同支援案例中請求提高花費閾值。或者，您可以使用單獨的案例。如需更多詳細資訊，請參閱 為 Amazon SNS 請求提升您的每月簡訊花費配額。

此外，如果您要求專用短碼來傳送將會或可能包含受保護的醫療資訊 (PHI) 的訊息，您應該在開啟支援案例時，於 Case description (案例描述) 中識別此目的，詳細說明如下。

開啟 Amazon SNS 短碼支援案例

完成下列步驟來使用 AWS Support 開立案例。

> **Note**
>
> 申請表中的某些欄位標示為「選用」。不過，AWS Support 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。

請求專用短碼

1. 前往 AWS 支援中心。
2. 前往 https://console.aws.amazon.com/ 登入 AWS Management Console。
3. 在 Support (支援) 功能表上，選擇 Support Center (支援中心)。
4. 在您的支援案例窗格中，選擇建立案例。
5. 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，然後完成以下操作：
   - (選用) 對於提供將傳送簡訊的網站或應用程式連結，請提供有關將發送簡訊的網站、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。
     - One Time Password (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) 對於您要傳送訊息的 AWS 區域來源，請選擇您要傳送訊息的區域來源。
   - (選用) 對於您打算傳送訊息的國家/地區，輸入您計劃向其發送簡訊的國家或地區。
   - (選用) 在您的客戶如何選擇接收您發送的訊息中，提供客戶如何選擇接收您發送的訊息說明。
   - (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。

6. 在 Requests (請求) 下，填寫以下部分：

   - 對於區域，為您的短碼請求選擇 AWS 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。

   - 對於 Resource Type (資源類型)，選擇 Dedicated SMS Short Codes (專用的簡訊短碼)。
   - 對於 Limit (限制)，請選擇與您的使用案例最類似的選項。
7. 對於新的限制值，選擇您要請求的寄件者 ID 數量。通常此值是 **1**。
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。如需所需的資訊，請參閱 使用 Amazon SNS 請求簡訊支援中的其他小節。
9. 在案例說明下，摘要說明您的使用案例，並包含您的收件人將如何使用您的短碼註冊您所傳送的訊息，並提供以下資訊。

   - 公司資訊：
     - 公司名稱
     - 公司郵寄地址
     - 您的請求的主要聯絡人姓名和電話號碼
     - 貴公司的電子郵件地址和免付費支援專線
     - 公司稅務 ID
     - 您的產品或服務的名稱
   - 使用者註冊程序：
     - 公司網站，或您的客戶將註冊以透過您的短碼接收訊息的網站。
     - 使用者將如何註冊，以透過您的短碼接收訊息。指定下列選項中的一或多個：
       - **`Text messages`**
       - **`Website`**
       - **`Mobile app`**
       - **`Other`** 如果有其他事項，請在此說明。
     - 在您的網站、應用程式或其他位置上註冊取得訊息的選項文字。

- 您計劃用於雙重選擇接收的訊息序列。請提供下列所有資訊：
  1. 您計劃在使用者登入時傳送的簡訊。此訊息會要求使用者同意重複的訊息。例如：ExampleCorp：回覆 YES 可接收帳戶交易提醒。適用訊息和資料傳輸費率。
  2. 您預期來自使用者的選擇接收回應。這通常是關鍵字，例如 YES。
  3. 當客戶傳送此關鍵字到您的短碼時，您要傳送的確認訊息。例如：您現在已註冊取得來自 ExampleCorp 的帳戶提醒。適用訊息和資料傳輸費率。請傳送 STOP 來取消或 HELP 來取得資訊。

- 您的訊息的目的：
  - 您計劃使用短碼傳送訊息的目的。指定下列其中一個選項：
    - **Promotions and marketing**
    - **Location-based services**
    - **Notifications**
    - **Information on demand**
    - **Group chat**
    - **Two-factor authentication (2FA)**
    - **Polling and surveys**
    - **Sweepstakes or contests**
    - **Other** 如果有其他事項，請在此說明。
  - 您是否計劃使用短碼傳送非您所擁有之公司的促銷或行銷訊息。
  - 您是否計劃使用短碼來傳送將會或可能包含受保護的醫療資訊 (PHI) (依美國健康保險流通與責任法案 (HIPAA) 及相關法規所定義) 的訊息。

- 訊息內容：
  - 當客戶將特定的關鍵字傳送給您以選擇接收訊息時，您計劃傳送的訊息。指定此關鍵字和訊息時請注意變更此訊息可能需要數週時間。當我們建立您的短碼時，我們會向您要使用短碼所在國家/地區的行動電話電信業者註冊該關鍵字和訊息。您的訊息可能類似下列範例：歡迎使用 *ProductName* 提醒！適用訊息和數據傳輸費率。每個月 *2* 則訊息。回覆 HELP 來取得說明，STOP 來取消。
  - 當客戶以關鍵字 HELP 回覆您的訊息時，您希望傳送的回應內容。此訊息必須包含客戶支援聯絡資訊。例如：*ProductName* 提醒：如需說明請至 *example.com/help* 或撥打 *(800) 555-0199*。適用訊息和數據傳輸費率。每個月 *2* 則訊息。回覆 STOP 來取消。

- 當客戶以關鍵字 STOP 回覆您的訊息時，您希望傳送的回應內容。此訊息必須確認您的使用者將不再接收來自您的訊息。例如：您已取消訂閱 *ProductName* 提醒。不會再傳送訊息。回覆 HELP 來取得說明或 *(800) 555-0199*。
- 您計劃用於傳送的文字，以定期提醒使用者已訂閱您的訊息。例如：提醒：您已向 ExampleCorp 訂閱帳戶提醒。適用訊息和資料傳輸費率。請傳送 STOP 來取消或 HELP 來取得資訊。
- 您計劃使用短碼傳送的每個類型訊息的範例。至少提供三個範例。如果您要傳送超過三種類型的訊息，請提供所有類型的範例。

> **Important**
>
> 無線通信業者要求我們提供上述的所有資訊，以佈建短碼。在您提供所有資訊之前，我們無法處理您的請求。

10. (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 使用 Amazon SNS 請求簡訊支援中的其他小節。
11. 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。
12. 完成後，請選擇 Submit (提交)。

我們收到您的請求後，會在 24 小時內提供初始回應。我們可能會與您聯絡以要求提供更多資訊。如果我們能夠提供您短碼，會將您在請求中指定之國家/地區取得短碼的成本相關資訊寄送給您。我們也提供在您所在國家/地區佈建短碼所需的預估時間。佈建短碼通常需要數週的時間，但依據短碼所在的國家/地區，這段時間可能會更短或更長。

> **Note**
>
> 一旦我方將您的短碼請求向電信業者提出，即會產生使用短碼的相關費用。無論短碼是否佈建完成，您都必須支付這些費用。

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，我們必須仔細考慮每個請求。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。

後續步驟

您已向無線電信業者註冊短碼，並在 Amazon SNS 主控台中檢閱您的設定。現在您可以使用 Amazon SNS 以您的短碼做為來源號碼傳送簡訊。

透過 Amazon SNS 要求 10DLC 號碼、免付費電話號碼和 P2P 長代碼以進行簡訊傳送

> **Important**
>
> 自 2021 年 6 月 1 日起，美國電信供應商不再支援使用 person-to-person (P2P) 長碼進行 application-to-person (A2P) 通訊至美國目的地。反之，您必須使用另一種類型的起始 ID 做為這些訊息。如需詳細資訊，請參閱 10DLC。

若要請求 10DLC 號碼、免付費電話號碼，以及 P2P 長代碼，請使用 Amazon Pinpoint 主控台。如需詳細說明，請參閱。請求號碼中的Amazon Pinpoint 使用者指南。

> **Important**
>
> 美國行動電信業者最近已變更其規定，要求所有免付費電話號碼 (TFN) 必須在 2022 年 9 月 30 日前向監管機構完成註冊程序。如需進一步瞭解如何註冊免付費電話號碼，請參閱 註冊免付費電話號碼。
>
> 如果您在 2022 年 9 月 30 日之前購買了免付費電話號碼，其將保持作用中狀態直到 2022 年 10 月 1 日為止，除非您已完成註冊、該註冊已傳回且狀態為已完成。否則，它將處於待定狀態，在您註冊該號碼前、註冊已傳回或註冊設定為作用中狀態前，您無法使用該號碼傳送訊息。
>
> 註冊最多需要 15 個工作日。

若要註冊您的 10DLC 公司和行銷活動，請在美國東部 (維吉尼亞北部) 區域開啟 Amazon Pinpoint 主控台。請使用「AWS Service Quotas」主控台建立服務限制提高案例，同時請求該區域的 10DLC 號碼，而不是要求 10DLC 號碼。如需 Amazon Pinpoint 可在其中使用之區域的詳細資訊，請參閱 AWS 一般參考 中的 Amazon Pinpoint 端點和配額。

> **Note**
>
> 如果您是第一次使用 Amazon SNS 處理簡訊，也應請求符合您簡訊使用案例預期需求的每月簡訊花費閾值。在預設情況下，您的每月花費閾值為 1.00 USD (美元)。如需更多詳細資訊，請參閱 為 Amazon SNS 請求提升您的每月簡訊花費配額。

使用 Amazon SNS 請求簡訊寄件者 ID

> **Important**
>
> 如果您是第一次使用 Amazon SNS 處理簡訊，應請求符合您簡訊使用案例預期需求的每月簡訊花費閾值。在預設情況下，您的每月花費閾值為 1.00 USD (美元)。您可以在包含您請求寄件者 ID 的相同支援案例中請求提高花費閾值。或者，如果您願意，您也可以開啟單獨的案例。如需更多詳細資訊，請參閱 為 Amazon SNS 請求提升您的每月簡訊花費配額。

在簡訊中，寄件者 ID 是出現在收件人裝置上，顯示為訊息寄件者的名稱。寄件者 ID 是向訊息收件人表明您身分的有用方法。

對寄件者 ID 的支援因國家而異。例如，美國的電信業者完全不支援寄件者 ID，但印度的電信業者卻要求寄件者使用寄件者 ID。如需支援寄件者 ID 的國家完整清單，請參閱 支援的國家和區域。

> **Important**
>
> 有些國家要求您先登錄寄件者 ID，才能使用它們傳送訊息。此登錄程序可能需要數週的時間，視國家而定。國家需要預先註冊寄件者 ID 的資料，都會顯示在支援的國家頁面上的表格。
> 您可以在多個 AWS 帳戶中使用和註冊相同的寄件者 ID，以取得簡訊。如果您有企業支援，並且正在跨多個範本註冊，請依照下列指示提出要求，並與您的技術客戶經理合作，以確保協調您的上線體驗。

如果您要向支援寄件者 ID 的國家的收件人傳送訊息，但該國家不需要您登錄寄件者 ID，您就不必執行任何額外的步驟。您可以立即開始傳送包含寄件者 ID 值的訊息。

如果您計劃向需要登錄寄件者 ID 的國家傳送訊息，您只需要完成本頁的程序。

步驟 1：開啟 Amazon SNS 簡訊案例

如果您計劃向需要寄件者 ID 的國家/地區的收件人傳送訊息，您可以在 AWS 支援中心建立新案例，以申請寄件者 ID。

> **Note**
>
> 如果您計劃向允許但不要求寄件者 ID 的國家/地區的收件人傳送訊息，即不需要在支援中心開啟案例。您可以立即開始傳送使用寄件者 ID 的訊息。

請求寄件者 ID

1. 前往 https://console.aws.amazon.com/ 登入 AWS Management Console。
2. 在 Support (支援) 功能表上，選擇 Support Center (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 Limit type (限制類型)，選擇 Pinpoint SMS。
   - (選用) 對於提供將傳送簡訊的網站或應用程式連結，找出對象成員選擇接收簡訊的網站或應用程式。
   - (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。
     - One Time Password (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) 對於您要傳送訊息的 AWS 區域來源，請選擇您要傳送簡訊的 AWS 區域 來源。
   - (選用) 對於您計畫傳送訊息的國家/地區，指定您要註冊寄件者 ID 的國家。對寄件者 ID 和寄件者 ID 註冊要求提供的支援因國家而異。如需更多詳細資訊，請參閱 支援的國家和區域。

     如果國家名單超過此文字方塊允許的字元數，您可以改為列出國家/地區中的案例說明區段。
   - (選用) 在您的客戶如何選擇接收您發送的訊息中，提供客戶如何選擇接收您發送的訊息說明。
   - (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，包含您將使用的任何訊息範本。
5. 在 Requests (請求) 下，填寫以下部分：

- 對於區域，為您的寄件者 ID 選擇 AWS 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 Resource Type (資源類型)，選擇 General Limits (一般限制)。
- 對於限制，選擇 SMS 生產存取權。

6. 對於新的限制值，選擇您要請求的寄件者 ID 數量。通常此值是 **1**。
7. (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。如需所需的資訊，請參閱 使用 Amazon SNS 請求簡訊支援中的其他小節。
8. 在 Case description (案例描述) 下，針對 Use case description (使用案例描述) 提供下列詳細資訊：
   - 您要註冊的寄件者 ID。
   - 您計劃要用於簡訊的範本。
   - 您計劃每月傳送給每個收件人的訊息數量。
   - 您的客戶選擇如何接收您訊息的資訊。
   - 您公司或組織的名稱。
   - 與您公司或組織相關聯的地址。
   - 您公司或組織所在的國家/地區。
   - 您公司或組織的電話號碼。
   - 您公司或組織的網站 URL。
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 Submit (提交)。

我們收到您的請求後，會在 24 小時內提供初始回應。我們可能會與您聯絡以要求提供更多資訊。如果我們能夠提供寄件者 ID 給您，我們會傳送其佈建所需的估計時間給您。

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，我們必須仔細考慮每個請求。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 主控台中更新您的簡訊設定

當我們完程取得寄件者 ID 的程序，我們會回覆您的案例。當您收到此通知時，請完成本節中的步驟，以設定 Amazon SNS 使用您的寄件者 ID 做為使用您帳戶傳送之所有郵件的預設寄件者 ID。或者，您可以選擇指定發布郵件時要使用的寄件者 ID。

1. 登入 Amazon SNS 主控台。
2. 在導覽面板上，選擇行動裝置，然後選擇簡訊 (SMS)。
3. 在簡訊偏好設定區段中，選擇編輯。
4. 在詳細資訊區段的預設寄件者 ID，輸入提供的寄件者 ID，做為帳戶中所有郵件的預設值。
5. 完成後，請選擇 Save changes (儲存變更)。

後續步驟

您已註冊寄件者 ID 並在 Amazon SNS 主控台中更新您的設定。現在您可以使用 Amazon SNS 來使用您的寄件者 ID 傳送簡訊。支援的國家/地區中的簡訊收件人，在其裝置上看到的訊息寄件者為您的寄件者 ID。如果發布郵件時使用不同的寄件者 ID，則會覆寫此處設定的預設 ID。

為 Amazon SNS 請求提升您的每月簡訊花費配額

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。如果您要申請調高帳戶的簡訊花費配額，則需在 AWS 支援中心開立配額提高案例。

步驟 1：開啟 Amazon SNS 簡訊案例

您可以在 AWS 支援中心開立配額提高案例，來申請調高您的每月花費配額。

> **Note**
>
> 申請表中的某些欄位標示為「選用」。不過，AWS Support 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. 前往 https://console.aws.amazon.com/ 登入 AWS Management Console。
2. 在 Support (支援) 功能表上，選擇 Support Center (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。
     - One Time Password (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) 對於您要傳送訊息的 AWS 區域來源，請選擇您要傳送訊息的區域來源。
   - (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 Requests (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 Resource Type (資源類型)，選擇 General Limits (一般限制)。
- 對於 Limit (限制)，選擇 Account Spend Threshold Increase (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 Case description (案例描述) 下，針對 Use case description (使用案例描述) 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 New quota value (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 使用 Amazon SNS 請求簡訊支援中的其他小節。
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 Submit (提交)。

AWS Support 團隊會在 24 小時內對您的請求提供初步回應。

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)。
4. 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。

5. 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。如果仍然遇到問題，請在 AWS 支援中心開啟案例。

設定簡訊喜好設定

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 發佈至行動電話。

主題

- 設定 SMS 訊息偏好設定 AWS Management Console
- 設定偏好設定 (AWS SDK)

設定 SMS 訊息偏好設定 AWS Management Console

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 Mobile (行動裝置)、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 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**
>
> S3 儲存貯體政策必須將寫入存取授予 Amazon SNS。

8. 選擇儲存變更。

設定偏好設定 (AWS SDK)

若要使用其中一個開 AWS 發套件設定簡訊喜好設定，請使用該開發套件中對應於 Amazon SNS API 中的`SetSMSAttributes`請求的動作。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。

下列程式碼範例會示範如何使用`SetSMSAttributes`。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for C++ 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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的 SetSMSAttributes。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的 SetSMSAttributes。

傳送簡訊

本節說明如何傳送簡訊。

主題

- 發佈到主題
- 發佈至行動電話

發佈到主題

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。訂閱者將收到所有發佈至主題的訊息，除非您取消訂閱，或者訂閱者從您的 AWS 帳戶退訂接收簡訊。

主題

- 傳送訊息至主題 (主控台)
- 傳送訊息至主題 (AWS 開發套件)

傳送訊息至主題 (主控台)

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 AWS 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 帳戶位於簡訊沙盒。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。如需更多詳細資訊，請參閱 簡訊沙盒。

1. 登入 Amazon SNS 主控台。
2. 在導覽窗格中，選擇 Subscriptions (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 發佈至行動電話。

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。

傳送訊息至主題 (AWS 開發套件)

若要使用 AWS 開發套件，您必須使用您的認證進行設定。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

適用於 Java 2.x 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。

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);
        }
    }
}
```

發佈至行動電話

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 發佈到主題。

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 開發套件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。

主題

- 傳送訊息 (主控台)
- 傳送訊息 (AWS SDK)

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 AWS 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 ID 的國家和區域，請參閱 支援的國家和區域。

     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。此字串必須符合您的 AWS 帳戶 中為目的地國家設定的來源號碼。起始號碼可以是 10DLC 號碼、免費電話號碼、 person-to-person 長碼或短碼。如需詳細資訊，請參閱 簡訊的來源身分。

     如果您未指定起始號碼，Amazon SNS 會根據您 AWS 帳戶 的組態選取 SMS 文字訊息使用的起始號碼。
9. 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 印度的寄件者 ID 註冊要求。

10. 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 SDK)

若要使用其中一個開 AWS 發套件傳送簡訊，請使用該開發套件中對應於 Amazon SNS API 中的`Publish`請求的 API 作業。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 ID 的國家或區域列表，請參閱 支援的國家和區域。

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。字串必須與您 AWS 帳戶中針對目的地國家/地區設定的原始號碼相符。起始號碼可以是 10DLC 號碼、免費電話號碼、 person-to-person (P2P) 長碼或短碼。如需詳細資訊，請參閱 來源號碼。

如果您未指定起始編號，Amazon SNS 會根據您的 AWS 帳戶組態選擇一個起始編號。

`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。

如果您的 month-to-date SMS 成本已經超過為屬性設定的配額，則此`MonthlySpendLimit`屬性沒有任何作用。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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for .NET API 參考中的發佈。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 * 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 AWS SDK for C++ API 參考中的發佈。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的發佈。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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。

監控簡訊活動

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。

主題

- 檢視簡訊傳遞統計
- 檢視簡訊傳遞的 Amazon CloudWatch 指標和日誌
- 檢視每日 SMS 用量報告

檢視簡訊傳遞統計

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。如需更多詳細資訊，請參閱 檢視每日 SMS 用量報告。

檢視簡訊傳遞的 Amazon CloudWatch 指標和日誌

您可以使用 Amazon CloudWatch 和 Amazon CloudWatch Logs 監控您的簡訊傳遞。

主題

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。如需為服務建立 IAM 角色的詳細資訊，請參閱 IAM 使用者指南中的建立 AWS 服務的角色。

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

檢視每日 SMS 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。SMS 用量報告需要 24 小時之後才能在 S3 儲存貯體中使用。

主題

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
- 目標電話號碼
- 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

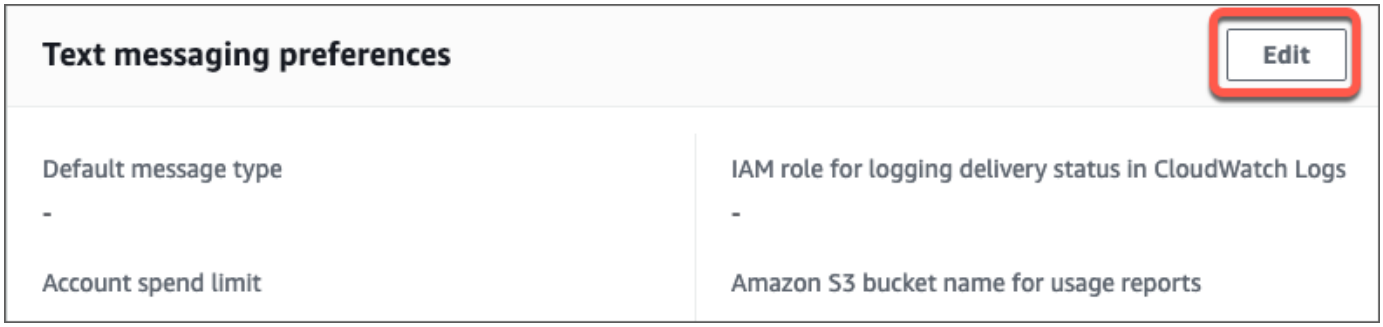
1. 從傳送簡訊的 AWS 帳戶，請登入 Amazon S3 主控台。
2. 選擇 Create Bucket (建立儲存貯體)。
3. 針對 Bucket Name (儲存貯體名稱)，建議您輸入帳戶和組織唯一的名稱。例如，使用樣式 `<my-bucket-prefix>-<account_id>-<org-id>`。

   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 Create (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。

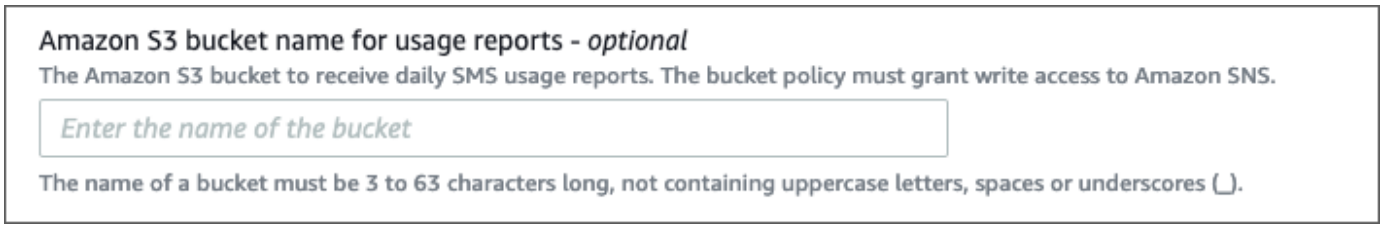
   如果您使用範例原則，請記得將 *my-s3-bucket* 取代為您在步驟 3 中選擇的儲存貯體名稱。
8. 選擇 Save (儲存)。

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)。



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)。



5. 選擇 Save changes (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 提供了各種工具，適用於具有已獲得帳戶中資源存取權之服務主體的所有服務。當 Amazon S3 儲存貯體政策陳述句中的委託人為 AWS 服務委託人，則可使用 `aws:SourceArn` 或 `aws:SourceAccount` 全域條件金鑰來防止混淆代理人問題。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:::my-s3-bucket/*",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   },
   {
    "Sid": "AllowGetBucketLocation",
    "Effect": "Allow",
    "Principal": {
     "Service": "sns.amazonaws.com"
    },
    "Action": "s3:GetBucketLocation",
    "Resource": "arn:aws:s3:::my-s3-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
    }
   },
   {
    "Sid": "AllowListBucket",
    "Effect": "Allow",
    "Principal": {
     "Service": "sns.amazonaws.com"
    },
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::my-s3-bucket",
    "Condition": {
     "StringEquals": {
      "aws:SourceAccount": "account_id"
     },
     "ArnLike": {
      "aws:SourceArn": "arn:aws:sns:region:account_id:*"
     }
```

```
    }
   }
  ]
}
```

> **Note**
>
> 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到另一個 AWS 帳戶 擁有的 Amazon S3 儲存貯體，請參閱如何從另一個 AWS 帳戶 複製 S3 物件?。

