使用者指南

AWS Identity and Access Management

AWS Identity and Access Management: 使用者指南

Table of Contents

什麼是 IAM？

Follow us on Twitter

AWS Identity and Access Management (IAM) 是可協助您安全控制 AWS 資源存取的 Web 服務。使用 IAM，您可以集中管理許可，以控制使用者可以存取的 AWS 資源。您可以使用 IAM 來控制能通過身分驗證 (登入) 和授權使用資源的 (具有許可) 的人員。

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需檢視需要您以根使用者身分登入的任務完整清單，請參閱 需要根使用者憑證的任務。

目錄

IAM 影片簡介

AWS 培訓和認證提供了一個 10 分鐘的 IAM 視頻介紹：

簡介 AWS Identity and Access Management

IAM 功能

IAM 為您提供以下功能：

共用存取您的 AWS 帳戶

您可以授與其他人管理和使用 AWS 帳戶資源的許可，而無需共用您的密碼或存取金鑰。

精密許可

您可以對不同的人員授與不同資源的不同許可。例如，您可能允許某些使用者完全存取 Amazon 彈性運算雲端 (Amazon EC2)、亞馬遜簡單儲存服務 (亞馬遜 S3)、亞馬遜 DynamoDB、Amazon Redshift 和其他服務。 AWS 對於其他使用者，您可以只允許對一些 S3 儲存貯體進行唯讀存取，或僅具有管理一些 EC2 執行個體的許可，或存取您的帳單資訊，但不能進行任何其他動作。

為在 Amazon EC2 上執行的應用程式安全存取 AWS 資源

您可以使用 IAM 功能為在 EC2 執行個體上執行的應用程式安全地提供憑證。這些認證為您的應用程式提供存取其他 AWS 資源的權限。範例包括 S3 儲存貯體和 DynamoDB 表格。

多重要素驗證 (MFA)

您可以為帳戶和個別使用者新增雙重要素身分驗證，以提高安全性。使用 MFA，您或您的使用者不僅必須提供密碼或存取金鑰才能使用您的帳戶，還必須提供來自特殊設定裝置的代碼。如果您已將 FIDO 安全性金鑰與其他服務搭配使用，且該金鑰具有 AWS 受支援的組態，則可以使 WebAuthn 用 MFA 安全性。如需詳細資訊，請參閱 使用金鑰和安全金鑰的支援組態。

聯合身分

您可以允許已經在其他地方擁有密碼的使用者（例如，在您的公司網路中或從網際網路身分提供者取得）取得您 AWS 帳戶的暫時存取權。

身分資訊保證

如果您使用 AWS CloudTrail，則會收到日誌記錄，其中包含有關在您的帳戶中請求資源的資訊。該資訊是根據 IAM 身分。

PCI DSS 合規

IAM 支援處理、儲存、傳輸商家或服務供應商的信用卡資料，並且已驗證合規於支付卡產業 (PCI) 資料安全標準 (DSS)。如需 PCI DSS 的詳細資訊，包括如何要求 AWS PCI 相容性 Package 的複本，請參閱 PCI DSS 等級 1。

與許多 AWS 服務集成

如需與 IAM 搭配使用的 AWS 服務清單，請參閱AWS 與 IAM 搭配使用的服務。

最終一致

IAM 與許多其他 AWS 服務一樣，最終是一致的。IAM 可跨 Amazon 全球資料中心內多部伺服器複寫資料，來達到高可用性。如果變更一些資料的請求成功完成，則該變更經認可並安全儲存。然而，該變更必須跨 IAM 複寫，這可能需要一些時間。此類變更包括建立或更新使用者、群組、角色或政策。在應用程式的關鍵、高可用性代碼路徑中，我們不建議進行此類 IAM 變更。而應在不常運作的、單獨的初始化或設定常式中進行 IAM 變更。另外，在生產工作流程套用這些變更之前，請務必確認變更已完成傳播。如需詳細資訊，請參閱 我所做的變更不一定都會立刻生效。