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,PartNumbe
2016-05-10T03:00:29.476Z,96a298ac-1458-4825-
a7eb-7330e0720b72,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.90084,0,1
2016-05-10T03:00:29.561Z,1e29d394-
d7f4-4dc9-996e-26412032c344,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.34322,0,1
2016-05-10T03:00:30.769Z,98ba941c-afc7-4c51-
ba2c-56c6570a6c08,1XXX5550100,Transactional,Message has been accepted by phone
 carrier,0.27815,0,1
```

管理電話號碼和簡訊訂閱

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。

主題

- 停止接收簡訊
- 管理電話號碼和簡訊訂閱 (主控台)
- 管理電話號碼和訂閱 (AWS SDK)

停止接收簡訊

依當地法律和法規 (例如美國和加拿大) 的要求，簡訊收件人可以使用其裝置透過下列項目回覆訊息來選擇停止：

- 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

若要停止接收，收件人必須回覆 Amazon SNS 用來傳送訊息的相同來源號碼。選擇退出後，除非您選擇輸入電話號碼， AWS 帳戶 否則收件人將不再收到您發送的 SMS 消息。

如果電話號碼訂閱到 Amazon SNS 主題，停止接收不會移除訂閱，但簡訊將無法傳遞到訂閱，除非您加入電話號碼。

管理電話號碼和簡訊訂閱 (主控台)

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。

加入已經停用的電話號碼

您可以檢視哪些電話號碼已經停止從您的帳戶接收簡訊，而您可以加入這些電話號碼以繼續傳送簡訊至這些電話號碼。

您只能每 30 天加入一次電話號碼。

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，選擇 View opted out phone numbers (檢視已停止接收的電話號碼)。Opted out phone numbers (已停止接收的電話號碼) 頁面會顯示已停止接收的電話號碼。
5. 選擇您要加入之電話號碼的核取方塊，然後選擇 Opt in (加入)。電話號碼就不再停止接收並且將接收您傳送來的簡訊。

刪除簡訊訂閱。

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。

管理電話號碼和訂閱 (AWS SDK)

您可以使用 AWS 開發套件向 Amazon SNS 發出程式設計請求，並管理哪些電話號碼可以從您的帳戶接收 SMS 訊息。

要使用 AWS SDK，您必須使用憑據對其進行配置。如需詳細資訊，請參閱 AWS SDK 和工具參考指南中的共用設定和認證檔案。

檢視所有已停止接收的電話號碼

若要檢視所有已停止接收的電話號碼，請使用 Amazon SNS API 提交 `ListPhoneNumbersOptedOut` 請求。

下列程式碼範例會示範如何使用`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[ListPhoneNumbersOptedOut](#)中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在[AWS 設定和執行程式碼範例儲存庫](#)。

```
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[ListPhoneNumbersOptedOut](#)中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListPhoneNumbersOptedOut中的。

檢查電話號碼是否已停止接收

若要檢查電話號碼是否已停止接收，請使用 Amazon SNS API 提交 `CheckIfPhoneNumberIsOptedOut` 請求。

下列程式碼範例會示範如何使用`CheckIfPhoneNumberIsOptedOut`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK for .NET API 參考CheckIfPhoneNumberIsOptedOut中的。

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已選擇不接收來自目前 AWS 帳戶的 SMS 訊息。

```
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CheckIfPhoneNumberIsOptedOut中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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CheckIfPhoneNumberIsOptedOut中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CheckIfPhoneNumberIsOptedOut中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CheckIfPhoneNumberIsOptedOut中的。

加入已經停用的電話號碼

若要加入電話號碼，請使用 Amazon SNS API 提交 `OptInPhoneNumber` 請求。

您只能每 30 天加入一次電話號碼。

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。

下列程式碼範例會示範如何使用`Unsubscribe`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考中的取消訂閱。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ 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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的取消訂閱。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的取消訂閱。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   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。

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。

下列程式碼範例會示範如何使用`DeleteTopic`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考DeleteTopic中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考DeleteTopic中的。

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DeleteTopic中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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



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

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteTopic中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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DeleteTopic中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteTopic中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteTopic中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考DeleteTopic中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件DeleteTopic中的 Python (博托 3) API 參考。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK DeleteTopic中的 SAP ABAP API 參考資料。

支援的國家和區域

> **Important**
>
> 以下規定自 2023 年 8 月 31 日起生效：須有專用號碼 (如 10DLC 號碼或免付費電話才能傳送簡訊至美國及其領土 (關島、波多黎各、美屬薩摩亞群島和 US 維京群島)。

目前，Amazon SNS 在下列 AWS 區域支援簡訊：

區域名稱	區域	端點	通訊協定
美國東部 (俄亥俄)	us-east-2	sns.us-east-2.amazonaws.com	HTTP 和 HTTPS
美國東部 (維吉尼亞北部)	us-east-1	sns.us-east-1.amazonaws.com	HTTP 和 HTTPS
美國西部 (加利佛尼亞北部)	us-west-1	sns.us-west-1.amazonaws.com	HTTP 和 HTTPS
美國西部 (奧勒岡)	us-west-2	sns.us-west-2.amazonaws.com	HTTP 和 HTTPS
非洲 (開普敦)	af-south-1	sns.af-south-1.amazonaws.com	HTTP 和 HTTPS
亞太區域 (海德拉巴)	ap-south-2	阿姆斯特丹南 2. 亚马孙	HTTP 和 HTTPS
亞太區域 (雅加達)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP 和 HTTPS
亞太區域 (墨爾本)	ap-southeast-4	SN.ap-東南部 4. 亞馬遜	HTTP 和 HTTPS

區域名稱	區域	端點	通訊協定
亞太區域 (孟買)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP 和 HTTPS
亞太區域 (大阪)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP 和 HTTPS
亞太區域 (新加坡)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP 和 HTTPS
亞太區域 (雪梨)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP 和 HTTPS
亞太區域 (東京)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP 和 HTTPS
加拿大 (中部)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP 和 HTTPS
歐洲 (法蘭克福)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP 和 HTTPS
歐洲 (愛爾蘭)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP 和 HTTPS
歐洲 (倫敦)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP 和 HTTPS
歐洲 (米蘭)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP 和 HTTPS
歐洲 (巴黎)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP 和 HTTPS
歐洲 (西班牙)	eu-south-2	南方小酒店 2. 亞馬遜	HTTP 和 HTTPS
歐洲 (斯德哥爾摩)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP 和 HTTPS

區域名稱	區域	端點	通訊協定
歐洲 (蘇黎世)	eu-central-2	小酒店-中央-2. 亞馬遜	HTTP 和 HTTPS
以色列 (特拉維夫)	il-central-1	sns.il-central-1.amazonaws.com	HTTP 和 HTTPS
中東 (巴林)	me-south-1	sns.me-south-1.amazonaws.com	HTTP 和 HTTPS
中東 (阿拉伯聯合大公國)	me-central-1	斯坦克我中心 1. 亞馬遜	HTTP 和 HTTPS
南美洲 (聖保羅)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP 和 HTTPS
AWS GovCloud (美國東部)	us-gov-east-1	SNS。 us-gov-east-1. 亞馬遜	HTTP 和 HTTPS
AWS GovCloud (美國西部)	us-gov-west-1	SNS。 us-gov-west-1. 亞馬遜	HTTP 和 HTTPS

您可以使用 Amazon SNS 傳送簡訊至下列國家和區域：

> **Note**
>
> 在支援的國家使用寄件者 ID 可改善簡訊傳送品質。

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
A						
阿富汗	AF	93	否	否	是	否
阿爾巴尼亞	AL	355	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
阿爾及利亞	DZ	213	否	否	是	否
安道爾	AD	376	否	否	是	否
安圭拉	AI	1-264	否	否	是	否
安地卡及巴布達	AG	1-268	否	否	是	否
阿根廷	AR	54	是	否	否	否
亞美尼亞	AM	374	否	否	是	否
阿魯巴島	AW	297	否	否	是	否
澳洲	AU	61	否	是	需要註冊[1]	是
奧地利	AT	43	是	是	是	是
亞塞拜然	AZ	994	否	否	是	否
B						
巴哈馬	BS	1-242	否	否	否	否
巴林	BH	973	否	否	是	否
孟加拉	BD	880	否	否	是	否
巴貝多	BB	1-246	否	否	是	否
白俄羅斯	BY	375	否	否	需要註冊[1]	否
比利時	BE	32	否	是	否	是
貝里斯	BZ	501	否	否	是	否
百慕達	BM	1-441	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
不丹	BT	975	否	否	是	否
玻利維亞	BO	591	否	否	是	否
波士尼亞與赫塞哥維納	BA	387	否	否	是	否
波札那	BW	267	否	否	是	否
巴西	BR	55	是	否	否	是
汶萊	BN	673	否	否	是	否
保加利亞	BG	359	是	否	是	是
布吉納法索	BF	226	否	否	是	否
蒲隆地	BL	257	否	否	是	否
C						
柬埔寨	KH	855	否	否	是	否
喀麥隆	CM	237	否	否	是	否
加拿大	CA	1	是	是	否	是
維德角	CV	238	否	否	是	否
開曼群島	KY	1-345	否	否	否	否
中非共和國	CF	236	否	否	是	否
查德	TD	235	否	否	是	否
智利	CL	56	是	是	否	是
中國	CN	86	是	否	否[2]	是

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
哥倫比亞	CO	57	是	是	否	是
葛摩	KM	269	否	否	是	否
庫克群島	CK	682	否	否	是	是
哥斯大黎加	CR	506	否	否	否	否
克羅埃西亞	HR	385	否	否	是	否
賽普勒斯	CY	357	否	否	是	否
捷克 (捷克共和國)	CZ	420	否	是	是	是
D						
剛果民主共和國	CD	243	否	否	是	否
丹麥	DK	45	是	是	是	是
吉布地	DJ	253	否	否	是	否
多米尼克	DM	1-767	否	否	是	否
多明尼加共和國	DO	1-809，1-829，1-849	是	否	否	是
E						
厄瓜多	EC	593	是	否	否	是
埃及	EG	20	是	否	需要註冊[1]	是
薩爾瓦多	SV	503	否	否	否	否
赤道幾內亞	GQ	240	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
厄利垂亞	ER	291	否	否	是	否
愛沙尼亞	EE	372	否	是	是	是
衣索比亞	ET	251	否	否	是	否
F						
法羅群島	FO	298	否	否	是	否
斐濟	FJ	679	否	否	是	否
芬蘭	FI	358	是	是	是	是
法國	FR	33	是	否	是	是
法屬圭亞那	GF	594	否	否	是	否
法屬玻里尼西亞	PF	689	否	否	是	否
G						
加彭	GA	241	否	否	是	否
甘比亞	GM	220	否	否	是	否
喬治亞	GE	995	否	否	是	否
德國	DE	49	是	是	是	是
迦納	GH	233	否	否	是	否
直布羅陀	GI	350	否	否	是	否
希臘	GR	30	否	否	是	否
格陵蘭	GL	299	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
格瑞那達	GD	1-473	否	否	是	否
瓜地洛普	GP	590	否	否	是	否
關島	GU	1-671	否	否	否	是
瓜地馬拉	GT	502	否	否	否	否
根西島	GG	44-1481	否	否	是	否
幾內亞	GN	224	否	否	是	否
幾內亞比索	GW	245	否	否	是	N/A
蓋亞納	GY	592	否	否	是	否
H						
海地	H	509	否	否	是	否
宏都拉斯	HN	504	否	否	是	否
香港	HK	852	否	是	是	是
匈牙利	HU	36	否	是	否	是
I						
冰島	IS	354	否	否	是	否
印度	IN	91	是	是[4]	需要註冊[3]	是
印尼	ID	62	否	否	是	否
伊拉克	IQ	964	否	否	是	否
愛爾蘭	IE	353	否	是	是	是
曼島	IM	44-1624	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
以色列	IL	972	否	是	是	是
義大利	IT	39	是	是	是	是
象牙海岸	CI	225	否	否	是	否
J						
牙買加	JM	1-876	否	否	是	否
日本	JP	81	是	是	是	是
澤西島	JE	44-1534	否	是	是	是
約旦	JO	962	否	否	需要註冊[1]	否
K						
哈薩克	KZ	7	否	否	是	否
肯亞	KE	254	否	否	是	否
科索沃	XK	383	否	否	是	否
科威特	KW	965	否	否	需要註冊[1]	否
吉爾吉斯	KG	996	否	否	是	否
L						
寮國	LA	856	否	否	是	否
拉脫維亞	LV	371	否	否	是	否
黎巴嫩	LB	961	否	否	是	否
賴索托	LS	266	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
賴比瑞亞	LR	231	否	是	否	
利比亞	LY	218	否	否	是	否
列支敦斯登	LI	423	否	否	是	否
立陶宛	LT	370	否	是	是	是
盧森堡	LU	352	否	是	是	是
M						
澳門	MO	853	否	否	是	否
馬其頓	MK	389	否	否	是	否
馬達加斯加	MG	261	否	否	是	否
馬拉威	MW	265	否	否	是	否
馬來西亞	MY	60	是	否	否	是
馬爾地夫	MV	960	否	否	是	否
馬利	ML	223	否	否	是	否
馬爾他	MT	356	否	否	是	否
馬紹爾群島	MH	692	否	否	否	否
馬丁尼克	MQ	596	否	否	是	否
茅利塔尼亞	MR	222	否	否	是	否
模里西斯	MU	230	否	否	是	否
馬約特島	YT	262	否	否	是	否
墨西哥	MX	52	是	否	否	是

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
密克羅尼西亞 (密克羅尼西亞聯邦)	FM	691	否	否	否	否
摩爾多瓦	MD	373	否	否	是	否
摩納哥	MC	377	否	否	否	否
蒙古	MN	976	否	否	是	否
蒙特內哥羅	ME	382	否	否	是	否
蒙特色拉特島	MS	1-664	否	否	是	否
摩洛哥	MA	212	是	否	是	是
莫三比克	MZ	258	否	否	否	否
緬甸	MM	95	否	是	是	是
N						
納米比亞	NA	264	否	否	是	否
尼泊爾	NP	977	否	否	是	否
荷蘭	NL	31	是	是	是	是
荷屬安地列斯	AN	599	否	否	是	否
新喀里多尼亞	NC	687	否	否	是	否
紐西蘭[6]	NZ	64	是	否	否	是

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
尼加拉瓜	NI	505	否	否	否	否
尼日	NE	227	否	否	是	否
奈及利亞	NG	234	否	否	是	否
紐埃島	NU	683	否	否	是	否
挪威	NO	47	否	是	是	是
O						
阿曼	OM	968	否	否	否	N/A
P						
巴基斯坦	PK	92	否	否	是	N/A
巴勒斯坦	PS	970	否	否	是	否
巴拿馬	PA	507	否	否	是	否
巴布亞紐幾內亞	PG	675	否	否	是	否
巴拉圭	PY	595	否	否	否	否
秘魯	PE	51	是	否	否	是
菲律賓	PH	63	否	是	需要註冊[1]	是
波蘭	PL	48	否	是	是	是
葡萄牙	PT	351	否	是	是	是
波多黎各	PR	1-797	否	否	否	是
Q						

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
卡達	QA	974	否	否	需要註冊[1]	否
R						
剛果共和國	CG	242	否	否	否	否
留尼旺 (法國)	RE	262	否	否	是	否
羅馬尼亞	RO	40	否	是	是	是
俄羅斯	RU	7	是	否	需要註冊[1]	是
盧安達	RW	250	否	否	是	否
S						
聖克里斯多福及尼維斯	KN	1-869	否	否	否	否
聖露西亞	LC	1-758	否	否	否	否
薩摩亞	WS	685	否	否	是	否
聖馬利諾	SM	378	否	否	是	否
聖多美普林西比	ST	239	否	否	是	否
沙烏地阿拉伯	SA	966	否	是[4]	需要註冊[1]	否
塞內加爾	SN	221	否	否	是	否
塞爾維亞	RS	381	否	否	是	否
賽席爾	SC	248	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
獅子山	SL	232	否	否	是	否
新加坡	SG	65	是	是	是[5]	是
斯洛伐克	SK	421	否	是	是	否
斯洛維尼亞	SI	386	否	否	是	否
索羅門群島	SB	677	否	否	是	否
索馬利亞	SO	252	否	否	是	否
南非	ZA	27	是	是	否	是
南韓	KR	82	否	否	否	否
南蘇丹	SS	211	否	否	是	否
西班牙	ES	34	是	是	是	是
斯里蘭卡	LK	94	否	否	需要註冊[1]	否
蘇利南	SR	597	否	否	是	否
史瓦濟蘭	SZ	268	否	否	是	否
瑞典	SE	46	是	是	是	是
瑞士	CH	4.1	否	是	是	是
T						
臺灣	TW	886	否	是	否	是
塔吉克	TJ	992	否	否	是	否
坦尚尼亞	TX	255	否	否	是	否

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
泰國	TH	66	否	是	需要註冊[1]	是
東帝汶	TL	670	否	否	是	否
多哥	TG	228	否	否	是	否
東加	TO	676	否	否	是	否
千里達及托巴哥	TT	1-868	否	否	是	否
突尼西亞	TN	216	否	否	是	否
土耳其	TR	90	否	否	需要註冊[1]	否
土庫曼	TM	993	否	否	否	否
英屬土克斯及開科斯群島	TC	1-649	否	否	是	否
吐瓦魯	TC	688	否	否	是	否
U						
烏干達	UG	256	否	否	是	否
烏克蘭	UA	380	否	是	是	是
阿拉伯聯合大公國	AE	971	是	是	需要註冊[1]	是
英國	GB	44	是	是	是	是
美國	US	1	是	是	否	是
烏拉圭	UY	598	是	否	否	是

國家/地區或區域	ISO 代碼	撥打代碼	支援短代碼	支援長代碼	支援寄件者 ID	支援雙向簡訊
烏茲別克	UZ	998	否	否	是	否
V						
萬那杜	VU	678	否	否	是	否
委內瑞拉	VE	58	否	否	否	否
越南	VN	84	否	否	需要註冊[1]	否
英屬維京群島	VG	1-284	否	否	是	否
美屬維京群島，US	VI	1-340	否	否	否	是
W						
X						
Y						
葉門	YE	967	否	否	是	否
Z						
尚比亞	ZM	260	否	否	是	否
辛巴威	ZW	263	否	否	是	否

備註

1. 寄件者必須使用已預先登錄的字母寄件者 ID。若要要求寄件者識別碼 AWS Support，請參閱使用 Amazon SNS 請求簡訊寄件者 ID。有些國家/地區要求寄件者符合特定要求或遵守某些限制，才能獲得核准。在這些情況下， AWS Support 可能會在您提交寄件者識別碼要求後與您聯絡以取得其他資訊。

2. 寄件者必須為計劃傳送的每種訊息類型使用預先註冊的範本。如果寄件者不符合此需求，他們的郵件將會遭到封鎖。若要註冊範本，請使用開啟 Amazon SNS 簡訊案例 AWS Support。建立案例時，請提供您請求寄件者 ID 時所提供的相同資訊。如需詳細資訊，請參閱 使用 Amazon SNS 請求簡訊寄件者 ID。有些國家/地區要求寄件者符合特定要求或遵守某些限制，才能獲得核准。在這些情況下， AWS Support 可能會要求您提供其他資訊。

   > **Note**
   >
   > 若要傳送訊息至中國，您必須先透過以下方式註冊您的範本以 AWS Support 供核准。

3. 寄件者必須使用已預先登錄的字母寄件者 ID。需額外註冊步驟。如需詳細資訊，請參閱 印度的寄件者 ID 註冊要求。

4. 這些國家的長代碼僅支援傳入簡訊。換句話說，您無法使用這些長代碼傳送訊息給收件人，但您可以使用長代碼接收從收件人傳送的訊息。如果您使用字母寄件者 ID 傳送訊息，這些長代碼可讓收件人選擇停止接收，因為寄件者 ID 僅支援傳出簡訊。

5. Amazon SNS 可以使用已在 Singapore SMS Sender ID Registry (SSIR) 註冊的寄件者 ID 傳送簡訊流量，該登錄檔由新加坡資訊通信媒體發展局 (IMDA) 建立。如需使用新加坡寄件者 ID 之需求的詳細資訊，請參閱 新加坡的寄件者 ID 註冊要求。

   您也可以使用未註冊的寄件者 ID 或其他原始身分類型 (例如短代碼或長代碼) 在新加坡傳送簡訊流量。

6. 即使沒有專用短碼，Amazon SNS 仍會嘗試使用共用短碼集區將訊息傳送給紐西蘭收件者。由於當地電信業者對於共用號碼的限制，我們只能盡最大努力透過這些共用號碼傳遞。因此，Amazon SNS 強烈建議針對傳送至紐西蘭的所有流量採購專用短碼。包含 URL 的訊息必須透過專用短碼程序加入允許清單。如需採購短碼的詳細資訊，請參閱 針對使用 Amazon SNS 的簡訊請求專用的短碼。

簡訊最佳實務

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。

主題

- 遵守法律、法規和電信業者要求
- 取得許可
- 不要傳送至舊清單
- 稽核您的客戶清單
- 保留記錄
- 訊息保持清晰、誠懇和簡潔
- 適當回應
- 根據參與度來調整傳送
- 適時傳送
- 避免跨管道疲勞轟炸
- 使用專用短碼
- 驗證您的目的地電話號碼
- 考量備援的設計
- 簡訊限制和規定
- 管理退出關鍵字
- CreatePool
- PutKeyword
- 管理號碼設定
- Amazon SNS 中的簡訊字元限制

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。

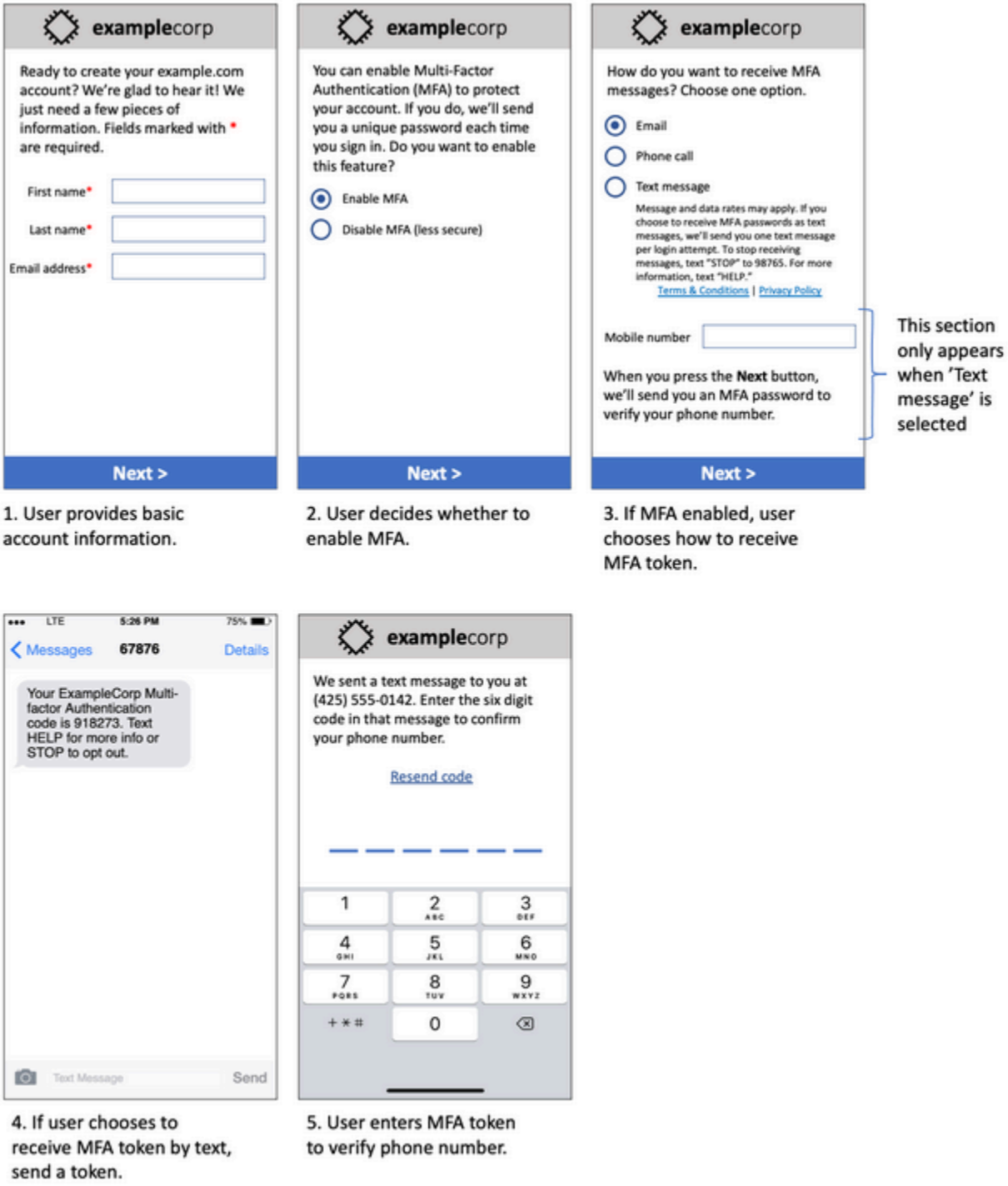
必要的披露

- 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。



多重要素驗證使用案例的圖樣

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圖樣，您可以提供選擇加入指令碼、行銷材料或收集數字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，AWS Support 會請您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 通道，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。若要進一步了解請求短碼，請參閱 針對使用 Amazon SNS 的簡訊請求專用的短碼。

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。如需簡訊定價的相關資訊，請參閱 Amazon SNS 定價。

當 Amazon SNS 接受傳送簡訊的請求時 (由於呼叫 SendMessagesA PI，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 Amazon SNS 電話號碼驗證服務來判斷電話號碼是否有效，以及電話號碼的類型 (例如行動電話、有線電話或 VoIP)。如需詳細資訊，請參閱 Amazon Pinpoint 開發人員指南中的驗證 Amazon Pinpoint 中的電話號碼。

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 可在多個 AWS 區域 中使用。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的收件者，則需要在您計劃要使用的每個 AWS 區域 中要求個別的短碼。

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。有多種電話號碼類型可用，無論是在相同 AWS 區域 或分散在多個 AWS 區域—提供額外的備援層，有助於提高恢復能力。

簡訊限制和規定

如需簡訊上限和限制，請參閱 Amazon Pinpoint 使用者指南中的 Amazon Pinpoint 中的簡訊上限和限制。

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。如需更多詳細資訊，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 Amazon Pinpoint SMS 和 Voice API 中的 CreatePool。

PutKeyword

使用 `PutKeyword` API 動作建立發送電話號碼或發送電話號碼的關鍵字設定。如需詳細資訊，請參閱 Amazon Pinpoint 簡訊和語音 API 中的 PutKeyword。

管理號碼設定

您可以使用 SMS and voice settings (簡訊和語音設定) 頁面的 Number settings (號碼設定) 區段中的選項，管理您向 AWS 支援部門請求並指派給您帳戶的專用短代碼和長代碼的設定。如需詳細資訊，請參閱 Amazon Pinpoint 使用者指南中的管理號碼設定。

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
否	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m
n	o	p	q	r	s	t	u	v	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

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/bill1.
```

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。

行動推送通知

透過 Amazon SNS，您可將推送通知訊息直接傳送至行動裝置上的應用程式。傳送至行動端點的推送通知訊息可能在行動應用程式中顯示為訊息提醒、識別證更新或者甚至聲音提醒。

主題

使用者通知運作方式

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 使用者通知程序概觀。

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。

使用者通知程序概觀

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 建立平台端點。
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 發佈到行動裝置 及發佈 API。

設定行動應用程式

本節說明如何搭配中所述 AWS Management Console 的資訊使用，Amazon SNS 使用者通知的先決條件以設定行動應用程式。

主題

- Amazon SNS 使用者通知的先決條件
- 建立平台應用程式
- 建立平台端點
- 新增裝置字符或註冊 ID
- Apple 身分驗證方法
- Firebase Cloud Messaging (FCM) 身份驗證方法

- 火力地堡雲消息傳遞（FCM）端點管理

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。裝置字符 (加上一些額外設定) 會以稱為平台端點的物件表示。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。一旦您取得這些先決條件資訊，即可使用 AWS Management Console 或 Amazon SNS 行動推送 API 傳送推送通知訊息。如需詳細資訊，請參閱 使用者通知程序概觀。

建立平台應用程式

為了讓 Amazon SNS 將通知訊息傳送到行動端點，無論是直接傳送或是透過主題訂閱傳送，您必須先建立一個平台應用程式。向 AWS 註冊應用程式後，下一步是建立應用程式和行動裝置的端點。接著 Amazon SNS 會使用端點，將通知訊息傳送給應用程式和裝置。

建立平台應用程式

1. 登入 Amazon SNS 主控台。
2. 於導覽窗格中，選擇 Mobile (行動裝置)，然後選擇 Push notifications (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)。

   如需可在建立行動應用程式的 AWS 區域清單，請參閱 行動應用程式的支援區域。
4. 對於 Application name (應用程式名稱)，請輸入代表應用程式的名稱。

   應用程式名稱只能包含大寫和小寫 ASCII 字母、數字、底線、連字號和句號。名稱的長度必須也是 1 至 256 個字元。
5. 對於 Push notification platform (推送通知平台)，選擇應用程式已註冊的平台，然後輸入適當的憑證。

   > **Note**
   >
   > 如果您使用其中一個 Apple 推送通知服務 (APNS) 平台，您可以選擇字符式或憑證式身分驗證，然後選擇 Choose file (選擇檔案)，將 .p8 或 .p12 檔案 (從 Keychain Access 匯出) 上傳至 Amazon SNS。
6. 選擇 Create platform application (建立平台應用程式)。

   此操作會向 Amazon SNS 註冊該應用程式，進而針對所選平台建立平台應用程式物件，然後傳回相對應的 PlatformApplicationArn。

建立平台端點

當應用程式和行動裝置註冊推送通知服務時，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。如需更多詳細資訊，請參閱 Amazon SNS 使用者通知的先決條件 及 使用者通知程序概觀。

本節說明建立平台端點的建議方法。

主題

- 建立平台端點
- 虛擬程式碼
- AWS SDK 範例

- 故障診斷

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。

此動CreatePlatformEndpoint作會執行下列動作：

- 如果平台端點已經存在，則請勿再次建立。將現有平台端點的 ARN 傳回呼叫者。
- 如果具有相同裝置字符但不同設定的平台端點已經存在，則請勿再次建立。將例外擲回呼叫者。
- 如果平台端點不存在，則請建立。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
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

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。也可在初始取得端點屬性動作呼叫因為找不到的例外狀況失敗時呼叫，在應用程式若知道其端點 ARN 但已被刪除時發生。
- 取得端點屬性動作會被呼叫，以確認平台端點的狀態，即使平台端點才剛建立。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 SDK 提供的 Amazon SNS 用戶端實作先前的虛擬程式碼。

要使用 AWS SDK，您必須使用憑據對其進行配置。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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
```

```
 */

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

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

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

特別是對於 FCM 端點，您可能認為最好存儲應用程序發布的第一個設備令牌，然後在每次啟動應用程序時都使用該設備令牌調用 create 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」。

有關管理 FCM 端點的更多信息，請參閱火力地堡雲消息傳遞（FCM）端點管理。

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。

新增裝置字符或註冊 ID

當您首次向通知服務 (例如 Apple Push Notification Service (APN) 及 Firebase Cloud Messaging (FCM)) 註冊應用程式和行動裝置時，系統會從通知服務傳回裝置字符或註冊 ID。將裝置字符或註冊 ID 新增到 Amazon SNS 時，上述兩者會與 `PlatformApplicationArn` API 搭配使用以為應用程式與裝置建立端點。Amazon SNS 建立端點時，系統會傳回一個 `EndpointArn`。Amazon SNS 即透過 `EndpointArn` 得知要傳送通知訊息到哪一個應用程式和行動裝置。

您可以使用以下方法，將裝置字符或註冊 ID 新增到 Amazon SNS：

- 使用 AWS Management Console 將單一字符手動新增到 AWS
- 使用 `CreatePlatformEndpoint` API 上傳多個字符
- 從將來要安裝您的應用程式的裝置上註冊字符

手動新增裝置字符或註冊 ID

1. 登入 Amazon SNS 主控台。
2. 選擇行動裝置，然後選擇推送通知。
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。如果尚未建立平台應用程式，請立即建立。如需如何執行此動作的詳細資訊，請參閱建立平台應用程式。
4. 選擇新增端點。
5. 在 Endpoint Token (端點字符) 方塊中，根據通知服務的種類輸入字符 ID 或註冊 ID。例如，使用 ADM 和 FCM 時，您需輸入註冊 ID。
6. (選用) 在 User Data (使用者資料) 方塊中，輸入任意資訊來與該端點相關聯。Amazon SNS 不會使用此資料。資料必須為 UTF-8 格式並小於 2KB。
7. 最後，選擇 Add Endpoints (新增端點)。

   現在具備已建立端點，您即可直接傳送訊息到行動裝置，或傳送訊息到已訂閱主題的行動裝置。

使用 **`CreatePlatformEndpoint`** API 上傳多個字符

下列步驟顯示如何使用 AWS 提供的範本 Java 應用程式 (`bulkupload` 套件)，將多個字符 (裝置字符或註冊 ID) 上傳到 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。此步驟假設您已經安裝 AWS SDK for Java 且您已持有自己 AWS 帳戶 帳戶的 AWS 安全憑證。如需更多詳細資訊，請參閱 AWS SDK for Java。如需憑證的詳細資訊，請參閱 AWS 一般參考 中的如何取得安全憑證？。

1. 下載並解壓縮 snsmobilepush.zip 檔案。
2. 在 Eclipse 中建立新 Java 專案。
3. 將 SNSSamples 資料夾匯入至新建立 Java 專案的最上層目錄。在 Eclipse 中，以滑鼠右鍵選擇 Java 專案的名稱，然後選擇 Import (匯入)，展開 General (一般)，選擇 File System (檔案系統)，選擇 Next (下一步)，導覽到 SNSSamples 資料夾，選擇 OK (確定)，再選擇 Finish (完成)。
4. 下載 OpenCSV 程式庫 的副本，並將其新增到 bulkupload 套件的建置路徑上。
5. 開啟 BulkUpload.properties 套件中的 bulkupload 檔案。
6. 將以下項目新增到 BulkUpload.properties：

   - 要向其新增終端節點的 ApplicationArn。
   - 包含字符之 CSV 檔案位置的絕對路徑。
   - 針對記錄 Amazon SNS 正確剖析或剖析失敗的字符所建立之 CSV 檔案的名稱 (如 goodTokens.csv 和 badTokens.csv)。
   - (選用) 用於指定包含字符的 CSV 檔案中之分隔符號和引號的字元。
   - (選用) 用於同時建立端點的執行緒數量。預設為 1 個執行緒。

   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
   ```

7. 執行 BatchCreatePlatformEndpointSample.java 應用程式以將字符上傳到 Amazon SNS。

在此範例中，針對成功上傳到 Amazon SNS 之字符所建立的端點，將會記錄到 goodTokens.csv 中，而格式不正確的字符會記錄到 badTokens.csv 中。此外，您應會看見 STD OUT 日誌寫入 Eclipse 的主控台，包含的內容與下列相似：

```
<1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-
west-2:111122223333:app/FCM/fcmpushapp/165j2214-051z-3176-b586-138o3d420071
<2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/mytokens.csv
```

從將來要安裝您的應用程式的裝置上註冊字符

您可以從下列兩個選項中使用其中一項來執行：

- 使用 Amazon Cognito 服務：您的行動應用程式將需要登入資料，以建立與您的 Amazon SNS 平台應用程式相關聯的端點。我們建議您使用經過一段期間後便會過期的暫時性憑證。在大部分情況下，建議您使用 Amazon Cognito 建立暫時性安全登入資料。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南。如果您想要在應用程式向 Amazon SNS 註冊時收到通知，您可以進行註冊，以接收可提供新端點 ARN 的 Amazon SNS 事件。您也可以使用 ListEndpointByPlatformApplication API 來取得已向 Amazon SNS 註冊的端點完整清單。
- 使用代理伺服器：如果您的應用程式基礎設施已經為行動應用程式設定完畢，可供其呼叫並在每次安裝中進行註冊，則可以繼續使用此設定。您的伺服器會以代理伺服器方式運作，並將裝置字符連同任何要存放的使用者資料一起傳遞到 Amazon SNS 行動推送通知。為達成此目的，代理伺服器會使用您的 AWS 憑證連線至 Amazon SNS，並使用 CreatePlatformEndpoint API 呼叫以上傳字符資訊。系統將傳回新建立的端點 Amazon 資源名稱 (ARN)，伺服器可以儲存該端點以便對 Amazon SNS 進行後續的發佈呼叫。

Apple 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。

要使用 AWS 管理主控台管理 APNS 設定

1. 登入 Amazon SNS 主控台。
2. 在 Mobile (行動裝置) 下，選擇 Push notifications (推送通知)。
3. 選取您要對其編輯 APN 設定的應用程式，然後選取 Edit (編輯)。
4. 在 Edit (編輯) 頁面，對 Authentication type (身分驗證類型)，選擇 Token (字符) 或者 Certificate (憑證)。
5. 為憑證或字符簽署金鑰載入適當的憑證。您可以從您的 Apple 開發人員帳戶取得資訊。
6. 根據您選擇的身分驗證類型，執行下列其中一個操作：
   - 如果您選擇 Token (字符)，請提供下列 Apple 開發人員帳戶中的資訊。Amazon SNS 需要此資訊才能建構身分驗證字符。
     - Signing key (簽署金鑰) - 來自 Apple 開發人員帳戶，您下載為 .p8 檔案的身分驗證字符簽署金鑰。Apple 僅可讓您下載一次簽署金鑰。
     - Signing Key ID (簽署金鑰 ID) – 指派給簽署金鑰的 ID。Amazon SNS 需要此資訊才能建構身分驗證字符。若要在 Apple 開發人員帳戶中尋找這個值，請選擇 Certificates, IDs & Profiles (憑證、ID 和設定檔)，然後在 Keys (金鑰) 區段選擇您的金鑰。
     - Team identifier (團隊識別符) – 指派給 Apple 開發人員帳戶團隊的 ID。您可以在 Membership (會員資格) 頁面上找到這個值。
     - Bundle identifier (搭售套件識別符) – 指派給應用程式的 ID。若要尋找此值，請選擇 Certificates, IDs & Profiles (憑證、ID 和設定檔)，然後在 Identifiers (識別符) 區段中選擇 App ID (應用程式 ID)，接著選擇您的應用程式。
   - 如果您選擇 Certificate (憑證)，請提供下列資訊：

- SSL certificate (SSL 憑證) – 您的 TLS 憑證的 .p12 檔案。從您的 Apple 開發人員帳戶下載並安裝憑證後，可以從 Keychain Access 匯出此檔案。
- Certificate password (憑證密碼) – 如果您已為憑證指派密碼，請在這裡指出。

7. 完成後，請選擇 Save changes (儲存變更)。

Firebase Cloud Messaging (FCM) 身份驗證方法

本主題介紹瞭如何從谷歌獲取所需的 FCM API（HTTP v1）憑據以與 AWS API 一起使用，以 AWS CLI 及. AWS Management Console

主題

- 先決條件
- 管理 FCM 設定 (API)
- 管理 FCM 設定 (CLI)
- 管理 FCM 設定 (主控台)

> **Important**
>
> 2023 年 6 月 20 日-谷歌棄用了他們的火力地堡雲消息傳遞（FCM）傳統 HTTP API。Amazon SNS 現在支持使用 FCM HTTP v1 API 交付到所有設備類型。我們建議您在 2024 年 6 月 1 日或之前將現有的移動推送應用程序遷移到最新的 FCM HTTP v1 API，以避免中斷。
> 2024 年 1 月 18 日 — Amazon SNS 推出了對 FCM HTTP v1 API 的支持，用於將移動推送通知交付到安卓設備。
> 2024 年 3 月 26 日 — Amazon SNS 支持用於蘋果設備和網絡推送目的地的 FCM HTTP v1 API。我們建議您在 2024 年 6 月 1 日或之前將現有的移動推送應用程序遷移到最新的 FCM HTTP v1 API，以避免應用程序中斷。

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 應用程序控制台獲取以下信息：

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，會使用一個短暫的存取權杖。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。

管理 FCM 設定 (API)

您可以使用 AWS API 創建 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 簡單通知服務端點和配額。

與 Amazon SNS 主題（AWS API）一起創建 FCM 推送通知

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`

檢索現有 Amazon SNS 主題（AWS API）的 FCM 憑據類型

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

管理 FCM 設定 (CLI)

您可以使用 AWS Command Line Interface （ CLI ）創建 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，檔案必須是字串格式，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)

使用以下步驟輸入應用程式用於連接到 FCM 的憑證。

1. 登入 Amazon SNS 主控台。
2. 在 Mobile (行動裝置) 下，選擇 Push notifications (推送通知)。
3. 選擇現有的 FCM 應用程式，然後選擇編輯。如果尚未建立平台應用程式，請見 建立平台應用程式。
4. 在編輯頁面上，針對 Firebase Cloud Messaging 登入資料，請選擇權杖或金鑰。您可以從 Firebase 應用程式主控台獲取下列資訊。
   - 如果您選擇權杖，請上傳有效的私有金鑰檔案。此檔案的內容用於在發送通知時生成短期存取權杖。

- 如果您選擇金鑰，請輸入 Google API 金鑰。

5. 完成後，請選擇儲存變更。

相關主題

- 在 Amazon SNS 中使用谷歌火力地堡雲消息傳遞（FCM）v1 有效載荷

火力地堡雲消息傳遞（FCM）端點管理

主題

- 管理和維護設備令牌
- 檢測無效令牌
- 刪除過時的令牌

管理和維護設備令牌

您可以按照以下步驟確保移動應用程序的推送通知的交付性：

1. 將所有裝置權杖、對應的 Amazon SNS 端點 ARN 和時間戳記儲存在應用程式伺服器上。
2. 移除所有過時的權杖，並刪除對應的 Amazon SNS 端點 ARN。

應用程序初次啟動後，您將收到該設備的設備令牌（也稱為註冊令牌）。該設備令牌由設備的操作系統鑄造，並與您的 FCM 應用程序綁定。收到此裝置權杖後，您可以在 Amazon SNS 註冊為平台端點。建議您將裝置權杖、Amazon SNS 平台端點 ARN 和時間戳記儲存到應用程式伺服器或其他永久性存放區，以儲存這些權杖。要設置 FCM 應用程序以檢索和存儲設備令牌，請參閱 Google 的 Firebase 文檔中的檢索和存儲註冊令牌。

維護 up-to-date 令牌很重要。在下列情況下，使用者的裝置權杖可能會變更：

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. 使用者清除應用程式資料。

當您的裝置權杖變更時，我們建議您使用新的權杖更新對應的 Amazon SNS 端點。這可讓 Amazon SNS 繼續與註冊的裝置進行通訊。您可以通過在移動應用程序中實現以下偽代碼來做到這一點。其中

說明建立和維護已啟用平台端點的建議作法。這種方法可以在每次移動應用程序啟動時執行，也可以在後台作為計劃作業執行。

虛擬程式碼

使用以下 FCM 偽代碼來管理和維護設備令牌。

```
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
```

要了解有關令牌更新要求的更多信息，請參閱 Google 的 Firebase 文檔中的定期更新令牌。

檢測無效令牌

將消息分派到具有無效設備令牌的 FCM v1 端點時，Amazon SNS 將收到以下異常之一：

- UNREGISTERED(HTTP 404) — 當 Amazon SNS 收到此例外狀況時，您將收到具有中`FailureType`的交付失敗事件`InvalidPlatformToken`，且與端點關聯的平台權杖無效。`FailureMessage`當交付失敗並出現此例外狀況時，Amazon SNS 將停用您的平台端點。
- `INVALID_ARGUMENT`(HTTP 400) — 當 Amazon SNS 收到此例外狀況時，表示裝置權杖或訊息承載無效。有關更多信息，請參閱谷歌ErrorCode的火力地堡文檔。

由於在上述任何一種情況下都`INVALID_ARGUMENT`可以傳回，因此 Amazon SNS 將傳回`FailureType`的一個 `InvalidNotificationFailureMessage`，且通知內文無效。當您收到此

錯誤時，請確認您的承載是否正確。如果正確，請驗證設備令牌是否正確 up-to-date。當交付失敗並出現此例外狀況時，Amazon SNS 不會停用您的平台端點。

您將遇到傳`InvalidPlatformToken`遞失敗事件的另一種情況是，當註冊的設備令牌不屬於嘗試發送該消息的應用程序時。在這種情況下，谷歌將返回一個發件人 _ID_ 不匹配錯誤。當交付失敗並出現此例外狀況時，Amazon SNS 將停用您的平台端點。

當您為應用程序設置交付狀態日誌記錄 CloudWatch時，可以使用從 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傳送事件，請參閱可用的應用程式事件。

刪除過時的令牌

一旦向端點設備傳遞消息開始失敗，令牌被視為過時。Amazon SNS 會將這些過時的權杖設定為平台應用程式的停用端點。當您發佈到已停用的端點時，Amazon SNS 將傳回具有`FailureType`的`EventDeliveryFailure`事件`EndpointDisabled`，而端點*`FailureMessage`*的一個已停用。若要接收應用程式的傳送事件，請參閱可用的應用程式事件。

當您從 Amazon SNS 收到此錯誤時，您需要在平台應用程式中移除或更新過時的權杖。

傳送行動裝置推送通知

本節說明使用如何傳送推送通知訊息到您的行動裝置。

主題

- 發佈到主題
- 發佈到行動裝置
- 使用平台特定承載進行發佈

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。

發佈到行動裝置

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 [行動裝置推播通知] 頁面的 [平台應用程式] 區段中，選擇應用程式的名稱，例如***MyApp***。
4. 在***MyApp***頁面的「端點」區段中，選擇端點，然後選擇「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。

使用平台特定承載進行發佈

您可以使用 AWS Management Console 或 Amazon SNS API，將具有平台特定承載的自訂訊息傳送到行動裝置。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。

主題

- 傳送 JSON 格式訊息
- 傳送平台特定訊息
- 在多個平台上向應用程式傳送訊息
- 將訊息做為提醒或背景通知傳送到 APN
- 在 Amazon SNS 中使用谷歌火力地堡雲消息傳遞（FCM）v1 有效載荷

傳送 JSON 格式訊息

當您傳送平台特定酬載時，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 現在支持火力地堡雲消息傳遞（FCM）HTTP v1 API，用於將移動推送通知發送到安卓設備。
> 2024 年 3 月 26 日 — Amazon SNS 支持用於蘋果設備和網絡推送目的地的 FCM HTTP v1 API。我們建議您在 2024 年 6 月 1 日或之前將現有的移動推送應用程序遷移到最新的 FCM HTTP v1 API，以避免應用程序中斷。

- 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 開發套件或指定`AWS.SNS.MOBILE.APNS.PUSH_TYPE`保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
```

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 中使用谷歌火力地堡雲消息傳遞（FCM）v1 有效載荷

Amazon SNS 支持使用 FCM HTTP v1 API 將通知發送到安卓，iOS 和網絡推送目的地。本主題提供使用 CLI 或 Amazon SNS API 發佈行動推送通知時的承載結構範例。

發送 FCM 通知時，您可以在有效負載中包含以下消息類型：

- 數據消息-數據消息由客戶端應用程序處理，並包含自定義鍵值對。建構資料訊息時，您必須包含 JSON 物件做為值的索引`data`鍵，然後輸入您的自訂索引鍵值配對。
- 通知消息或顯示消息 — 通知消息包含 FCM SDK 處理的預定義鍵集。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：
  - 安卓通知鍵
  - APNS 通知金鑰
  - Web 推送通知鍵

有關 FCM 消息類型的更多信息，請參閱在谷歌的 Firebase 文檔中的消息類型。

內容

使用 FCM v1 有效負載結構發送消息

如果您是第一次創建 FCM 應用程序，或者希望利用 FCM v1 功能，則可以選擇發送 FCM v1 格式的有效載荷。若要這麼做，您必須包含頂層金鑰`fcmV1Message`。有關構建 FCM v1 有效載荷的更多信息，請參閱從舊版 FCM API 遷移到 HTTP v1 和在 Google 的 Firebase 文檔中跨平台自定義消息。

FCM v1 發送到 Amazon SNS 的示例有效載荷：

> **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\": {
                    \"badge\": \"5\",
                    \"title\": \"string\",
                    \"body\": \"string\"
                },
                \"data\": {
                    \"dataWeb\": \"priority message\",
                }
            }
        }
    }
  }"
}
```

傳送 JSON 承載時，請務必在要求中包含該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
```

有關發送 FCM v1 格式化的有效載荷的更多信息，請參閱 Google 的 Firebase 文檔中的以下內容：

- 從傳統的 FCM API 遷移到 HTTP V1
- 關於 FCM 消息

- 其餘資源：projects.messages

使用舊有效負載結構將消息發送到 FCM v1 API

遷移到 FCM v1 時，您無需更改用於遺留憑據的有效負載結構。Amazon SNS 將您的有效載荷轉換為新的 FCM v1 有效負載結構，並發送給谷歌。

輸入消息有效負載格式：

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

傳送給谷歌的訊息：

```
{
  "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

- 舊版到 v1 的映射不支持蘋果推送通知服務（APNS）`headers`或`fcm_options`密鑰。如果您想使用這些字段，請發送 FCM v1 有效載荷。
- 在某些情況下，FCM v1 需要消息標題才能向您的 APNs 設備發送靜默通知。如果您目前正在向 APNs 設備發送靜默通知，則它們將無法與舊版方法一起使用。相反，我們建議使用 FCM v1 有效載荷以避免意外問題。若要尋找 APNs 標頭清單及其用途，請參閱 Apple 開發人員指南中的與 APN 通訊。
- 如果您在傳送通知時使用 TTL Amazon SNS 屬性，則只會在`android`欄位中更新該屬性。如果要設置 TTL APNS 屬性，請使用 FCM v1 有效負載。
- `androidapns`、和`webpush`金鑰將會對應並填入所有提供的相關金鑰。例如，如果您提供`title`（這是所有三個平台之間共享的密鑰），則 FCM v1 映射將使用您提供的標題填充所有三個平台。
- 平台之間的某些共享密鑰需要不同的值類型。例如，傳遞給的`badge`鍵`apns`需要一個整數值，而傳遞給的`badge`鍵`webpush`需要一個字符串值。在提供`badge`密鑰的情況下，FCM v1 映射將僅填充您為其提供有效值的密鑰。

FCM 交付失敗事件

下表提供了與 Google 針對 FCM v1 通知請求收到的錯誤/狀態碼相對應的 Amazon SNS 故障類型。當您為應用程序設置交付狀態日誌記錄 CloudWatch 時，可以使用從 FCM v1 API 收到的所有觀察到的錯誤代碼。

FCM 錯誤/狀態碼	Amazon SNS 故障類型	失敗訊息	原因和緩解
`UNREGISTERED`	`InvalidPlatformToken`	與端點關聯的平台令牌無效。	連接到端點的設備令牌過時或無效。Amazon SNS 已停用您的端點。將 Amazon SNS 端點更新為最新的裝置權杖。
`INVALID_ARGUMENT`	`InvalidNotification`	通知內文無效。	設備令牌或消息有效負載可能無效。確認

FCM 錯誤/狀態碼	Amazon SNS 故障類型	失敗訊息	原因和緩解
			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權杖。
SENDER_ID_MISMATCH	InvalidPlatformToken	與端點關聯的平台令牌無效。	與設備令牌關聯的平台應用程序沒有發送到設備令牌的權限。驗證您在 Amazon SNS 平台應用程序中使用了正確的 FCM 憑據。
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了。您可以將這些訊息儲存在無效字母佇列 (DLQ) 中，稍後再重新驅動它們。
INTERNAL	UnexpectedFailure	意外故障; 請聯繫 Amazon。失敗短語 [內部錯誤]。	FCM 服務器在嘗試處理請求時遇到錯誤。Amazon SNS 執行的所有重試都失敗了。您可以將這些訊息儲存在無效字母佇列 (DLQ) 中，稍後再重新驅動它們。

FCM 錯誤/狀態碼	Amazon SNS 故障類型	失敗訊息	原因和緩解
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	平台應用程式認證無效。	無法傳送針對 iOS 裝置或 Webpush 裝置的訊息。驗證您的開發和生產認證是否有效。
QUOTA_EXCEEDED	Throttled	由 [gcm] 限制的請求。	已超過郵件速率配額、裝置訊息速率配額或主題郵件速率配額。有關如何解決此問題的信息，請參閱ErrorCode在谷歌的 Firebase 文檔中。
PERMISSION_DENIED	InvalidNotification	通知內文無效。	在發生PERMISSION_DENIED 異常的情況下，調用者（您的 FCM 應用程序）沒有在有效負載中執行指定操作的權限。導航到 FCM 控制台，並驗證您的憑據已啟用所需的 API 操作。

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 (SDK) 或查詢 API。

主題

使用 AWS Management Console 設定訊息傳遞狀態屬性

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": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\":[{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```

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": "examp1e29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken,
 id=1, cause=null)",
```

```
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/
APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

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 AWS SDK for Java 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。

推送通知服務	回應代碼
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」。

行動應用程式事件

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 主控台或 AWS 開發套件來設定應用程式事件通知。 AWS CLI

主題

- 可用的應用程式事件
- 傳送行動裝置推送通知

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。

屬性名稱	通知觸發
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 開發套件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 [行動裝置推播通知] 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 開發套件

使用 AWS 開發套件使用 Amazon SNS API 提交`SetPlatformApplicationAttributes`請求，以設定應用程式事件通知。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，包括入門說明和舊版的相關資訊，請參閱搭配 AWS 開發套件使用 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。如需更多詳細資訊，請參閱 使用者通知程序概觀。

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。

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`

將 Amazon SNS 存留時間 (TTL) 訊息屬性用於行動推送通知

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。這是在 Amazon SNS 消息正文中為支持此功能的移動推送通知服務（例如 Amazon 設備消息傳遞（ADM）和火力地堡雲消息傳遞（FCM）在發送到 Android 時設置 TTL 的功能之外。

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。若要在訊息屬性內指定 TTL，您可以使用 AWS Management Console、 AWS 軟體開發套件 (SDK) 或查詢 API。

主題

- 推送通知服務的 TTL 訊息屬性
- 決定 TTL 的優先順序
- 使用指定 TTL AWS Management Console

推送通知服務的 TTL 訊息屬性

以下是推播通知服務的 TTL 訊息屬性清單，您可以在使用 AWS SDK 或查詢 API 時用來設定這些屬性：

推送通知服務	TTL 訊息屬性
Amazon Device Messaging (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple Push Notification Service (APN)	`AWS.SNS.MOBILE.APNS.TTL`
Apple Push Notification Service Sandbox (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
百度雲端推送 (百度)	`AWS.SNS.MOBILE.BAIDU.TTL`

推送通知服務	TTL 訊息屬性
火力地堡雲消息傳遞（發送到安卓系統時的 FCM）	`AWS.SNS.MOBILE.FCM.TTL`
Windows 推送通知服務 (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

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。使用指定 TTL (以秒為單位) 時，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。 AWS Management Console

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。

如果您為推播通知訊息指定 TTL，則 TTL 值必須是正整數，除非的值對推播通知服務0具有特定意義，例如 APNs 和 FCM (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***頁面的「端點」區段中，選擇應用程式端點，然後選擇「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)。

行動應用程式的支援區域

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)
- 亞太區域 (香港)
- 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)
- Europe (Milan)
- 歐洲 (巴黎)
- 歐洲 (斯德哥爾摩)
- 中東 (巴林)

- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (US-West)

行動推播通知最佳實務

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

如果由於使用者在裝置上的動作而導致裝置字符發生變更 (例如，應用程式在裝置上重新安裝) 或憑證更新影響在特定 iOS 版本上執行的裝置，可能會發生交付問題。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 Token 本身無效，或端點有效但已停用 (例如，生產環境和沙箱環境不相符)，這可能會為相同裝置引入重複項目。

可以使用裝置字符管理機制，例如虛擬程式碼。

有關管理和維護 FCM v1 設備令牌的信息，請參閱火力地堡雲消息傳遞（FCM）端點管理。

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。

電子郵件通知

本頁說明如何使用 AWS Management Console、 AWS SDK for Java或訂閱 Amazon SNS 主題的電子郵件地址 AWS SDK for .NET。

備註

- 您無法自訂電子郵件內文。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。
- 電子郵件傳遞輸送量是根據 Amazon SNS 配額而定。

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 Amazon SNS 主題的電子郵件地址 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 (DLQ)。
   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 開發套件訂閱 Amazon SNS 主題的電子郵件地址

要使用 AWS SDK，您必須使用憑據對其進行配置。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。

下列程式碼範例會示範如何使用Subscribe。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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《AWS SDK for .NET API 參考》中的訂閱。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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();
}
```

訂閱某個主題的行動應用程式。

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

訂閱某個主題的 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 SQS 佇列以取得主題。

```
        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;
            }
```

使用篩選器訂閱主題。

```
        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 ostringstream;
                ostringstream << "Filter messages for \"" << queueName
```

```
                            << "\"'s subscription to the topic \""
                            << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

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

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
```

```
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for C++ 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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用篩選條件訂閱主題的佇列。

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考中的訂閱。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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);
        }
    }
}
```

訂閱某個主題的 HTTP 端點。

```
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);
        }
    }
}
```

訂閱某個主題的 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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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'
  // }
};
```

訂閱某個主題的行動應用程式。

```
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;
};
```

訂閱某個主題的 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 SQS 佇列以取得主題。

```
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;
};
```

使用篩選器訂閱主題。

```
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-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的訂閱。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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()
    }
}
```

訂閱某個主題的 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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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());
}
```

訂閱某個主題的 HTTP 端點。

```
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 AWS SDK for PHP API 參考中的訂閱。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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
```

- 如需詳細資訊，請參閱《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的訂閱。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

訂閱主題的電子郵件地址。

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- 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。

使用 AWS 開發套件的 Amazon SNS 的程式碼範例

下列程式碼範例說明如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon SNS。

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和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

Cross-service examples (跨服務範例) 是跨多個 AWS 服務執行的應用程式範例。

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 開發套件使用 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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for .NET API 參考ListTopics中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

C MakeLists .txt 的 CMake 文件的代碼。

```
# 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 AWS SDK for C++ API 參考ListTopics中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
```

```
    fmt.Printf("\t%v\n", *topic.TopicArn)
   }
  }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListTopics中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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ListTopics中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for JavaScript API 參考ListTopics中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

- 有關 API 的詳細信息，請參閱 AWS SDK ListTopics中的 Kotlin API 參考。

程式碼範例

- 使用 AWS 開發套件針對 Amazon SNS 的動作
  - 搭CheckIfPhoneNumberIsOptedOut配 AWS 開發套件或 CLI 使用
  - 搭ConfirmSubscription配 AWS 開發套件或 CLI 使用
  - 搭CreateTopic配 AWS 開發套件或 CLI 使用
  - 搭DeleteTopic配 AWS 開發套件或 CLI 使用
  - 搭GetSMSAttributes配 AWS 開發套件或 CLI 使用
  - 搭GetTopicAttributes配 AWS 開發套件或 CLI 使用
  - 搭ListPhoneNumbersOptedOut配 AWS 開發套件或 CLI 使用
  - 搭ListSubscriptions配 AWS 開發套件或 CLI 使用
  - 搭ListTopics配 AWS 開發套件或 CLI 使用
  - 搭Publish配 AWS 開發套件或 CLI 使用
  - 搭SetSMSAttributes配 AWS 開發套件或 CLI 使用
  - 搭SetSubscriptionAttributes配 AWS 開發套件或 CLI 使用
  - 搭SetSubscriptionAttributesRedrivePolicy配 AWS 開發套件或 CLI 使用
  - 搭SetTopicAttributes配 AWS 開發套件或 CLI 使用
  - 搭Subscribe配 AWS 開發套件或 CLI 使用
  - 搭TagResource配 AWS 開發套件或 CLI 使用
  - 搭Unsubscribe配 AWS 開發套件或 CLI 使用
- 使用 AWS 開發套件的 Amazon SNS 案例
  - 使用 AWS 開發套件為 Amazon SNS 推送通知建立平台端點
  - 使用開發套件 AWS 建立並發佈至 FIFO Amazon SNS 主題

  - 使用開發 AWS 套件將簡訊發佈到 Amazon SNS 主題
  - 使用開發 AWS 套件使用 Amazon S3 將大型訊息發佈到 Amazon SNS
  - 使用開發 AWS 套件發佈 Amazon SNS 簡訊
  - 使用開發套件將 Amazon SNS 訊息發佈到 Amazon SQS 佇列 AWS
- 使 AWS 用開發套件的 Amazon SNS 的無伺服器範例
  - 使用 Amazon SNS 觸發條件調用 Lambda 函數
- 使 AWS 用開發套件的 Amazon SNS 跨服務範例
  - 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
  - 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
  - 使用開發套件使用 Amazon Rekognition 偵測影片中的人物和物件 AWS
  - 使用開發套件將 Amazon SNS 訊息發佈到 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

使用 AWS 開發套件針對 Amazon SNS 的動作

下列程式碼範例示範如何使用 AWS 開發套件執行個別 Amazon SNS 動作。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。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執行程式碼的指示。

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 (Amazon SNS) API 參考資料。

範例

- 搭CheckIfPhoneNumberIsOptedOut配 AWS 開發套件或 CLI 使用
- 搭ConfirmSubscription配 AWS 開發套件或 CLI 使用
- 搭CreateTopic配 AWS 開發套件或 CLI 使用
- 搭DeleteTopic配 AWS 開發套件或 CLI 使用
- 搭GetSMSAttributes配 AWS 開發套件或 CLI 使用
- 搭GetTopicAttributes配 AWS 開發套件或 CLI 使用
- 搭ListPhoneNumbersOptedOut配 AWS 開發套件或 CLI 使用

- 搭ListSubscriptions配 AWS 開發套件或 CLI 使用
- 搭ListTopics配 AWS 開發套件或 CLI 使用
- 搭Publish配 AWS 開發套件或 CLI 使用
- 搭SetSMSAttributes配 AWS 開發套件或 CLI 使用
- 搭SetSubscriptionAttributes配 AWS 開發套件或 CLI 使用
- 搭SetSubscriptionAttributesRedrivePolicy配 AWS 開發套件或 CLI 使用
- 搭SetTopicAttributes配 AWS 開發套件或 CLI 使用
- 搭Subscribe配 AWS 開發套件或 CLI 使用
- 搭TagResource配 AWS 開發套件或 CLI 使用
- 搭Unsubscribe配 AWS 開發套件或 CLI 使用

搭`CheckIfPhoneNumberIsOptedOut`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CheckIfPhoneNumberIsOptedOut`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for .NET API 參考CheckIfPhoneNumberIsOptedOut中的。

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已選擇不接收來自目前 AWS 帳戶的 SMS 訊息。

```
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CheckIfPhoneNumberIsOptedOut中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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CheckIfPhoneNumberIsOptedOut中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CheckIfPhoneNumberIsOptedOut中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CheckIfPhoneNumberIsOptedOut中的。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ConfirmSubscription配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

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ConfirmSubscription中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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ConfirmSubscription中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ConfirmSubscription中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考ConfirmSubscription中的。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CreateTopic**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考CreateTopic中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考CreateTopic中的。

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，請參閱命 AWS 令列介面使用者指南中的 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CreateTopic中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
```

```
func (actor SnsActions) CreateTopic(topicName string, isFifoTopic bool,
 contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(context.TODO(), &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CreateTopic中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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CreateTopic中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CreateTopic中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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()
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateTopic中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreateTopic中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件CreateTopic中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreateTopic中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱 AWS SDK CreateTopic中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   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 AWS SDK CreateTopic中的 SAP ABAP API 參考資料。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DeleteTopic`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考DeleteTopic中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考DeleteTopic中的。

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DeleteTopic中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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



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

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteTopic中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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DeleteTopic中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteTopic中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteTopic中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考DeleteTopic中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件DeleteTopic中的 Python (博托 3) API 參考。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK DeleteTopic中的 SAP ABAP API 參考資料。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetSMSAttributes配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetSMSAttributes。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ 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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

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

                Usage:    <topicArn>

                Where:
```

```
                   topicArn - The ARN of the topic from which to retrieve
attributes.
                """";

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

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

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
entry.getValue());
            }

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

        System.out.println("\n\nStatus was good");
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的 GetSMSAttributes。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的 GetSMSAttributes。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP 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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetTopicAttributes配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetTopicAttributes。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK for .NET API 參考GetTopicAttributes中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考GetTopicAttributes中的。

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GetTopicAttributes中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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GetTopicAttributes中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetTopicAttributes中的。

適用於 JavaScript (v2) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetTopicAttributes中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetTopicAttributes中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
$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 AWS SDK for PHP API 參考GetTopicAttributes中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK GetTopicAttributes中的 SAP ABAP API 參考資料。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListPhoneNumbersOptedOut`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`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ListPhoneNumbersOptedOut中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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ListPhoneNumbersOptedOut中的。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListPhoneNumbersOptedOut中的。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListSubscriptions**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListSubscriptions。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK for .NET API 參考ListSubscriptions中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考ListSubscriptions中的。

CLI

AWS CLI

列出您的 SNS 訂閱

下列list-subscriptions範例會顯示您 AWS 帳戶中的 SNS 訂閱清單。

```
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ListSubscriptions中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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ListSubscriptions中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListSubscriptions中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

- 有關 API 的詳細信息，請參閱 AWS SDK ListSubscriptions中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考ListSubscriptions中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件ListSubscriptions中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListSubscriptions中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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 AWS SDK ListSubscriptions中的 SAP ABAP API 參考資料。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListTopics`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListTopics`。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for .NET API 參考ListTopics中的。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ API 參考ListTopics中的。

CLI

AWS CLI

列出您的 SNS 主題

下列`list-topics`範例會列出您 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ListTopics中的。

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(context.TODO())
```

```
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListTopics中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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ListTopics中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListTopics中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

- 有關 API 的詳細信息，請參閱 AWS SDK ListTopics中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考ListTopics中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件ListTopics中的 Python (博托 3) API 參考。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- 如需詳細資訊，請參閱《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListTopics中的。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱 AWS SDK ListTopics中的 Rust API 參考資料。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   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 AWS SDK ListTopics中的 SAP ABAP API 參考資料。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭Publish配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

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

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

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

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

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

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

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

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

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 AWS SDK for .NET API 參考中的發佈。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ 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!" \
    --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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考中的發佈。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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-xxxxxxxxxxxx'
  // }
  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, deduplicationId, choices;

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

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

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

```
    }

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

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

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

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的發佈。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的發佈。

PowerShell

適用的工具 PowerShell

範例 1：此範例顯示以單一 MessageAttribute 宣告的內嵌方式發佈郵件。

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

示例 2：此示例顯示發布具有多個事先 MessageAttributes 聲明的消息。

```
$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 指令程式參考中發佈。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的發佈。

Rust

適用於 Rust 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

```
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- 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。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭SetSMSAttributes配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用SetSMSAttributes。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for C++ 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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的 SetSMSAttributes。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP 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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭配 AWS 開發套件或 CLI 使用 `SetSubscriptionAttributes`

下列程式碼範例會示範如何使用`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SetSubscriptionAttributes中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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SetSubscriptionAttributes中的。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 的詳細資訊，請參閱AWS 開發套件SetSubscriptionAttributes中的 Python (博托 3) API 參考。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭SetSubscriptionAttributesRedrivePolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用SetSubscriptionAttributesRedrivePolicy。

Java

適用於 Java 1.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
// 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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭SetTopicAttributes配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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SetTopicAttributes中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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SetTopicAttributes中的。

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考SetTopicAttributes中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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.")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK SetTopicAttributes中的 Kotlin API 參考。

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for PHP API 參考SetTopicAttributes中的。

Ruby

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考SetTopicAttributes中的。

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK SetTopicAttributes中的 SAP ABAP API 參考資料。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭Subscribe配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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《AWS SDK for .NET API 參考》中的訂閱。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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();
}
```

訂閱某個主題的行動應用程式。

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

訂閱某個主題的 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 SQS 佇列以取得主題。

```
        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;
           }
```

使用篩選器訂閱主題。

```
       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 ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;
```

```
                    request.AddAttributes("FilterPolicy", jsonPolicy);
                }
                else {
                    std::cout
                            << "Because you did not select any attributes, no
 filter "
                            << "will be added to this subscription." <<
 std::endl;
                }
            }
        }  // if (isFifoTopic)
        Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

        if (outcome.IsSuccess()) {
            Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
            std::cout << "The queue '" << queueName
                      << "' has been subscribed to the topic '"
                      << "'" << topicName << "'" << std::endl;
            std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                      << std::endl;
            subscriptionARNS.push_back(subscriptionARN);
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Subscribe. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

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

            return false;
        }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
```

```
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";
```

```
        result = jsonPolicyStream.str();
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for C++ 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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

使用篩選條件訂閱主題的佇列。

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



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
```

```
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考中的訂閱。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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);
        }
    }
}
```

訂閱某個主題的 HTTP 端點。

```
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);
        }
    }
}
```

訂閱某個主題的 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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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'
  // }
};
```

訂閱某個主題的行動應用程式。

```
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;
};
```

訂閱某個主題的 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 SQS 佇列以取得主題。

```
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;
};
```

使用篩選器訂閱主題。

```
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-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 如需詳細資訊，請參閱《AWS SDK for JavaScript 開發人員指南》。

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的訂閱。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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()
    }
}
```

訂閱某個主題的 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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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());
}
```

訂閱某個主題的 HTTP 端點。

```
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 AWS SDK for PHP API 參考中的訂閱。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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
```

- 如需詳細資訊，請參閱《AWS SDK for Ruby 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的訂閱。

Rust

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

訂閱主題的電子郵件地址。

```
    TRY.
        oo_result = lo_sns->subscribe(                      "oo_result is
returned for testing purposes."
                iv_topicarn = iv_topic_arn
                iv_protocol = 'email'
```

```
                iv_endpoint = iv_email_address
                iv_returnsubscriptionarn = abap_true
            ).
        MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.
```

- 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。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭TagResource配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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TagResource中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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TagResource中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK TagResource中的 Kotlin API 參考。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭Unsubscribe配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 AWS SDK for .NET API 參考中的取消訂閱。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
//! 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 AWS SDK for C++ 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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for JavaScript 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的取消訂閱。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的取消訂閱。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    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。

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 開發套件使用 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 開發套件的 Amazon SNS 案例

下列程式碼範例說明如何使用 AWS SDK 在 Amazon SNS 中實作常見案例。這些案例會向您展示如何呼叫 Amazon SNS 中的多個函數來完成特定任務。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

範例

- 使用 AWS 開發套件為 Amazon SNS 推送通知建立平台端點
- 使用開發套件 AWS 建立並發佈至 FIFO Amazon SNS 主題

- 使用開發 AWS 套件將簡訊發佈到 Amazon SNS 主題
- 使用開發 AWS 套件使用 Amazon S3 將大型訊息發佈到 Amazon SNS
- 使用開發 AWS 套件發佈 Amazon SNS 簡訊
- 使用開發套件將 Amazon SNS 訊息發佈到 Amazon SQS 佇列 AWS

使用 AWS 開發套件為 Amazon SNS 推送通知建立平台端點

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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

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

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

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

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件 AWS 建立並發佈至 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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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");

            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

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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),
                },
            )
            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《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - CreateTopic
  - 發布
  - Subscribe

SAP ABAP

適用於 SAP ABAP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                          "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
```

```
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

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

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

- 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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發 AWS 套件將簡訊發佈到 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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發 AWS 套件使用 Amazon S3 將大型訊息發佈到 Amazon SNS

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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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());
        }
}
```

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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發 AWS 套件發佈 Amazon SNS 簡訊

下列程式碼範例示範如何使用 Amazon SNS 發佈簡訊。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 AWS SDK for .NET API 參考中的發佈。

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 * 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 AWS SDK for C++ API 參考中的發佈。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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《AWS SDK for PHP 開發人員指南》。
- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的發佈。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件將 Amazon SNS 訊息發佈到 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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

在命令提示中執行互動式案例。

```
/// <summary>
/// Console application to run a workflow scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues workflow is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
```

```
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the workflow.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this workflow, you will create an SNS topic and
 subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
 configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
```

```
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"An AWS Identity and Access Management (IAM) policy must
 be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
 queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
```

```
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
{_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
filters.");
            for (int i = 0; i < _tones.Length; i++)
```

```
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

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

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
 a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
 set a message group ID." +
```

```
                                "\r\nAll messages within the same group will be
received in the order " +
                                "they were published.");

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

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

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

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

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

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

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

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

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

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
```

```
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
```

```
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
```

```
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

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="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

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

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

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
```

```
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

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

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
```

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

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

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

```
    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

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

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- 發布
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

C++

適用於 C++ 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

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

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;
```

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

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or
 automatically generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }
```

```
    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

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

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

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

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }
```

```
            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);
```

```
            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

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

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
```

```
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
```

```
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
```

```
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
```

```
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

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

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

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

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

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);
```

```
            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

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

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
```

```
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}
```

```
bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "Unsubscribe of subscription ARN '" <<
 subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

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

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
```

```
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for C++ 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**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

在命令提示中執行互動式案例。

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic() (string, string, bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
```

```
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}
```

```
func (runner ScenarioRunner) CreateQueue(ordinal string, isFifoTopic bool)
 (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
 }
 log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
  "messages to it.")
 log.Println(strings.Repeat("-", 88))

 var filterPolicy map[string][]string
 if isFifoTopic {
  if ordinal == "first" {
   log.Println("Subscriptions to a FIFO topic can have filters.\n" +
```

```
    "If you add a filter to this subscription, then only the filtered messages\n"
 +
    "will be received in the queue.\n" +
    "For information about message filtering, see\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
    "For this example, you can filter messages by a \"tone\" attribute.")
  }

  wantFiltering := runner.questioner.AskBool(
   fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
    "from the %v topic? (y/n) ", queueName, topicName), "y")
  if wantFiltering {
   log.Println("You can filter messages by one or more of the following \"tone\"
 attributes.")

   var toneSelections []string
   askAboutTones := true
   for askAboutTones {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelections = append(toneSelections, ToneChoices[toneIndex])
    askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
 the filter? (y/n) ", "y")
   }
   log.Printf("Your subscription will be filtered to only pass the following
 tones: %v\n", toneSelections)
   filterPolicy = map[string][]string{TONE_KEY: toneSelections}
  }
 }

 subscriptionArn, err := runner.snsActor.SubscribeQueue(topicArn, queueArn,
 filterPolicy)
 if err != nil {
  panic(err)
 }
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(topicArn string, isFifoTopic bool,
 contentBasedDeduplication bool, usingFilters bool) {
```

```
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
```

```
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(queueUrl, messages)
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
```

```
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, you will create an SNS topic and subscribe %v SQS queues to
 the\n"+
  "topic. You can select from several options for configuring the topic and the
\n"+
  "subscriptions for the queues. You can then post to the topic and see the
 results\n"+
  "in the queues.\n", queueCount)

 log.Println(strings.Repeat("-", 88))

 runner := ScenarioRunner{
  questioner: questioner,
  snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
  sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
 }
 resources.snsActor = runner.snsActor
 resources.sqsActor = runner.sqsActor
```

```
 topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
 runner.CreateTopic()
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup()
 }

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

定義一個結構，以包裝此範例中使用的 Amazon SNS 動作。

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



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

 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &sns.DeleteTopicInput{
```

```
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(topicArn string, queueArn string,
 filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(topicArn string, message string, groupId string,
 dedupId string, filterKey string, filterValue string) error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

定義一個結構，以包裝此範例中使用的 Amazon SQS 動作。

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
```

```
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
```

```
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(queueUrl string, queueArn string,
 topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(context.TODO(),
 &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}
```

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}
```

```
// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(queueUrl string, messages []types.Message)
 error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(context.TODO(),
 &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- 發布
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

JavaScript

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

這是此工作流程的進入點。

```
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";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

先前的程式碼提供了必要的相依性並啟動工作流程。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.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
        queueUrl: response.QueueUrl,
```

```
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
          "${QUEUE_NAME}",
```

```
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
            "${QUEUE_NAME}",
```

```
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

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

    let groupId, deduplicationId, choices;

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

      if (this.autoDedup === false) {
```

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

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

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

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

    if (publishAnother) {
```

```
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

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

```
    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
```

```
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript 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 開發套件使用 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 AWS 用開發套件的 Amazon SNS 的無伺服器範例

下列程式碼範例說明如何搭配 AWS 開發套件使用 Amazon SNS。

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

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。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.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }
```

```
    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK for Go V2

> **Note**
>
> 還有更多關於 GitHub。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

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。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 與 Lambda 一起使用 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

適用於 JavaScript (v3) 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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。

透過使 Lambda JavaScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

透過使 Lambda TypeScript.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

適用於 PHP 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。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

適用於 Ruby 的開發套件

> **Note**
>
> 還有更多關於 GitHub。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。

使用紅寶石使用與 Lambda 的 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

適用於 Rust 的 SDK

> **Note**
>
> 還有更多關於 GitHub。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 開發套件使用 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 AWS 用開發套件的 Amazon SNS 跨服務範例

下列範例應用程式使用 AWS 開發套件將 Amazon SNS 與其他 AWS 服務應用程式結合使用。每個範例都包含一個連結 GitHub，您可以在其中找到如何設定和執行應用程式的指示。

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
- 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
- 使用開發套件使用 Amazon Rekognition 偵測影片中的人物和物件 AWS
- 使用開發套件將 Amazon SNS 訊息發佈到 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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- DynamoDB
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的開發套件

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例也可在 AWS SDK for JavaScript v3 開發人員指南中取得。

此範例中使用的服務

- DynamoDB
- Amazon SNS

Kotlin

適用於 Kotlin 的 SDK

示範如何使用 Amazon DynamoDB Kotlin API 建立提交資料的原生 Android 應用程式，並使用 Amazon SNS Kotlin API 傳送文字訊息。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

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 開發套件使用 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

AWS SDK for .NET

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon SNS
- Amazon Translate

Java

適用於 Java 2.x 的 SDK

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

有關如何設置和運行使用 Java Async API 的示例的完整源代碼和說明，請參閱上GitHub的完整示例。

此範例中使用的服務

- Amazon SNS

- Amazon Translate

Kotlin

適用於 Kotlin 的 SDK

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。

有關如何創建 Web 應用程序的完整源代碼和說明，請參閱中的完整示例GitHub。

有關如何創建本機 Android 應用程序的完整源代碼和說明，請參閱上的完整示例GitHub。

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 開發套件使用 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

AWS SDK for .NET

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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

JavaScript

適用於 JavaScript (v3) 的開發套件

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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

適用於 PHP 的開發套件

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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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

適用於 Rust 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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例 GitHub。

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 開發套件使用 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) 的開發套件

示範如何使用建置 React 應用程式，該應用程式使用 Amazon Textract 擷取文件影像中的資料，並將其顯示在互動式網頁中。 AWS SDK for JavaScript 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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

適用於 Python (Boto3) 的 SDK

示範如何使用 AWS SDK for Python (Boto3) 搭配 Amazon Textract 來偵測文件影像中的文字、表單和表格元素。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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- 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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件使用 Amazon Rekognition 偵測影片中的人物和物件 AWS

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。

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。

此範例最佳檢視時，請參閱 GitHub。有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- Amazon Rekognition
- 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 開發套件使用 Amazon SNS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用開發套件將 Amazon SNS 訊息發佈到 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。

Java

適用於 Java 2.x 的 SDK

使用 Amazon Simple Notification Service (Amazon SNS) 和 Amazon Simple Queue Service (Amazon SQS) 示範主題和佇列的訊息。

如需在 Amazon SNS 和 Amazon SQS 中示範包含主題和佇列的完整原始程式碼和指示，請參閱上GitHub的完整範例。

此範例中使用的服務

- Amazon SNS
- Amazon SQS

Kotlin

適用於 Kotlin 的 SDK

使用 Amazon Simple Notification Service (Amazon SNS) 和 Amazon Simple Queue Service (Amazon SQS) 示範主題和佇列的訊息。

如需在 Amazon SNS 和 Amazon SQS 中示範包含主題和佇列的完整原始程式碼和指示，請參閱上GitHub的完整範例。

此範例中使用的服務

- 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 開發套件使用 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

下列程式碼範例說明如何建立 Amazon API Gateway 叫用的 AWS Lambda 函數。

Java

適用於 Java 2.x 的 SDK

示範如何使用 Lambda Java 執行階段 API 建立 AWS Lambda 函數。此範例會呼叫不同的 AWS 服務來執行特定使用案例。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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的開發套件

示範如何使用 Lambda JavaScript 執行階段 API 建立 AWS Lambda 函數。此範例會呼叫不同的 AWS 服務來執行特定使用案例。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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例也可在 AWS SDK for JavaScript 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

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 開發套件使用 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

下列程式碼範例說明如何建立 Amazon EventBridge 排程事件所叫用的 AWS Lambda 函數。

Java

適用於 Java 2.x 的 SDK

說明如何建立叫用 AWS Lambda 函數的 Amazon EventBridge 排程事件。設定 EventBridge 為在叫用 Lambda 函數時使用 cron 運算式來排程。在此範例中，您會使用 Lambda Java 執行期 API 建立 Lambda 函數。此範例會呼叫不同的 AWS 服務來執行特定使用案例。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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例中使用的服務

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

適用於 JavaScript (v3) 的開發套件

說明如何建立叫用 AWS Lambda 函數的 Amazon EventBridge 排程事件。設定 EventBridge 為在叫用 Lambda 函數時使用 cron 運算式來排程。在此範例中，您可以使用 Lambda JavaScript

執行階段 API 建立 Lambda 函數。此範例會呼叫不同的 AWS 服務來執行特定使用案例。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。

有關如何設置和運行的完整源代碼和說明，請參閱中的完整示例GitHub。

此範例也可在 AWS SDK for JavaScript v3 開發人員指南中取得。

此範例中使用的服務

- 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 開發套件使用 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

本節提供 Amazon SNS 安全性、身分驗證和存取控制以及 Amazon SNS 存取原則語言的相關資訊。

主題

資料保護

AWS 共同的責任模型適用於 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 AWS 雲端 的全球基礎設施。您必須負責維護在此基礎設施上託管之內容的控制權。此內容包括您所使用 AWS 服務的安全組態和管理任務。如需有關資料隱私權的詳細資訊，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，建議您使用 AWS 帳戶 (IAM) 保護 AWS Identity and Access Management 憑證，並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：

- 每個帳戶都使用多重要素驗證 (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 AWS CloudTrail 設定 API 和使用者活動記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務內的所有預設安全控制。
- 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。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的詳細資訊，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 Amazon SNS 或使用主控台、API、AWS CLI 或 AWS 開發套件的其他 Amazon Web Services。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。

以下各節提供有關 Amazon SNS 中內容保護的其他資訊。

主題

- 資料加密
- 網際網路流量隱私權
- 訊息資料保護安全性

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。您可以請求 Amazon SNS 加密訊息，然後再將訊息儲存到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 進行最佳化的資料加密。

主題

- 靜態加密
- 金鑰管理
- 對 Amazon SNS 主題啟用伺服器端加密 (SSE)

- 為已訂閱加密 Amazon SQS 佇列的 Amazon SNS 主題啟用伺服器端加密 (SSE)

靜態加密

伺服器端加密 (SSE) 可讓您使用 () 中管理的金鑰來保護 Amazon SNS 主題中訊息的內容，藉此將敏感資料儲存在AWS Key Management Service加密主題中。AWS KMS

Amazon SNS 一收到訊息，SSE 就會將其加密。這些消息以加密形式存儲，並且僅在發送時才解密。

- 如需使用 AWS Management Console 或 AWS SDK for Java 來管理 SSE (透過使用 `CreateTopic` 和 `KmsMasterKeyId` API 動作來設定 `SetTopicAttributes` 屬性) 的詳細資訊，請參閱 對 Amazon SNS 主題啟用伺服器端加密 (SSE)。
- 如需使用 AWS CloudFormation 建立加密主題 (透過使用 `AWS::SNS::Topic` 資源來設定 `KmsMasterKeyId` 屬性) 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 結合了安全且高可用軟硬體的服務，可提供針對雲端調整的金鑰管理系統。當您使用 Amazon SNS 搭配 AWS KMS，加密訊息資料的資料金鑰也會一併加密並與它們保護的資料一起存放。

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。
- 您也可以為 Amazon SNS 使用 AWS 受管 KMS 金鑰，每個帳戶和區域都是唯一的。
- AWS KMS 安全標準可以協助您符合加密相關合規要求。

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？。

主題

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

AWS KMS key的別名、別名 ARN、金鑰 ID 或金鑰 ARN，或在您的帳戶或其他帳戶中的自訂 AWS KMS。適用於 Amazon SNS 的 AWS 受管 AWS KMS 的別名一律是 `alias/aws/sns`，而自訂 AWS KMS 的別名可以是 `alias/MyAlias` (舉例來說)。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 以針對某個主題指定 Amazon SNS 的 AWS 受管 KMS，AWS KMS 會建立 Amazon SNS 的 AWS 受管 KMS。
> - 或者，第一次使用 `Publish` 在啟用 SSE 的主題上執行動作、AWS KMS 會建立 Amazon SNS 的 AWS 受管 KMS。

您可以使用 AWS KMS 主控台的 AWS KMS keys區段，或使用 `CreateKey` AWS KMS 動作來建立 AWS KMS 金鑰、定義控制 AWS KMS 金鑰使用方式的政策，以及稽核 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AWS KMS識別碼的更多範例，請參閱 AWS Key Management ServiceAPI 參考資料KeyId中的。如需尋找 AWS KMS 識別碼的資訊，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 AWS KMS 需另外付費。如需詳細資訊，請參閱 估算 AWS KMS 成本 和 AWS Key Management Service定價。

金鑰管理

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。深入了解

> **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。

主題

- 估算 AWS KMS 成本
- 設定 AWS KMS 許可
- AWS KMS 錯誤

估算 AWS KMS 成本

若要預測成本並更佳了解您的 AWS 帳單，您可能想要知道 Amazon SNS 使用您 AWS KMS key的頻率。

> **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 個發布者和 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

您必須先將 AWS KMS key政策設定為允許加密主題以及加密和解密訊息，才可以使用 SSE。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 設定使用伺服器端加密的 Amazon SNS 主題。

> **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，請參閱使用 IAM 政策搭配 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。

您也必須確定 AWS KMS key的金鑰政策允許必要的許可。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，您也可以在指派給委託人 (這些委託人在 Amazon SNS 中發布和訂閱以接收加密訊息) 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 AWS 服務的事件來源和加密主題之間的相容性

許多AWS 服務會發布事件到 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 服務具備 `kms:GenerateDataKey*` 和 `kms:Decrypt` 許可，請將以下陳述式加入 KMS 政策中。

```
{
  "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
AWS CodeCommit	codecommit.amazonaws.com
AWS CodeStar	codestar-notifications.amazonaws.com
AWS Database Migration Service	dms.amazonaws.com
AWS Directory Service	ds.amazonaws.com
Amazon DynamoDB	dynamodb.amazonaws.com
Amazon Inspector	inspector.amazonaws.com
Amazon Redshift	redshift.amazonaws.com
Amazon RDS	events.rds.amazonaws.com
Amazon S3 Glacier	glacier.amazonaws.com
Amazon Simple Email Service	ses.amazonaws.com
Amazon Simple Storage Service	s3.amazonaws.com
AWS Snowball	importexport.amazonaws.com
AWS Systems Manager Incident Manager	AWSSystems 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` 和 `aws:SourceArn` 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-
type: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

當您搭配使用 Amazon SNS 和 AWS KMS，可能會發生錯誤。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 需要訂閱服務。

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

對 Amazon SNS 主題啟用伺服器端加密 (SSE)

使用伺服器端加密 (SSE) 可讓您儲存加密主題中的敏感資料。SSE 使用 AWS Key Management Service (AWS KMS) 中管理的金鑰來保護 Amazon SNS 主題中的訊息內容。如需 Amazon SNS 的伺服器端加密的詳細資訊，請參閱 靜態加密。如需建立 AWS KMS 金鑰的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的建立金鑰。

> **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。

使用 AWS Management Console 對 Amazon SNS 主題啟用伺服器端加密 (SSE)

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，選擇主題，然後選擇 Actions (動作)、Edit (編輯)。
4. 展開 Encryption (加密) 區段並執行下列動作：

   a. 選擇 Enable encryption (啟用加密)。

   b. 指定為 AWS KMS 金鑰。如需更多詳細資訊，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。

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

  > **Note**
  >
  > 請謹記以下幾點：
  >
  > - 第一次使用 AWS Management Console 以針對某個主題指定 Amazon SNS 的 AWS 受管 KMS，AWS KMS 會建立 Amazon SNS 的 AWS 受管 KMS。
  > - 或者，第一次使用 `Publish` 在啟用 SSE 的主題上執行動作、AWS KMS 會建立 Amazon SNS 的 AWS 受管 KMS。

- 若要使用 AWS 帳戶中的自訂 KMS，請選擇 KMS 金鑰欄位，然後從清單中選擇自訂 KMS。

  > **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

- 若要使用 AWS 帳戶或其他 AWS 帳戶中的自訂 KMS ARN，請將它輸入至 KMS 金鑰欄位中。

5. 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 Encryption (加密) 狀態、AWSAccount (帳戶)、Customer master key (CMK) (客戶主金鑰 (CMK))、CMK ARN，以及 Description (描述) 會顯示在 Encryption (加密) 索引標籤。

設定使用伺服器端加密的 Amazon SNS 主題

建立 KMS 金鑰時，請使用下列 KMS 金鑰政策：

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "service.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
```

```
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

為已訂閱加密 Amazon SQS 佇列的 Amazon SNS 主題啟用伺服器端加密 (SSE)

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 靜態加密。

本頁說明如何使用 AWS Management Console 在訂閱加密 Amazon SQS 佇列的 Amazon SNS 主題中啟用 SSE。

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，以代表您帳戶中的 AWS 受管金鑰。

   > **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，請編輯 AWS Management Console 中的描述或使用 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。

   將標籤新增到 AWS 資源時，AWS 會產生成本配置報告，內含按標籤彙總的用量與成本。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**
>
> 此金鑰政策授予 AWS 帳戶 完全控制此 KMS 金鑰。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。如需更多詳細資訊，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。

網際網路流量隱私權

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) 來託管您的 AWS 資源，可以在您的 VPC 與 Amazon SNS 之間建立私人連線。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，您可用來在自己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。這類端點可讓您將 VPC 連線到 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 使用者指南中的界面 VPC 端點。

本節資訊適用於 Amazon VPC 使用者。如需詳細資訊以及開始建立 VPC 的方法，請參閱 Amazon VPC 使用者指南中的 Amazon VPC 入門。

> **Note**
>
> VPC 端點不允許您訂閱私有 IP 地址的 Amazon SNS 主題。

主題

- 為 Amazon SNS 建立 Amazon VPC 端點
- 為 Amazon SNS 建立 VPC 端點政策
- 從 Amazon VPC 發佈 Amazon SNS 訊息

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 開發套件 AWS Management Console、Amazon SNS API 或 AWS CloudFormation在 VPC 中建立 Amazon SNS 端點。 AWS CLI

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

如需使用建立和設定端點的相關資訊 AWS CloudFormation，請參閱使AWS CloudFormation 用指南中的`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 Instances (執行個體)、選擇您的執行個體，然後找到 Public DNS (IPv4) (公有 DNS (IPv4)) 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 是該主題所在的 AWS 區域。
- sns-topic-arn是該主題的 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：

- 可執行動作的委託人。
- 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。

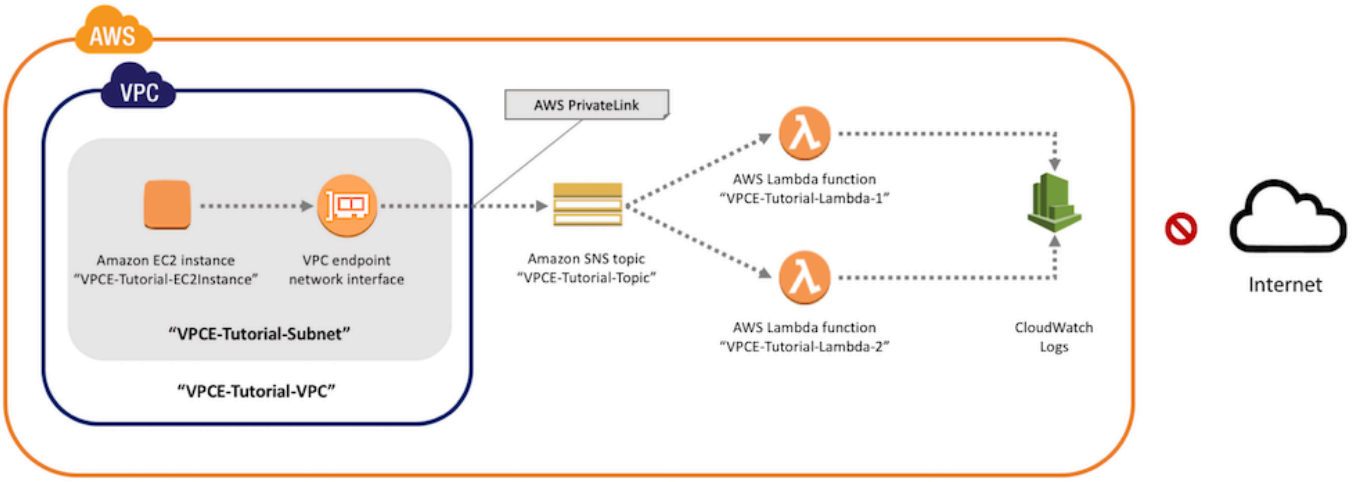
從 Amazon VPC 發佈 Amazon SNS 訊息

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。該訊息會保存在 AWS 網路中，不會進入公有網際網路。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，以在 AWS 帳戶 帳戶中自動建立臨時私有網路。
- 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。
- 刪除您在此過程中建立的資源，使它們不會保留在您的 AWS 帳戶 中。

下圖說明您完成這些步驟時，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。這類端點連接至採用 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。該主題會將收到的訊息散發給兩個訂閱 AWS Lambda 函數。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

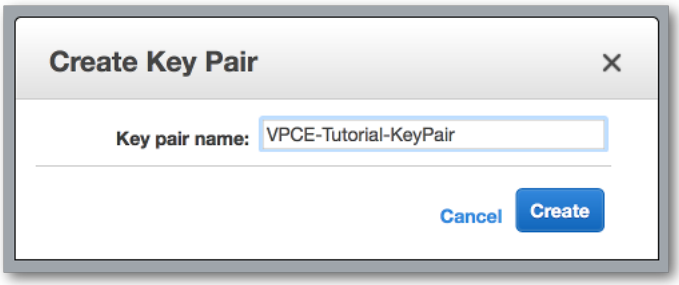
開始之前，您需要 Amazon Web Services (AWS) 帳戶。註冊時，您的帳戶會自動註冊所有 AWS 中的服務，包括 Amazon SNS 及 Amazon VPC。如果您尚未建立帳戶，請前往 https://aws.amazon.com/，然後選擇 Create a Free Account (建立免費帳戶)。

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. 請登入 AWS Management Console，並在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
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，可使用 AWS CloudFormation 範本。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。

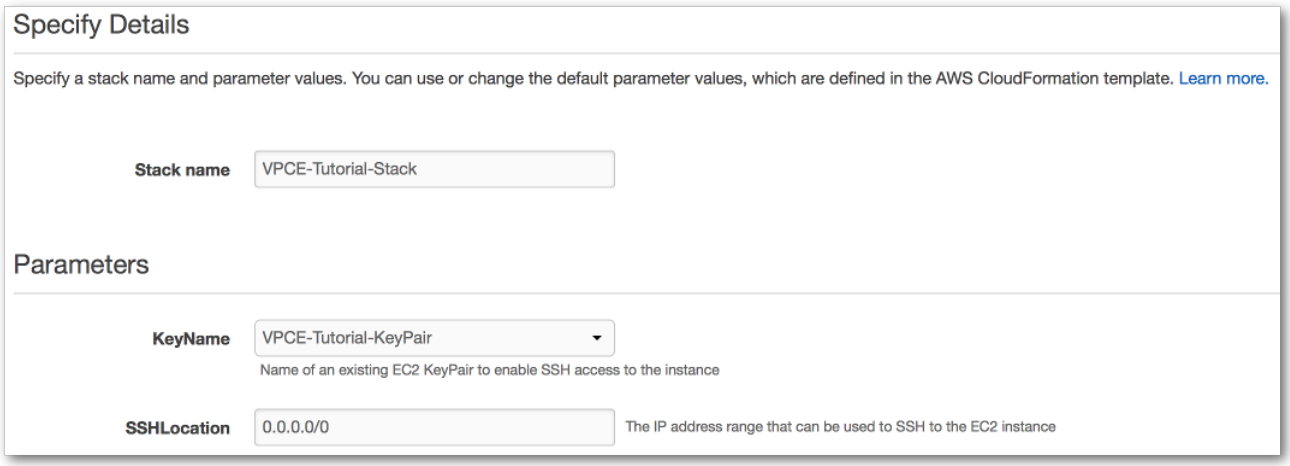
您提供範本至 AWS CloudFormation，而 AWS CloudFormation 則在您的 AWS 帳戶 帳戶中佈建所需的資源，以做為堆疊。堆疊是一組視為單一單位進行管理的資源。當您完成步驟後，您可以使用 AWS CloudFormation 一次刪除堆疊中所有資源。除非您希望保留這些資源，否則它們不會保留在您的 AWS 帳戶 中。

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

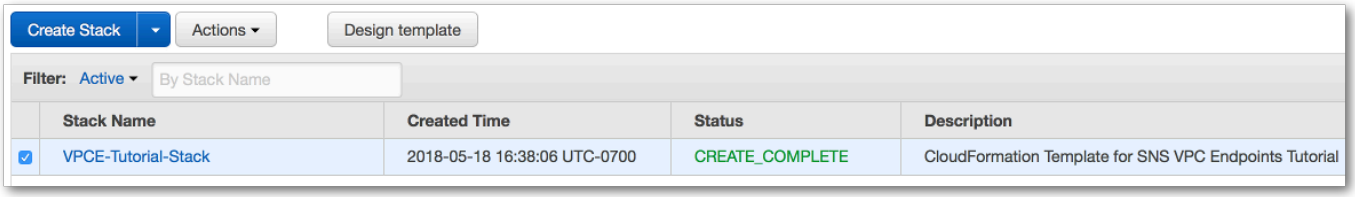
1. 從 GitHub 網站下載範本檔案。
2. 登入 AWS CloudFormation 主控台。
3. 請選擇 Create Stack (建立堆疊)。
4. 在 Select Template (選擇範本) 頁面中，選擇 Upload a template to Amazon S3 (上傳範本到 Amazon S3)、選擇檔案，並選擇 Next (下一步)。
5. 在 Specify Details (指定詳細資訊) 頁面，指定堆疊和金鑰名稱：
   a. 在 Stack Name (堆疊名稱) 中輸入 **`VPCE-Tutorial-Stack`**。
   b. 對於 KeyName (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`**。



d. 選擇 Next (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 Capabilities (功能) 下，確認 AWS CloudFormation 可能會使用自訂名稱建立 IAM 資源。
9. 選擇 Create (建立)。

   AWS CloudFormation 主控台會開啟 Stacks (堆疊) 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)。



> **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. 在 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. 複製在 Public DNS (IPv4) (公有 DNS (IPv4)) 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

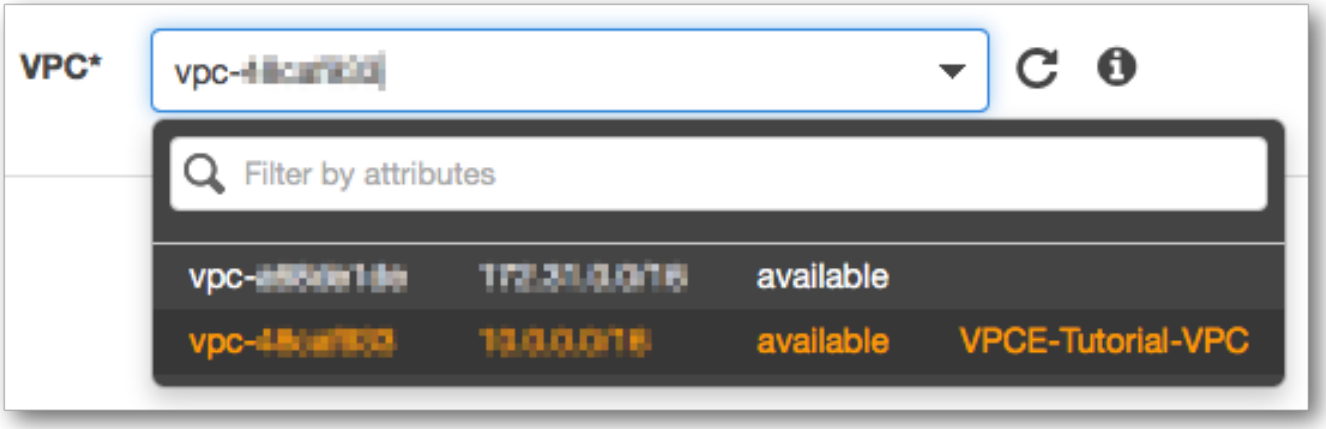
將 VPC 連接到 Amazon SNS，請定義介面 VPC 端點。在您新增端點後，您便可在 VPC 中登入 Amazon EC2 執行個體，且從這裡您可使用 Amazon SNS API。您可以將訊息發佈到主題，該訊息為私下發佈。它們會保持在 AWS 網路中，且不會進出公有網際網路。

> **Note**
>
> 該執行個體仍無法存取網際網路上其他 AWS 服務和端點。

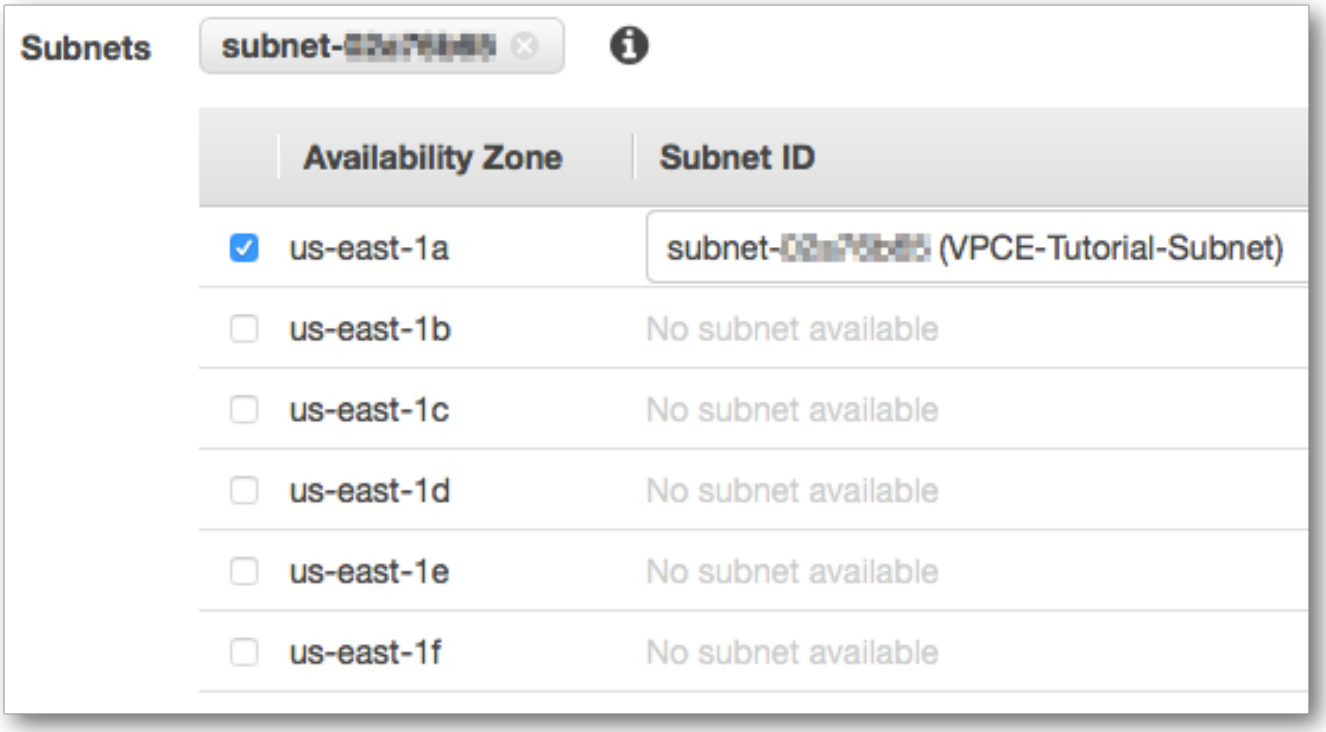
建立端點

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。

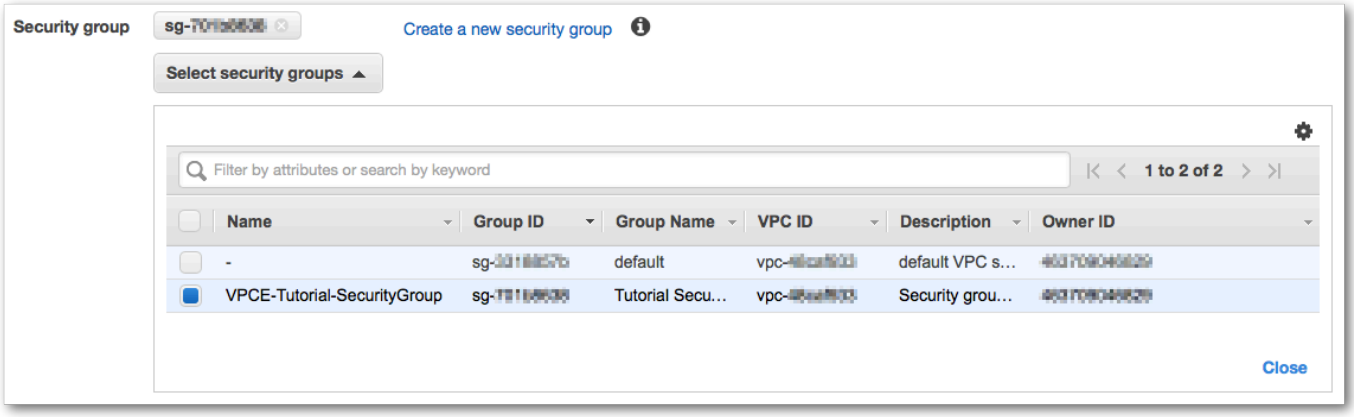
   該服務名稱取決於所選區域。例如，如果您選擇美國東部 (維吉尼亞北部)，服務名稱會是 com.amazonaws.*us-east-1*.sns。
6. 對於 VPC，選擇名稱為 VPCE-Tutorial-VPC (VPCE-教學-VPC) 的 VPC。



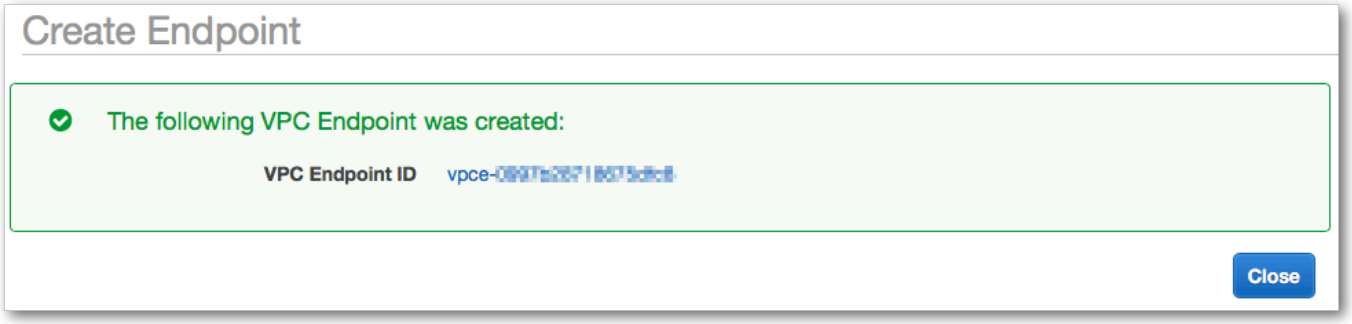
7. 對於 Subnets (子網路)，在子網路 ID 中選擇有 VPCE-Tutorial-Subnet 的子網路。



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)。

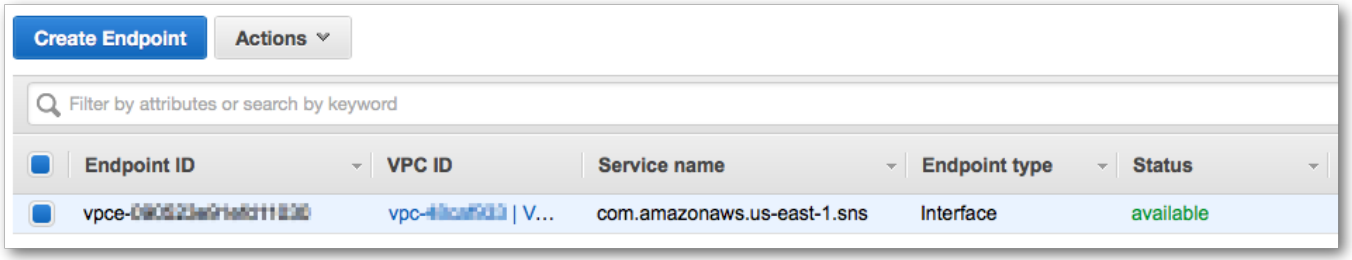


10. 選擇 Create endpoint (建立端點)。Amazon VPC 主控台確認 VPC 端點已建立。



11. 選擇 Close (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"
}
```

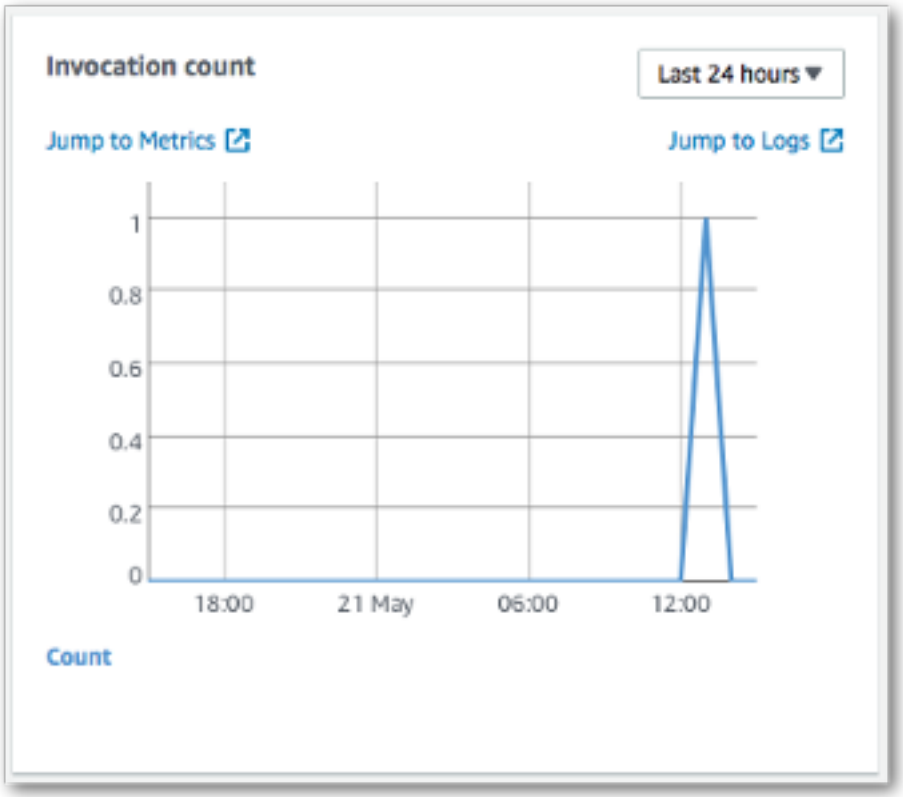
步驟 6：驗證您的訊息交付

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

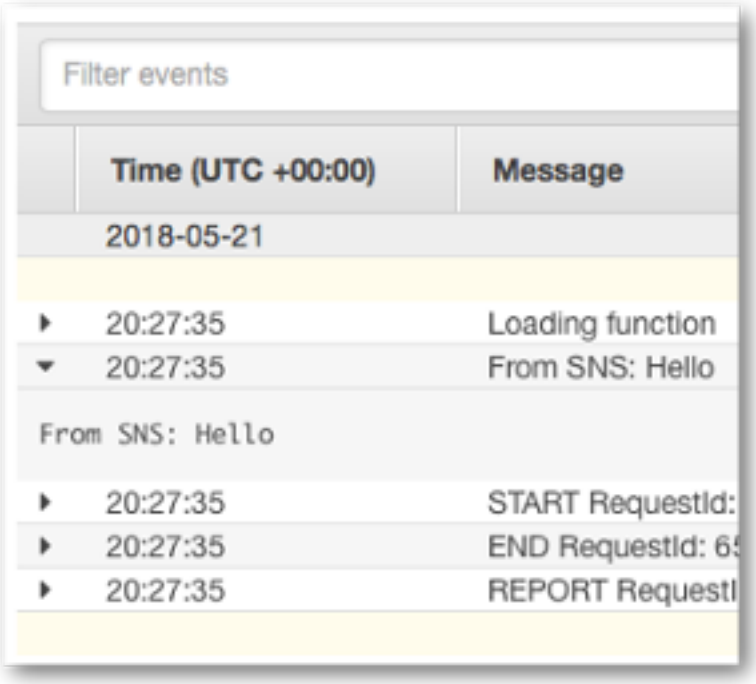
1. 於 https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 在 Functions (函數) 頁面，選擇 VPCE-Tutorial-Lambda-1 (VPCE-教學-Lambda-1)。
3. 選擇 Monitoring (監控)。
4. 檢查 Invocation count (呼叫次數) 圖表。此圖表顯示 Lambda 函數已執行的次數。

   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. 在 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`。



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。透過刪除您不再使用的 AWS 資源，可為 AWS 帳戶 避免不必要的費用。

首先，請使用 Amazon VPC 主控台刪除您的 VPC 端點。然後，透過在 AWS CloudFormation 主控台中刪除堆疊，您可以刪除您所建立的其他資源。刪除堆疊時，AWS CloudFormation 會從 AWS 帳戶移除堆疊的資源。

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。

刪除您的 AWS CloudFormation 堆疊

1. 開啟位於 https://console.aws.amazon.com/cloudformation 的 AWS CloudFormation 主控台。
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 安全性部落格：透過 AWS PrivateLink 保護發佈至 Amazon SNS 之訊息的安全
- 什麼是 Amazon VPC？
- VPC 端點
- 什麼是 Amazon EC2？
- AWS CloudFormation 概念

訊息資料保護安全性

- 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的內容，而不是靜態資料。
- 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權限授與個別訂閱者，並稽核整個內容流程程序。

Amazon SNS 中的 Identity and Access Management

存取 Amazon SNS 時需要提供登入資料，以供 AWS 驗證您的請求。這些登入資料必須有權存取 AWS 資源，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，讓管理員能夠安全地控制對 AWS 資源的存取權。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 是一種您可以免費使用的 AWS 服務。

物件

AWS Identity and Access Management (IAM) 的使用方式會不同，取決於您在 Amazon SNS 中所執行的工作。

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。瞭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 如何搭配使用 Amazon Simple Notification Service 與 IAM。

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

身分驗證是使用身分憑證登入 AWS 的方式。您必須以 AWS 帳戶 根使用者身分、IAM 使用者身分，或使用 IAM 角色進行 驗證 (登入至 AWS)。

您可以使用透過身分來源提供的憑證，以聯合身分登入 AWS。AWS IAM Identity Center(IAM Identity Center) 使用者、貴公司的單一登入身分驗證和您的 Google 或 Facebook 憑證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。您 AWS 藉由使用聯合進行存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入至 AWS 的相關資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

如果您是以程式設計的方式存取 AWS，AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的憑證透過密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，您必須自行簽署請求。如需使用建議的方法自行簽署請求的相關資訊，請參閱《IAM 使用者指南》中的簽署 AWS API 請求。

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 建議您使用多重要素驗證 (MFA) 以提高帳戶的安全。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》中的在 AWS 中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

如果是建立 AWS 帳戶，您會先有一個登入身分，可以完整存取帳戶中所有 AWS 服務 與資源。此身分稱為 AWS 帳戶 根使用者，使用建立帳戶時所使用的電子郵件地址和密碼即可登入並存取。強烈建議您不要以根使用者處理日常作業。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)搭配身分提供者使用聯合功能，使用暫時憑證來存取 AWS 服務。

聯合身分是來自您企業使用者目錄的使用者、Web 身分供應商、AWS Directory Service、Identity Center 目錄或透過身分來源提供的憑證來存取 AWS 服務 的任何使用者。聯合身分存取 AWS 帳戶時，會擔任角色，並由角色提供暫時憑證。

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，也可以連線並同步到自己身分來源中的一組使用者和群組，以便在您的所有 AWS 帳戶和應用程式中使用。如需 IAM Identity Center 的相關資訊，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 使用者是您 AWS 帳戶 中的一種身分，具備單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，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。如需詳細資訊，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。您無法以群組身分登入。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過程變得更為容易。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，但角色僅提供暫時憑證。若要進一步了解，請參閱《IAM 使用者指南》中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 中的一種身分，具備特定許可。它類似 IAM 使用者，但不與特定的人員相關聯。您可以在 AWS Management Console 中透過切換角色來暫時取得 IAM 角色。您可以透過呼叫 AWS CLI 或 AWS API 操作，或是使用自訂 URL 來取得角色。如需使用角色的方法的相關資訊，請參閱《IAM 使用者指南》中的使用 IAM 角色。

使用暫時憑證的 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，並取得由角色定義的許可。如需有關聯合角色的詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-idp.html中的為第三方身分提供者建立角色。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 (信任的委託人)存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，針對某些 AWS 服務，您可以將政策直接連接到資源 (而非使用角色作為代理)。若要了解跨帳戶存取角色和資源型政策間的差異，請參閱《IAM 使用者指南》中的 IAM 角色與資源類型政策的差異。
- 跨服務存取 – 有些 AWS 服務 會使用其他 AWS 服務 中的功能。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。
  - 轉發存取工作階段 (FAS)：當您使用 IAM 使用者或角色在 AWS 中執行動作時，系統會將您視為主體。當您使用某些服務時，您可能會執行一個動作，而該動作之後會在不同的服務中啟動另一個動作。FAS 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。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之後，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給 AWS 服務 服務。
- 服務連結角色 – 服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。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 – 針對在 EC2 執行個體上執行並提出 AWS CLI 和 AWS API 請求的應用程式，您可以使用 IAM 角色來管理暫時憑證。這是在 EC2 執行個體內儲存存取金鑰的較好方式。如需指派 AWS 角色給 EC2 執行個體並提供其所有應用程式使用，您可以建立連接到執行個體的執行個體設定檔。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暫時憑證。如需詳細資訊，請參閱《IAM 使用者指南》中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱《IAM 使用者指南》中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透過建立政策並將其附加到 AWS 身分或資源，在 AWS 中控制存取。政策是 AWS 中的一個物件，當其和身分或資源建立關聯時，便可定義其許可。AWS 會在主體 (使用者、根使用者或角色工作階段)發出請求時評估這些政策。政策中的許可，決定是否允許或拒絕請求。大部分政策以 JSON 文件形式儲存在 AWS 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哪些內容。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。若要授與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。具備該政策的使用者便可以從 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。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

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。受管政策則是獨立的政策，您可以將這些政策附加到 AWS 帳戶 中的多個使用者、群組和角色。受管政策包含 AWS 管理政策和客戶管理政策。如需瞭解如何在受管政策及內嵌政策間選擇，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。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。主體可以包括帳戶、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) 可控制哪些委託人 (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 Simple Storage Service (Amazon S3)、AWS WAF 和 Amazon VPC 是支援 ACL 的服務範例。若要進一步了解 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 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授與您的最大許可。

- 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。如需許可範圍的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可範圍。

- 服務控制政策 (SCP) – SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。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 會限制成員帳戶中的實體許可，包括每個 AWS 帳戶 的根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。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。如需更多資訊，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。如需瞭解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，請參閱 IAM 使用者指南中的政策評估邏輯。

存取控制

Amazon SNS 本身有以資源為基礎的許可系統，該系統使用與 AWS Identity and Access Management (IAM) 政策相同的語言所撰寫的政策。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 帳戶可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。

在 Amazon SNS 中管理存取的概觀

本節說明使用存取原則語言撰寫政策所需要了解的基本概念。同時也說明存取控制如何搭配存取原則語言以及如何評估政策的一般程序。

主題

- 使用存取控制的時機
- 重要概念
- 架構概觀

- 使用存取原則語言
- 評估邏輯
- Amazon SNS 存取控制的範例案例

使用存取控制的時機

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：

- 您想要授予其他 AWS 帳戶 帳戶特定類型的主題動作 (例如發佈)。如需詳細資訊，請參閱 授予 AWS 帳戶 存取主題。
- 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。

主題

- 權限
- 陳述式
- 政策
- 發行者
- Principal
- 動作
- 資源
- 條件和金鑰
- 要求者
- 評估
- Effect
- 預設拒絕
- 允許

- 明確拒絕

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。您可以利用 `aws:SourceAccount` 和 `aws:SourceOwner` 來解決需要插入包含非 ASCII 字元的其他 AWS 服務 ARN。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。根據定義，發行者一律為資源擁有者。AWS 不允許 AWS 服務使用者為不是自己所擁有的資源建立政策。如果 John 是資源擁有者，AWS 負責在 John 提交他所撰寫的政策以授予該資源的許可時，驗證他的身分。

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 AWS 請求中的操作。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) 也可以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

申請者是傳送請求至 AWS 服務並要求存取特定資源的人員。申請者傳送請求至 AWS 時主要表示：「您可以允許我對適用 D 的 C 執行 B 嗎？」

評估

評估是 AWS 服務用來根據適用的政策判定進來的請求是否應該予以拒絕或允許。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

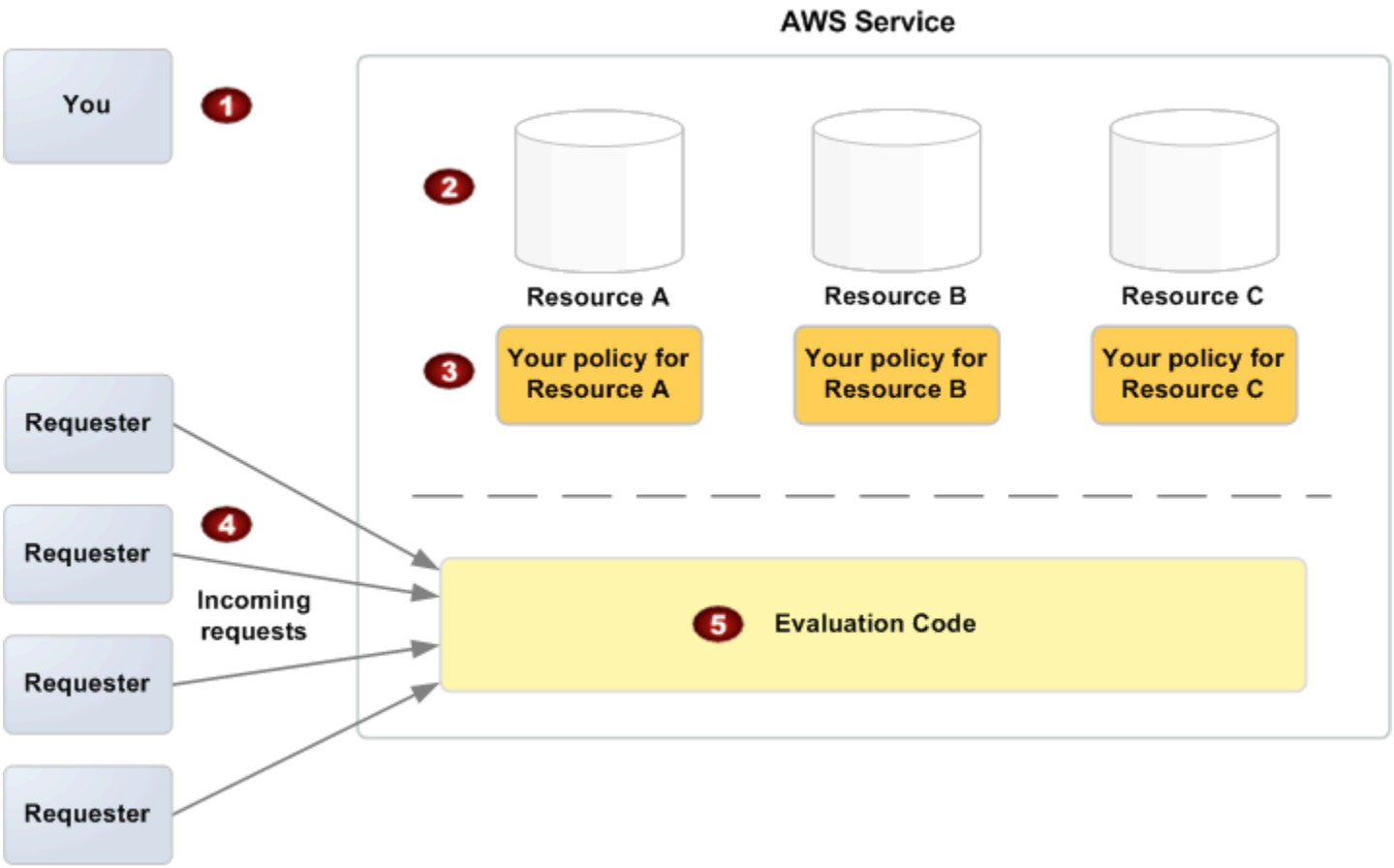
允許源自於具有效用=允許的陳述式，假設所有陳述的條件都符合。範例：如果請求在 2010 年 4 月 30 日下午 1:00 前收到即允許。允許覆寫所有預設拒絕，但是明確拒絕則無法覆寫。

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。

架構概觀

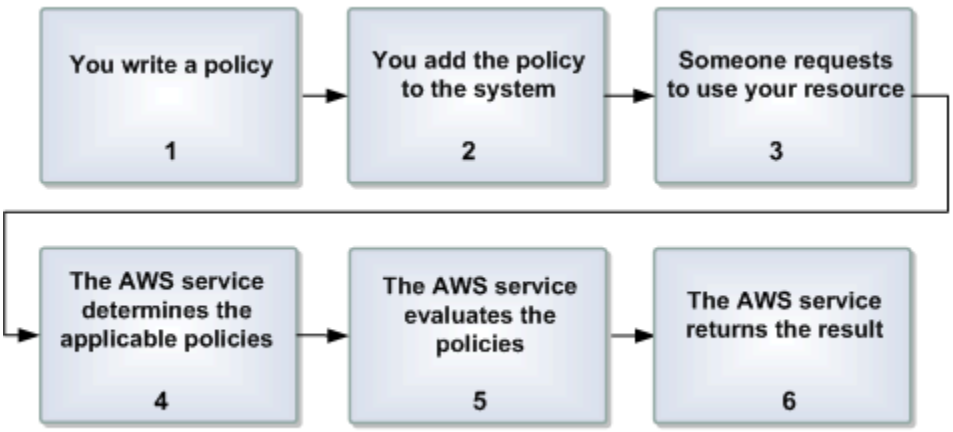
下圖和表格說明與之互動的主要元件，以提供您的資源的存取控制。



1	您是資源擁有者。
2	包含在 AWS 服務內的您的資源 (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 服務本身提供您用來上傳和管理您的政策的 API。
4	申請者及其傳入的 AWS 服務請求。
5	存取原則語言評估代碼。 這是 AWS 服務內的代碼集，根據適用的政策評估傳入的請求並判定是否允許申請者存取資源。如需服務如何做出決定的資訊，請參閱 評估邏輯。

使用存取原則語言

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	您上傳您的政策到 AWS。 AWS 服務本身提供您用來上傳您的政策的 API。例如，您使用 Amazon SNS `SetTopicAttributes` 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 服務拒絕請求或繼續處理它。

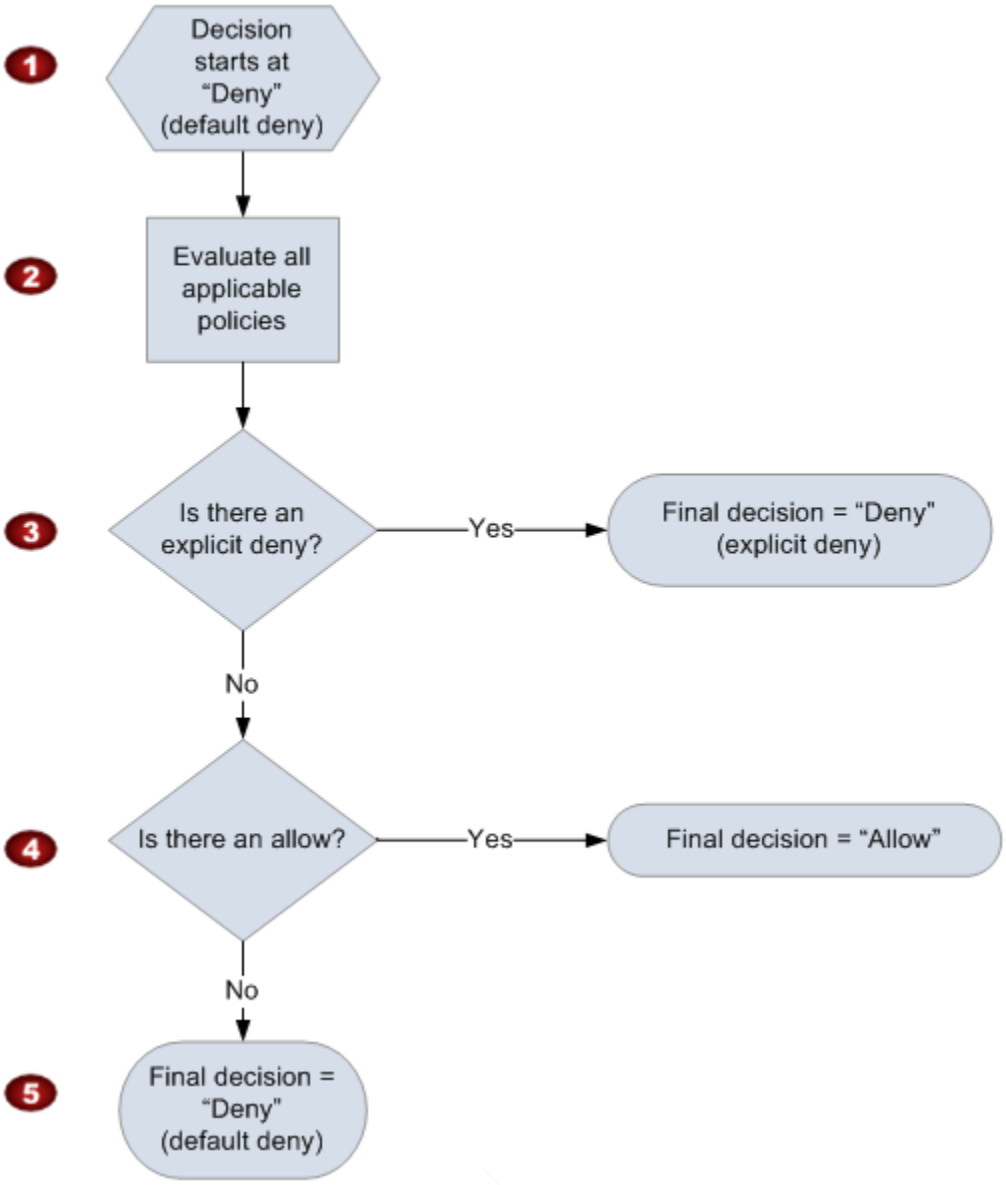
例如，根據政策評估結果，服務傳回「存取遭拒」錯誤給申請者或繼續處理請求。

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	然後強制執行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)。 強制執行程式碼評估政策的順序並不重要。
3	在所有那些政策中，強制執行程式碼會尋找適用於請求的明確拒絕。 如果找到即使一個，強制執行程式碼也會傳回「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)。

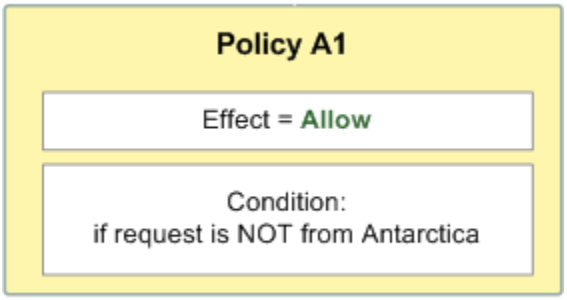
4	如果找不到明確拒絕，強制執行程式碼會尋找適用於請求的任何「允許」指示。 如果找到即使一個，強制執行程式碼也會傳回「允許」的決定，然後完成程序 (服務繼續處理請求)。
5	如果找不到允許，則最後決定是「拒絕」(因為沒有明確拒絕或允許，這會被視為預設拒絕 (如需詳細資訊，請參閱 預設拒絕))。

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，若使用者請求使用 Amazon SNS，但是主題的政策完全不參照使用者的 AWS 帳戶 帳戶，則該政策會導致預設拒絕。

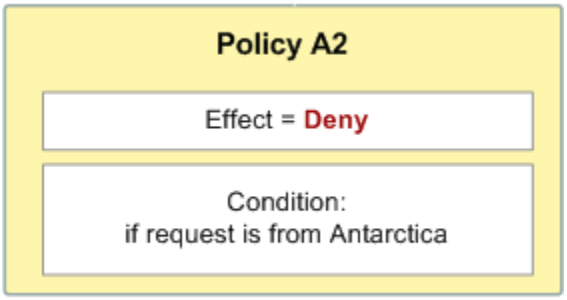
如果未符合陳述式中的條件，政策也會導致預設拒絕。如果符合陳述式中的所有條件，則政策會根據政策中的效用元素的值導致允許或明確拒絕。如果未符合條件，政策不會指定做什麼，所以該狀況下的預設結果是預設拒絕。

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。



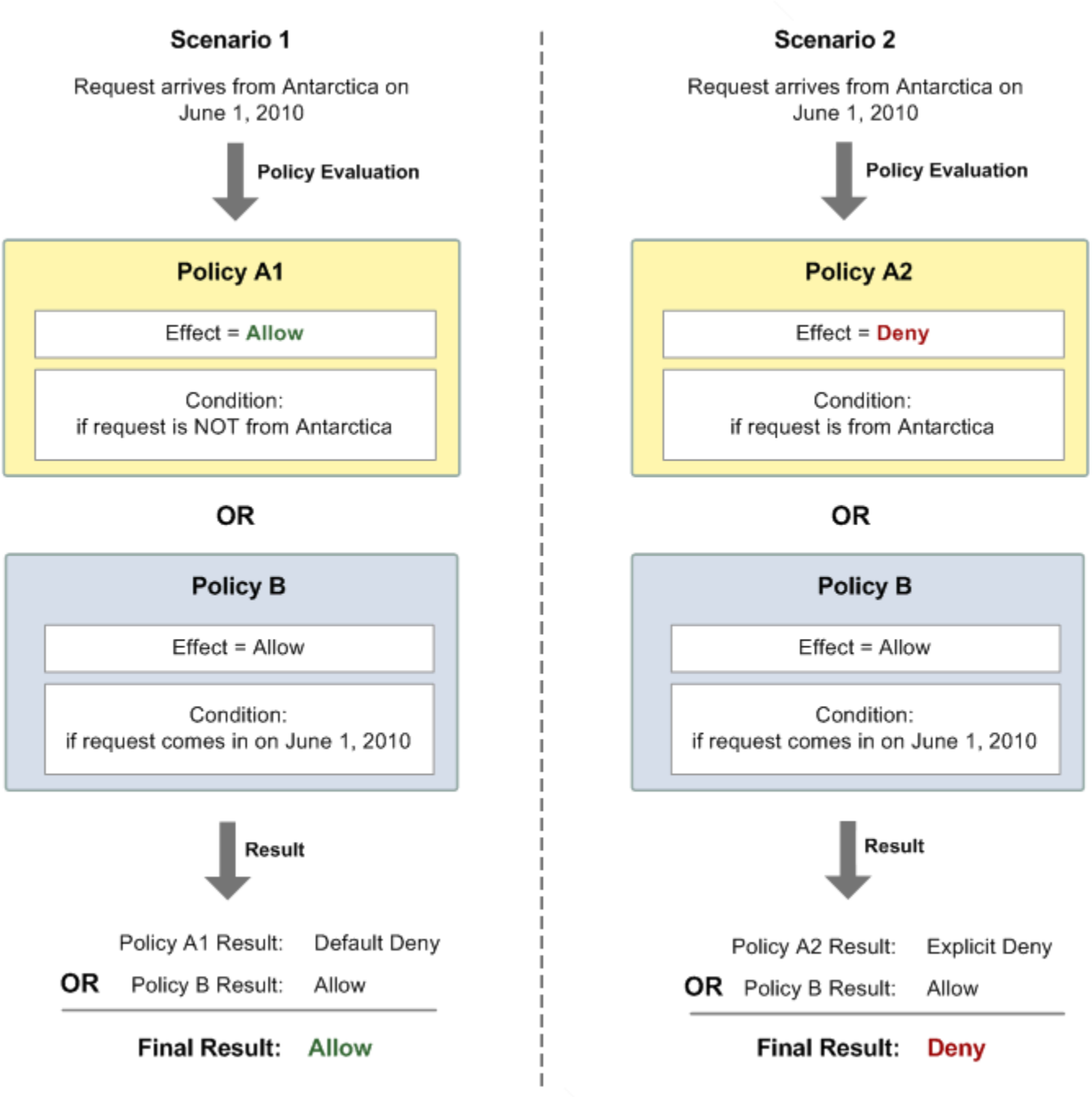
如果某人從美國傳送請求，便符合條件 (請求不是來自南極)。因此，允許該請求。但是，若某人從南極傳送請求，便不符合條件，於是請求的結果是預設拒絕。

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。

主題

授予 AWS 帳戶 存取主題

假設您在 Amazon SNS 系統中有一主題。在最簡單的案例中，您希望允許一或多個 AWS 帳戶 帳戶存取特定主題動作 (例如發佈)。

您可以使用 Amazon SNS API 動動作 `AddPermission` 執行這項作業。該動作會接受一個主題、一份 AWS 帳戶 帳戶 ID 清單、一份動作清單，以及一個標籤，並會在主題的存取控制政策中自動建立新的陳述式。在此案例中，您不用自己撰寫政策，因為 Amazon SNS 會自動為您產生新的政策陳述式。您可以透過呼叫 `RemovePermission` 及其標籤稍後將政策陳述式移除。

例如，如果您呼叫`AddPermission`主題 arn: aw: sn: US-東-2:444455556666:，AWS 帳戶識別碼為 1111-2222-3333、動作和標籤MyTopic，Amazon SNS 會產生並插入下列存取控制政策陳述式：`Publishgrant-1234-publish`

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
```

```
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

一旦加入此陳述式，具有 AWS 帳戶 1111-2222-3333 的使用者即可發佈訊息到主題。

限制訂閱到 HTTPS

在下列範例中，您將通知傳遞通訊協定限定於 HTTPS。

您需要知道如何為該主題撰寫您自己的政策，因為 Amazon SNS `AddPermission` 動作會讓您在授予某人對您主題的存取權時，無法指定協定限制。在此狀況下，您撰寫您自己的政策，然後使用 `SetTopicAttributes` 動作設定主題的 `Policy` 屬性到您的新政策。

下列完整政策的範例授予 AWS 帳戶 帳戶 ID 1111-2222-3333 從主題訂閱通知的能力。

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

發佈訊息至 Amazon SQS 佇列

在此使用案例中，您想要從您的主題發佈訊息至您的 Amazon SQS 佇列。如同 Amazon SNS，Amazon SQS 使用 Amazon 的存取控制政策語言。若要允許 Amazon SNS 傳送訊息，您需要使用 Amazon SQS 動作 `SetQueueAttributes` 對佇列設定政策。

此外，您還需要知道如何撰寫您自己的政策，因為 Amazon SQS `AddPermission` 動作不會建立具有條件的政策陳述式。

> **Note**
>
> 下面範例展示的是 Amazon SQS 政策 (控制對您佇列的存取)，而不是 Amazon SNS 政策 (控制對您主題的存取)。動作是 Amazon SQS 動作，資源是佇列的 Amazon 資源名稱 (ARN)。您可以透過使用 `QueueArn` 動作擷取佇列的 `GetQueueAttributes` 屬性來決定佇列的 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，根據將傳送至佇列之訊息的來源，來限制對於佇列的存取。您可以使用此類型的政策，唯有當訊息是來自您自己的主題之一時，才允許 Amazon SNS 傳送訊息至您的佇列。在這種情況下，您可以指定您的主題中的一個特定主題，其 ARN 是 ARN：AW：SNS：美國東部-2：444455556666：。MyTopic

前述政策是您可撰寫和新增特定佇列的 Amazon SQS 政策範例。它會授予對 Amazon SNS 和其他 AWS 服務的存取。Amazon SNS 提供預設政策給所有新建立的主題。預設政策可讓所有其他 AWS 服務存取您的主題。此預設政策使用 `aws:SourceArn` 條件，來確保 AWS 服務僅代表您自己的 AWS 資源存取您的主題。

允許 Amazon S3 事件通知發佈至主題

在此案例中，您想要設定主題的政策，以便其他 AWS 帳戶 帳戶的 Amazon S3 儲存貯體可發佈至您的主題。如需關於 Amazon S3 發佈通知的詳細資訊，請前往設定儲存貯體事件的通知。

此範例假設您撰寫您自己的政策，然後使用 `SetTopicAttributes` 動作設定主題的 `Policy` 屬性到您的新政策。

下列範例陳述式會使用 SourceAccount 條件來確保只有 Amazon S3 擁有者帳戶可以存取主題。在此範例中，主題擁有者是 111122223333，而 Amazon S3 擁有者是 444455556666。這個範例指出，任何由 444455556666 擁有的 Amazon S3 儲存貯體都可以發佈到。 MyTopic

```
{
  "Statement": [{
    "Effect": "Allow",
     "Principal": {
      "Service": "s3.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      }
    }
  }]
}
```

將事件發佈到 Amazon SNS 時，下列服務支援 aws:SourceAccount：

- Amazon API Gateway
- Amazon CloudWatch
- Amazon DevOps 大師
- Amazon ElastiCache
- Amazon GameLift
- 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

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 服務發佈至另一個 AWS 帳戶 擁有的主題。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 大師
- Amazon ElastiCache
- Amazon GameLift
- 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。在支援時，該值將是 12 位數 AWS 帳戶 ID 代表服務發佈資料。有些服務支持一個，有些支持另一個。

- 請參閱 允許 Amazon S3 事件通知發佈至主題 了解 Amazon S3 通知如何使用 `aws:SourceAccount` 和支援該條件的 AWS 服務列表。
- 請參閱 允許 Amazon SES 發佈到其他帳戶擁有的主題 了解 Amazon SES 如何使用 `aws:SourceOwner` 和支援該條件的 AWS 服務列表。

允許 AWS Organizations 中組織的帳戶發佈至不同帳戶中的主題

AWS Organizations 服務可協助集中管理帳單、控制存取和安全，以及在您的 AWS 帳戶 帳戶間共享資源。

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，組織 `myOrgId` 中的任何 AWS 帳戶 帳戶都可以發佈至帳戶 `444455556666` 中的 Amazon SNS 主題 `MyTopic`。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 警示發佈至不同帳戶中的主題

在此情況下，帳戶中的任何 CloudWatch 警示111122223333都可以發佈到帳戶中的 Amazon SNS 主題444455556666。

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

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"
      }
    }
  }]
}
```

如何搭配使用 Amazon Simple Notification Service 與 IAM

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

如要全面瞭解 Amazon SNS 和其他 AWS 服務如何與大多數的 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的 AWS 服務。

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哪些內容。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。政策動作的名稱通常會和相關聯的 AWS API 操作相同。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些操作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與執行相關聯操作的許可。

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
```

如需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哪些內容。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 (例如列出作業)，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索引鍵	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。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。若您為單一條件索引鍵指定多個值，AWS 會使用邏輯 OR 操作評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在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與該 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 支援全域條件索引鍵和服務特定的條件索引鍵。若要查看 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。您可以將標籤附加到 IAM 實體 (使用者或角色)，以及許多 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。

若要根據標籤控制存取，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索引鍵，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索引鍵，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱《IAM 使用者指南》中的什麼是 ABAC?。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

您使用臨時憑證進行登入時，某些 AWS 服務 無法運作。如需詳細資訊，包括那些 AWS 服務 搭配臨時憑證運作，請參閱《IAM 使用者指南》中的可搭配 IAM 運作的 AWS 服務。

如果您使用使用者名稱和密碼之外的任何方法登入 AWS Management Console，則您正在使用臨時憑證。例如，當您使用公司的單一登入(SSO)連結存取 AWS 時，該程序會自動建立臨時憑證。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 使用者指南中的切換至角色 (主控台)。

您可使用 AWS CLI 或 AWS API，手動建立臨時憑證。接著，您可以使用這些臨時憑證來存取 AWS。AWS 建議您動態產生臨時憑證，而非使用長期存取金鑰。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 AWS 中執行動作時，您會被視為主體。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 使用主體的許可呼叫 AWS 服務，搭配請求 AWS 服務 以向下游服務發出請求。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之後，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》。

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給 AWS 服務 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

服務連結角色是一種連結到 AWS 服務 的服務角色類型。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 AWS 帳戶 中，並由該服務所擁有。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 執行任務。若要授與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。

若要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。

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。

主題

- 政策最佳實務
- 使用 Amazon SNS 主控台
- 其他政策類型

- 多種政策類型
- 允許使用者檢視他們自己的許可

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 受管政策並朝向最低權限許可的目標邁進：如需開始授予許可給使用者和工作負載，請使用 AWS 受管政策，這些政策會授予許可給許多常用案例。它們可在您的 AWS 帳戶 中使用。我們建議您定義特定於使用案例的 AWS 客戶管理政策，以便進一步減少許可。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。
- 套用最低許可許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的權限。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。您也可以使用條件來授予對服務動作的存取權，前提是透過特定 AWS 服務 (例如 AWS CloudFormation)使用條件。如需更多資訊，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，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多重要素驗證 (MFA)：如果存在需要 AWS 帳戶中 IAM 使用者或根使用者的情況，請開啟 MFA 提供額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

有關 IAM 中最佳實務的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 最佳安全實務。

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您 AWS 帳戶 中 Amazon SNS 資源的詳細資訊。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，您不需要允許其最基本主控台許可。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。

為確保使用者和角色仍可使用 Amazon SNS 主控台，還請將 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 支援其他較少見的政策類型。這些政策類型可設定較常見政策類型授與您的最大許可。

- 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。如需許可範圍的更多相關資訊，請參閱《IAM 使用者指南》中的 IAM 實體許可範圍。
- 服務控制政策 (SCP) – SCP 是 JSON 政策，可指定 AWS Organizations 中組織或組織單位 (OU) 的最大許可。AWS Organizations 服務可用來分組和集中管理您企業所擁有的多個 AWS 帳戶。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 會限制成員帳戶中的實體許可，包括每個 AWS 帳戶 的根使用者。如需 Organizations 和 SCP 的詳細資訊，請參閱《AWS Organizations 使用者指南》中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。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。如需更多資訊，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。如需瞭解 AWS 在涉及多種政策類型時如何判斷是否允許一項請求，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 使用者檢視附加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，或是使用 AWS CLI 或 AWS API 透過編寫程式的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

身分型政策是可以連接到身分 (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。若要瞭解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

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，因為這會套用至附加的使用者或角色。如要瞭解您在 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。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。主體可以包括帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

若要啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作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。當主體和資源在不同的 AWS 帳戶 中時，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 存取資源的許可。其透過將身分型政策附加到實體來授予許可。不過，如果資源型政策會為相同帳戶中的主體授與存取，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 角色與資源型政策有何差異。

使用以身分為基礎的政策搭配 Amazon SNS

主題

- IAM 和 Amazon SNS 政策一起搭配
- Amazon SNS 資源 ARN 格式
- Amazon SNS API 動作
- Amazon SNS 政策金鑰
- 用於 Amazon SNS 的政策範例

Amazon Simple Notification Service 已與 AWS Identity and Access Management (IAM) 整合，如此您便可指定使用者在您的 AWS 帳戶 中可以利用 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。

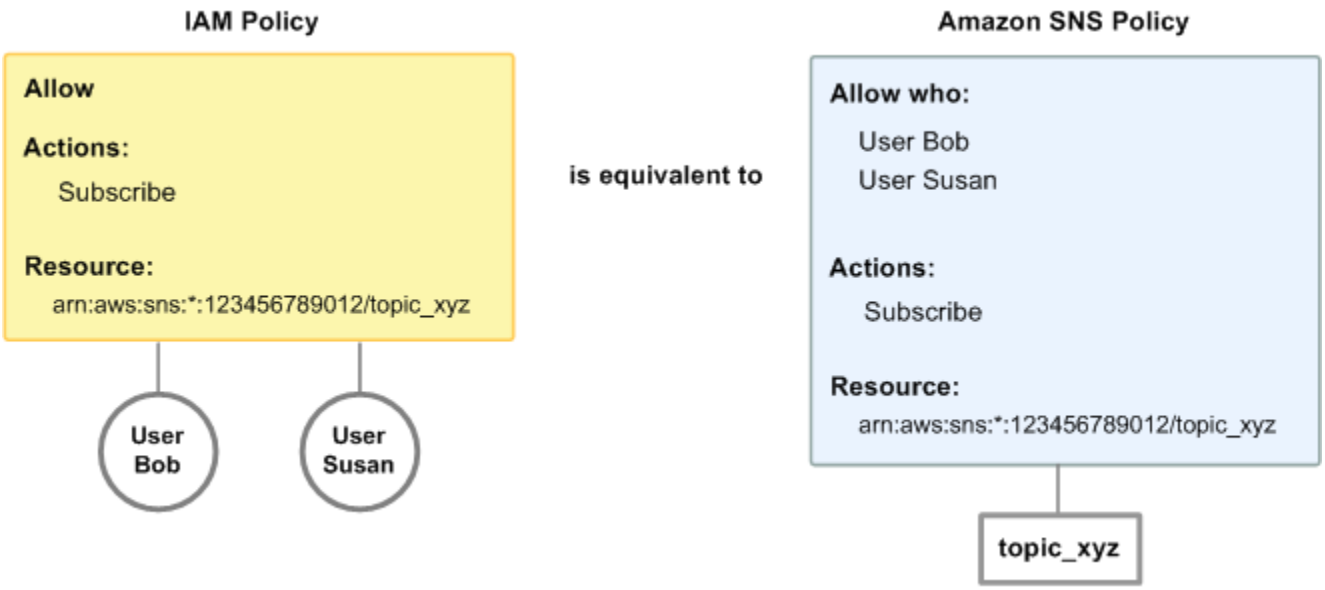
如需涵蓋 Amazon SNS 動作和資源之政策的詳細資訊，請參閱 用於 Amazon SNS 的政策範例。

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 政策僅可限制對 AWS 帳戶內使用者的存取，而非對其他 AWS 帳戶 的存取。

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 政策可授予對其他 AWS 帳戶 或者您自己的 AWS 帳戶 中使用者的存取。

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 政策允許對您 AWS 帳戶內稱為 topic_xyz 的主題使用 Amazon SNS `Subscribe` 動作。IAM 政策會附加到使用者 Bob 和 Susan (表示 Bob 和 Susan 擁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`。



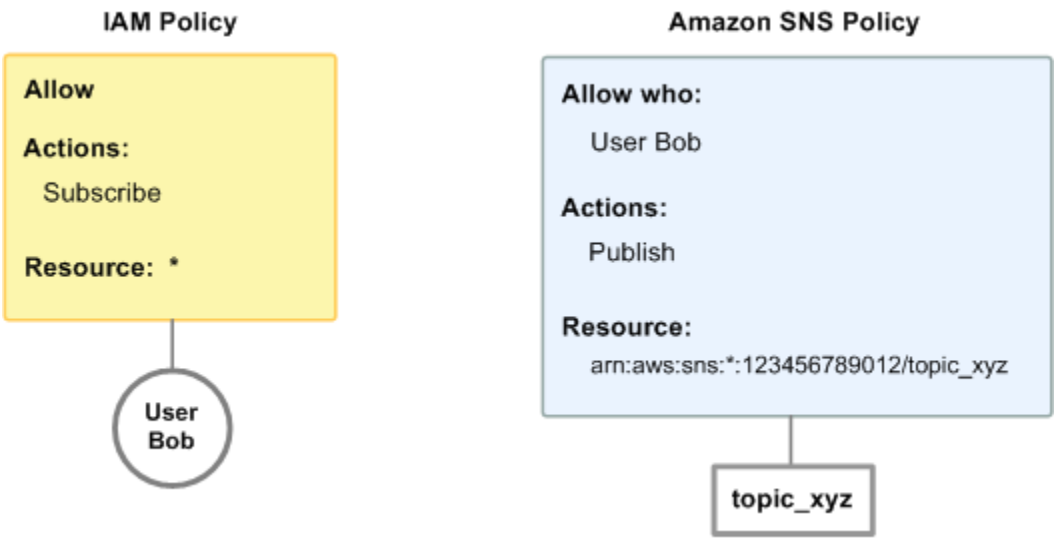
> **Note**
>
> 前面的範例顯示沒有條件的簡易政策。您可以在任一政策中指定特定條件，並取得相同結果。

AWS IAM 和 Amazon SNS 政策間有一點不同：Amazon SNS 政策系統可讓您授予許可給其他 AWS 帳戶，而 IAM 政策無法做到這一點。

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 政策授予他對於任何 `Subscribe` 之主題的 AWS 帳戶 許可，而 Amazon SNS 政策授予他對特定主題 (topic_xyz) 上使用 `Publish` 的許可。此圖說明了此概念。

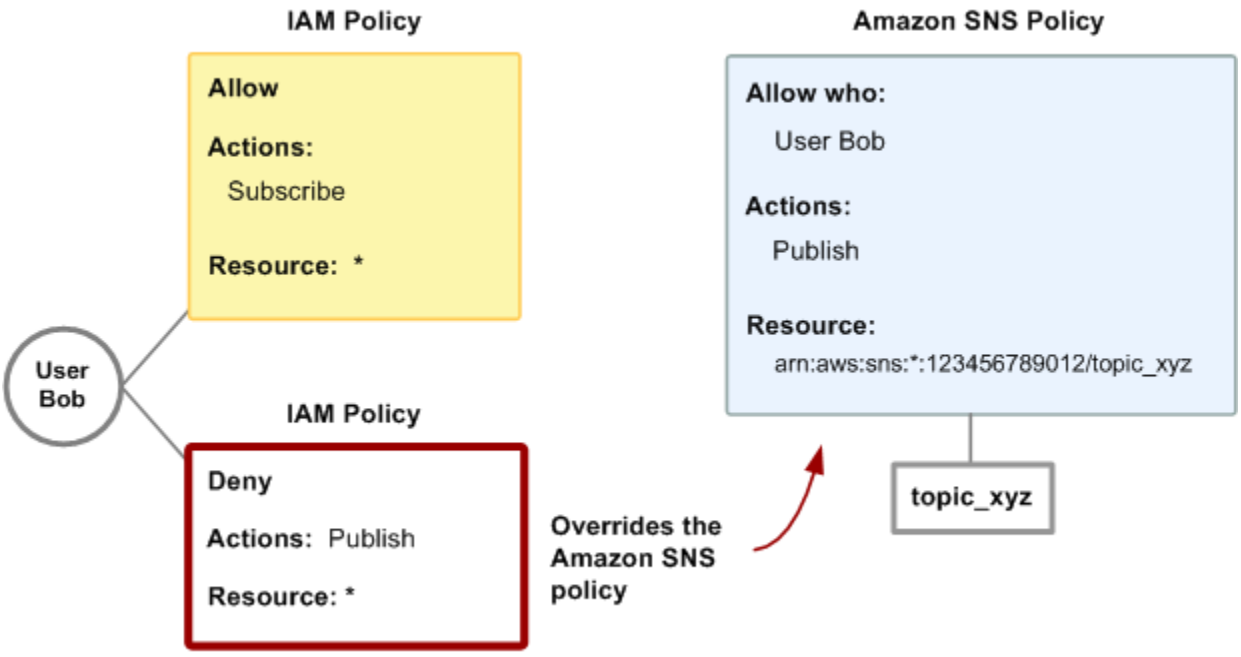


如果 Bob 會傳送訂閱 AWS 帳戶中任何主題的請求，IAM 政策會允許該動作。如果 Bob 會傳送發佈訊息到 topic_xyz 的請求，Amazon SNS 政策會允許該動作。

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。如需關於撰寫 Amazon SNS 政策的詳細資訊，請前往 Amazon SNS 的技術文件。

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

以下是在屬於 123456789012 的美國東部 2 區域 MyTopic 中命名的主題的 ARN。AWS 帳戶

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

Example

如果您 MyTopic 在 Amazon SNS 支援的每個不同區域中都有命名的主題，則可以使用下列 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 會實作下列全 AWS 政策的金鑰加上一些服務特定的金鑰。

如需各項 AWS 服務支援的條件金鑰清單，請參閱 IAM 使用者指南中的適用於 AWS 服務的動作、資源及條件金鑰。如需可在多項 AWS 服務中使用的條件金鑰清單，請參閱 IAM 使用者指南中的 AWS 全域條件內容金鑰。

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: 在 AWS 帳戶 中提供使用者訂閱主題的能力

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。如果您的合作夥伴擁有 AWS 帳戶，可能使用 Amazon SNS 政策會比較容易。不過，合作夥伴的公司中持有 AWS 安全憑證的任何人都可發佈訊息到主題。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。

在這個例子中，我們創建了一 WidgetCo 個名為代表合作夥伴公司的組；我們為需要訪問的合作夥伴公司的特定人員（或應用程序）創建一個用戶；然後我們將用戶放在該組中。

然後，我們會附加一個政策，以授與名為的特定主題的群組Publish存取權WidgetPartnerTopic。

我們也希望防止 WidgetCo 群組針對主題執行任何其他動作，因此我們新增一個陳述式，拒絕任何 Amazon SNS 動作的權限，而非Publish其他主題。 WidgetPartnerTopic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"
    }
  ]
}
```

使用臨時安全登入資料搭配 Amazon SNS

除了以使用者自己的安全登入資料建立 IAM 使用者之外，IAM 也可讓您將暫時的安全登入資料授予任何允許此使用者存取您的 AWS 服務與資源的使用者。您可以管理擁有 AWS 帳戶 帳戶的使用者；這些使用者為 IAM 使用者。您也可以管理您系統中沒有 AWS 帳戶 的使用者；這些使用者被稱為聯合身分使用者。此外，「使用者」也可以是您建立用來存取 AWS 資源的應用程式。

您可使用這些暫時性安全登入資料，對 Amazon SNS 提出請求。API 程式庫會使用這些登入資料來運算出必要的簽章值，以驗證您的請求。若使用過期的登入資料傳送請求，Amazon SNS 會拒絕該請求。

如需暫時安全登入資料之支援的詳細資訊，請前往使用 IAM 中的授予暫時存取權給您的 AWS 資源。

Example 使用暫時的安全登入資料驗證 Amazon SNS 請求

以下範例示範如何取得暫時的安全登入資料，來驗證 Amazon SNS 請求。

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

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。
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 簡單通知服務 API 參考PublishBatch中的發佈和。
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。

動作	描述
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。

主題

- 我未獲授權，不得在 Amazon SNS 中執行動作
- 我沒有授權執行 iam：PassRole
- 我想要允許 AWS 帳戶 外的人員存取我的 Amazon SNS 資源

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。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我沒有授權執行 iam：PassRole

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 允許您傳遞現有的角色至該服務，而無須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，該動作要求服務具備服務角色授與的許可。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。

如需任何協助，請聯絡您的 AWS 管理員。您的管理員提供您的登入憑證。

我想要允許 AWS 帳戶 外的人員存取我的 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的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 如何搭配使用 Amazon Simple Notification Service 與 IAM。
- 如需了解如何存取您擁有的所有 AWS 帳戶 所提供的資源，請參閱《IAM 使用者指南》中的將存取權提供給您所擁有的另一個 AWS 帳戶 中的 IAM 使用者。
- 如需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擁有的 AWS 帳戶。

- 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)。
- 若要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 角色與資源型政策的差異。

在 Amazon SNS 中記錄和監控

本節提供記錄和監控 Amazon SNS 主題的相關資訊。

主題

- 使用 CloudTrail 記錄 Amazon SNS API 呼叫
- 使用 Amazon CloudWatch 監控 Amazon SNS 主題

使用 CloudTrail 記錄 Amazon SNS API 呼叫

Amazon SNS 已與 AWS CloudTrail 整合，這項服務可提供由使用者、角色或 Amazon SNS 中 AWS 服務所採取之動作的記錄。CloudTrail 會擷取 Amazon SNS 的 API 呼叫當作事件。擷取的呼叫包括來自 Amazon SNS 主控台的呼叫，以及對 Amazon SNS API 操作進行的程式碼呼叫。如果您建立追蹤，就可以將 CloudTrail 事件持續交付到 Amazon S3 儲存貯體，包括 Amazon SNS 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的 Event history (事件歷史記錄) 檢視最新事件。您可以利用 CloudTrail 所收集的資訊來判斷向 Amazon SNS 發出的請求，以及發出請求的 IP 地址、人員、時間和其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定及啟用，請參閱 AWS CloudTrail 使用者指南。

CloudTrail 中的 Amazon SNS 資訊

當您建立帳戶時，系統即會在 AWS 帳戶 中啟用 CloudTrail。當 Amazon SNS 發生支援的事件活動時，系統便會將該活動記錄至 CloudTrail 事件，並將其他 AWS 服務事件記錄到事件歷史記錄中。您可以檢視、搜尋和下載 AWS 帳戶 的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷史記錄檢視事件。

若要持續記錄您 AWS 帳戶 帳戶中正在進行的事件 (包括 Amazon SNS 的事件)，請建立追蹤。追蹤能讓 CloudTrail 將日誌檔交付至 Amazon S3 儲存貯體。根據預設，當您在主控台建立線索時，線索會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 Amazon S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

CloudTrail 中的控制平面事件

Amazon SNS 支援將下列 API 動作記錄為 CloudTrail 日誌檔案中的事件：

- 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 動作受到叫用，則不會在 CloudTrail 中記錄這些動作。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。

每一筆事件或日誌項目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：

- 該請求是否透過根或 AWS Identity and Access Management (IAM) 使用者憑證來提出。
- 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。

如需詳細資訊，請參閱 CloudTrail userIdentity 元素。

CloudTrail 中的資料平面事件

若要在 CloudTrail 檔案中啟用下列 API 動作的日誌紀錄，您需要在 CloudTrail 中啟用資料平面 API 活動的日誌紀錄。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。

您可以依資源類型篩選資料平面事件，以精細控制您要在 CloudTrail 中選擇性地記錄和付費的 Amazon SNS API 呼叫。例如，透過指定 `AWS::SNS::Topic` 為資源類型，您可以記錄主題的 `Publish` 和 `PublishBatch` API 動作的呼叫。同樣，透過指定 `AWS::SNS::PlatformEndpoint` 為資源類型，您可以記錄對平台端點的發布 API 動作的呼叫。如需詳細資訊，請參閱《AWS CloudTrail API 參考》中的 `AdvancedEventSelector`。

> **Note**
>
> CloudTrail 不會記錄 Amazon SNS 資源類型 `AWS::SNS::PhoneNumber`。

Amazon SNS 資料平面 API

- `Publish`
- `PublishBatch`

範例：Amazon SNS 日誌檔案項目

追蹤是一種組態，能讓事件以日誌檔案的形式交付到您指定的 Amazon S3 儲存貯體。CloudTrail 日誌檔案包含一或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

以下範例顯示的是展示 `ListTopics`、`CreateTopic` 及 `DeleteTopic` 動作的 CloudTrail 日誌項目。

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

以下範例顯示的是示範 Publish 和 PublishBatch 動作的 CloudTrail 日誌項目。

發布

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
```

```
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T16:44:05Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T16:48:37Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "Publish",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "message": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "subject": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageStructure": "json",
  "messageAttributes": "HIDDEN_DUE_TO_SECURITY_REASONS"
 },
 "responseElements": {
  "messageId": "0787cd1e-d92b-521c-a8b4-90434e8ef840"
 },
 "requestID": "0a8ab208-11bf-5e01-bd2d-ef55861b545d",
 "eventID": "bb3496d4-5252-4660-9c28-3c6aebdb21c0",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
```

```
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

PublishBatch

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T19:20:49Z",
    "mfaAuthenticated": "false"
   }
  }
 },
 "eventTime": "2023-08-21T19:22:01Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "PublishBatch",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "publishBatchRequestEntries": [{
    "id": "1",
    "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
   },
   {
```

```
    "id": "2",
    "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
   }
  ]
 },
 "responseElements": {
  "successful": [{
    "id": "1",
    "messageId": "30d68101-a64a-5573-9e10-dc5c1dd3af2f"
   },
   {
    "id": "2",
    "messageId": "c0aa0c5c-561d-5455-b6c4-5101ed84de09"
   }
  ],
  "failed": []
 },
 "requestID": "e2cdf7f3-1b35-58ad-ac9e-aaaea0ace2f1",
 "eventID": "10da9a14-0154-4ab6-b3a5-1825b229a7ed",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

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 行動推送通知。

> **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。
5. 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

1. 登入 AWS Management Console 並開啟位於 https://console.aws.amazon.com/cloudwatch/ 的 CloudWatch 主控台。
2. 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。
3. 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)。
4. 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)。
5. 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。

6. 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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：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 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

命名空間	指標	描述
		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。如需更多詳細資訊，請參閱 Amazon SNS 訊息傳遞重試。 單位：計數 有效維度：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：總和、平均
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：CountValid 維度：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：總和、平均

命名空間	指標	描述
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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：總和、平均
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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：總和、平均

命名空間	指標	描述
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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：總和、平均
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：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：總和、平均
AWS/SNS	PublishSize	已發佈訊息的大小。 單位：位元組 有效維度：應用程式、PhoneNumber、平台和 TopicName 有效的統計資訊：下限、上限、平均和計數

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 設定簡訊喜好設定。 單位：美元 有效維度：PhoneNumber 有效的統計資訊：上限
AWS/SNS	SMSSuccessRate	成功傳送簡訊的比率。 單位：計數 有效維度：PhoneNumber 有效的統計資訊：總和、平均、資料樣本

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	• 發佈至您 AWS 帳戶中 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 帳戶中待處理訂閱的大致數量。 • 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 帳戶中選取之 Amazon SNS API 的 API 呼叫次數。 • 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
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

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

第三方稽核人員評估 Amazon SNS 安全與合規性是多個 AWS 合規計畫中的一環，而評估內容包括美國健康保險流通與責任法案 (HIPAA)。

如需特定合規計畫的 AWS 服務範圍清單，請參閱合規計畫的 AWS 服務範圍。如需一般資訊，請參閱 AWS 合規計劃。

您可使用 AWS Artifact 下載第三方稽核報告。如需詳細資訊，請參閱 AWS Artifact 中的下載報告。

您使用 Amazon SNS 時的合規責任取決於資料的敏感度、您的公司的合規目標，以及適用的法律和法規。AWS 提供以下資源協助您處理合規事宜：

- 安全與合規快速入門指南 – 這些部署指南就在 AWS 上部署以安全及合規為重心之基準環境，討論架構考量並提供相關步驟。
- HIPAA 安全與合規架構白皮書：本白皮書說明公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源：這組手冊和指南可能適用於您的產業和位置。
- AWS Config 開發人員指南中的使用規則評估資源：AWS Config 服務可評估資源組態對於內部實務、業界準則和法規的合規狀態。
- AWS Security Hub – 此 AWS 服務可供您檢視 AWS 中的安全狀態，可助您檢查是否符合安全產業標準和最佳實務。

Amazon SNS 的恢復能力

利用圍繞可用區域的 AWS 全球基礎設施來確保 Amazon SNS 的彈 AWS 區域 性。 AWS 區域 透過低延遲、高輸送量和高冗餘網路連接，提供實體分離和隔離的可用區域。這種架構允許在可用區域之間進行無縫容錯移轉，而不會中斷，與傳統的資料中心基礎架構相比，應用程式和資料庫本質上具有更高的容錯能力和擴充性 透過使用可用區域，Amazon SNS 訂閱者將受益於增強的可用性和可靠性，並確保在可能發生中斷的情況下傳遞訊息。如需 AWS 區域 和可用區域的詳細資訊，請參閱AWS 全域基礎結構。

此外，Amazon SNS 主題的訂閱可以設定交付重試次數和無效字母佇列，以自動處理暫時性故障，並確保訊息可靠地到達預定目的地。

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，提升應用程式的整體穩定性。

Amazon SNS 的基礎設施安全性

作為受管服務，Amazon SNS 受到安 AWS 全、身分和合規最佳實務說明文件中所提供的全球網路安全程序的保護。

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。這樣可以有效地將對指定 Amazon SNS 主題的網路存取從網路中的特定 VPC 隔離出來 AWS 。如需更多詳細資訊，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 存取權的應用程式和 AWS 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夠讀取或寫入您的 Amazon SNS 主題，否則您應確保您的主題無法公開存取 (全世界所有人或任何已驗證的 AWS 使用者都能存取)。

- 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 存取權的應用程式和 AWS 服務使用 IAM 角色

若為要存取 Amazon SNS 主題的應用程式或 AWS 服務 (例如 Amazon EC2)，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，因此不得將 AWS 登入資料直接儲存在應用程式或 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，該角色提供應用程式在呼叫其他 AWS 資源時可以使用的臨時許可。

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 會使用 AWS Key Management Service 中管理的金鑰。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 靜態加密 及 金鑰管理。

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，您無法在發佈至加密的 SNS 主題時使用 HTTP。

AWS 建議您使用 HTTPS 而不是 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。

Amazon SNS 主題進行故障診斷

本節提供針對 Amazon SNS 進行故障診斷主題的相關資訊。

Amazon SNS 主題使用 AWS X-Ray 進行故障診斷

AWS X-Ray 收集有關您的應用程式服務所請求的資料，並讓您檢視和篩選資料以識別潛在問題和進行優化的機會。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 以檢視 Amazon SNS 和應用程式所使用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

當使用者請求透過 Amazon SNS 主題傳送AWS X-Ray至您的 Amazon 資料 Firehose、Amazon SQS 和 HTTP/ S 端點訂閱時 AWS Lambda，您可以用來追蹤和分析使用者請求。由於 X-Ray 可讓您 end-to-end 檢視整個請求，因此您可以檢視呼叫 Amazon SNS 主題的內容，以及主題訂閱的下游項目。您可以分析訊息及其後端服務中的延遲情況 (例如，請求在某個主題中花費的時間，以及將訊息傳遞至每個主題的訂閱所花費的時間)。

> **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相關資訊，請參閱《AWS 一般參考》中的 X-Ray service quotas。

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。

主題

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 及 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。

在 Amazon SNS 啟用主動追蹤 (主控台)

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)。

為 Amazon SNS 主題啟用 X-Ray 後，您可以使用 X-Ray 服務對應來檢視主題的 end-to-end 追蹤和服務對應。

在 Amazon SNS 主題 (AWS SDK) 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 SDK 以啟用 Amazon SNS 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());
        }
    }
```

在 Amazon SNS 主題 (AWS CLI) 啟用主動追蹤

下列程式碼範例示範如何使用 AWS CLI 以啟用 Amazon SNS 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
```

在 Amazon SNS 主題 (AWS CloudFormation) 啟用主動追蹤

以下 AWS CloudFormation 堆疊展示如何在 Amazon SNS 主題啟用主動追蹤。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
```

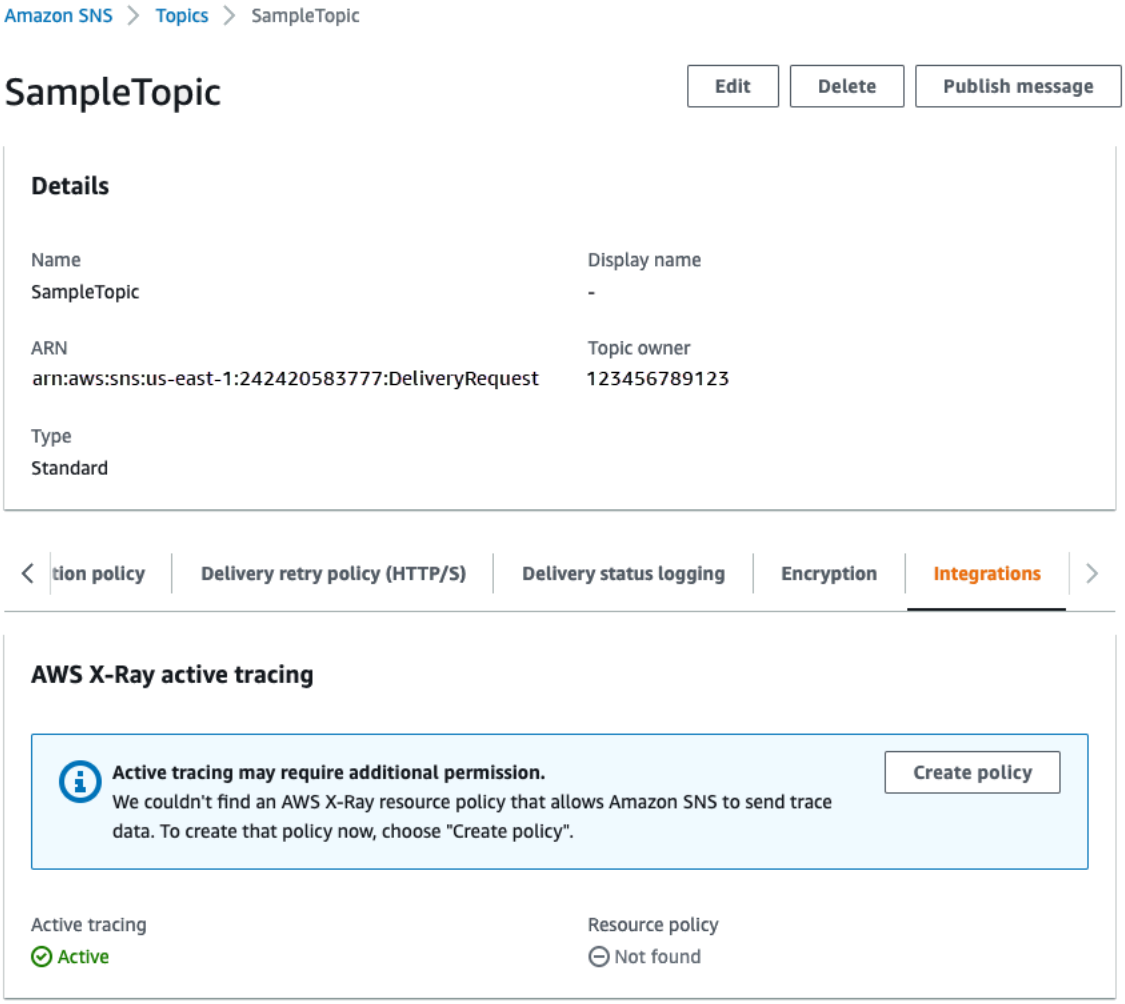
```
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

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。



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，請參閱若要使用建立主題 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 服務對應來檢視主題的 end-to-end 追蹤和服務對應。


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
Client
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
77ms
0.2 t/min
xray-lambda
AWS::Lambda
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

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的 AWS 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。如需有關服務功能的區域可用性，以及訂閱更新相關通知的資訊，請參閱有什麼新功能 AWS？ 。

變更	描述	日期
加拿大西部（卡爾加里）支持 FIFO 主題	Amazon SNS 在加拿大西部（卡爾加里）支持 FIFO 主題。	2024年3月28日
在五個新區域提供 Amazon SNS 簡訊支援	Amazon SNS 在下列區域新增簡訊支援：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
火力地堡雲消息傳遞（FCM）HTTP V1 支持	Amazon SNS 支持 FCM v1 憑據。	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 支援設DeliveryStatusLogging 定 Amazon SNS 主題	AWS CloudFormation 支援在建立或更新 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	2023 年 10 月 26 日

	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。	
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。	之前僅支援 Amazon SNS 標準主題， AWS X-Ray 現在會在使用者請求中透過 FIFO 主題傳送至 Amazon 資料 Firehose、Amazon SQS 和 HTTP/S 端點訂閱時 AWS Lambda，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 在使用者請求透過 Amazon SNS 標準主題傳送至您的 Amazon 資料 Firehose、Amazon SQS 和 HTTP/S 端點訂閱時 AWS Lambda，追蹤和分析這些請求。	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 支援下列區域的 SMS 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	訊息資料保護使用資料保護政策稽核和封鎖在應用程式或 AWS 服務之間移動的敏感資訊，以保護發佈到 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，根據可附加至 IAM 資源 (例如 IAM 使用者和角色) 的標籤，以及 Amazon SNS 主題等 AWS 資源 (例如 Amazon SNS 主題) 來定義存取許可，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。當您的 AWS 帳戶位於 SMS 沙箱中時，您只能將 SMS 訊息傳送至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。當您的 AWS 帳戶位於 SMS 沙箱中時，您只能將 SMS 訊息傳送至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 person-to-person (P2P) 長碼進行 application-to-person (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
Support 1 分鐘的 Amazon CloudWatch 指標	Amazon SNS 的 1 分鐘 CloudWatch 指標現在可在所有 AWS 區域使用。	2021 年 1 月 28 日
Support Amazon 資料 Firehose 端點	您可以訂閱 Firehose 傳遞串流至 SNS 主題。這可讓您將通知傳送到存檔和分析端點，例如 Amazon 簡單儲存服務 (Amazon S3) 儲存貯體、Amazon Redshift 表格、亞馬遜 OpenSearch 服OpenSearch務 (服務) 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
Support 使用的主題疑難排解 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

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。

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。