免費使用

AWS Identity and Access Management （IAM）和 AWS Security Token Service （AWS STS）是您 AWS 帳戶的功能，無需額外付費。只有當您使用 IAM 使用者或 AWS STS 臨時安全登入資料存取其他 AWS 服務時，才會向您收費。如需其他 AWS 產品定價的相關資訊，請參閱 Amazon Web Services 定價頁面。

存取 IAM

您可以使用下 AWS Identity and Access Management 列任一方式使用。

AWS Management Console

主控台是用於管理 IAM 和 AWS 資源的瀏覽器介面。如需有關透過主控台存取 IAM 的詳細資訊，請參閱 《AWS 登入 使用者指南》 中的 如何登入 AWS。

AWS 命令行工具

您可以使用 AWS 命令列工具在系統的命令列發出指令，以執行 IAM 和工 AWS 作。使用命令列可以比主控台更快，也更便利。如果您想要建置執行工作的指令碼，指令行 AWS 工具也很有用。

AWS 提供兩組指令行工具：AWS Command Line Interface(AWS CLI) 和 AWS Tools for Windows PowerShell. 若要取得有關安裝和使用的資訊 AWS CLI，請參閱《使AWS Command Line Interface用指南》。若要取得有關安裝和使用 Windows 工具的資訊 PowerShell，請參閱使用AWS Tools for Windows PowerShell 者指南。

登入主控台後，您可以使用 AWS CloudShell 瀏覽器執行 CLI 或 SDK 命令。存取資 AWS 源的權限取決於您用來登入主控台的認證。根據您的經驗，您可能會發現 CLI 是管理 AWS 帳戶的更

有效率的方法。如需更多資訊，請參閱使用 AWS CloudShell 來使用 AWS Identity and Access Management

AWS 開發套件

AWS 提供 SDK（軟件開發工具包），其中包含各種編程語言和平台（Java，Python，紅寶石，.NET，iOS，安卓等）的示例代碼。SDK 提供了一種方便的方式來建立 IAM 和. AWS例如，開發套件會負責的工作諸如以密碼演算法簽署請求、管理錯誤以及自動重試請求。如需 AWS SDK 的相關資訊，包括如何下載和安裝這些軟體開發套件，請參閱 Amazon Web Services 工具頁面。

IAM 查詢 API

您可以使用 IAM 查詢 API 以 AWS 程式設計方式存取 IAM，這可讓您直接向服務發出 HTTPS 請求。使用查詢 API 時，您必須加入程式碼，才能夠使用您的登入資料以數位方式簽署請求。如需詳細資訊，請參閱 使用 HTTP 查詢請求呼叫 IAM API 和 IAM API 參考。

IAM 的使用時機為何？

執行不同工作職能時

AWS Identity and Access Management 是一項核心基礎結構服務，可根據內部身分識別提供存取控制的基礎架構服務 AWS。每次存取 AWS 帳戶時都會使用 IAM。

使用 IAM 的方式會有所不同，具體取決於您在 AWS中所執行的工作。

- 服務使用者：如果使用 AWS 服務執行任務，管理員會為您提供所需的憑證和許可。隨著您為了執行作業而使用更多的進階功能，您可能會需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。
- 服務管理員 — 如果您負責公司的 AWS 資源，則可能擁有 IAM 的完整存取權。您的任務是判斷服務使用者應存取的 IAM 功能及資源。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。
- IAM 管理員：如果您是 IAM 管理員，您可以透過管理 IAM 身分和寫入政策來管理 IAM 存取權。

獲得存取 AWS 資源的授權時

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，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。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 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) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

以 IAM 使用者身分登入時

IAM 使用者是您內部的身分，具 AWS 帳戶 有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。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱 IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

擔任 IAM 角色時

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法更多相關資訊，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 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。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，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取權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。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。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。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，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 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至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。如需更多資訊，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。

若要了解是否要使用 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。

建立政策和許可時

您能夠建立政策，進而將許可授與給使用者。該政策文件中會列出使用者可執行的動作，以及會受到這些動作影響的資源。根據預設，未明確允許的任何動作或資源將被拒絕。可建立政策並連接至主體 (使用者、使用者群組、使用者擔任的角色以及資源)。

這些政策會與 IAM 角色搭配使用：

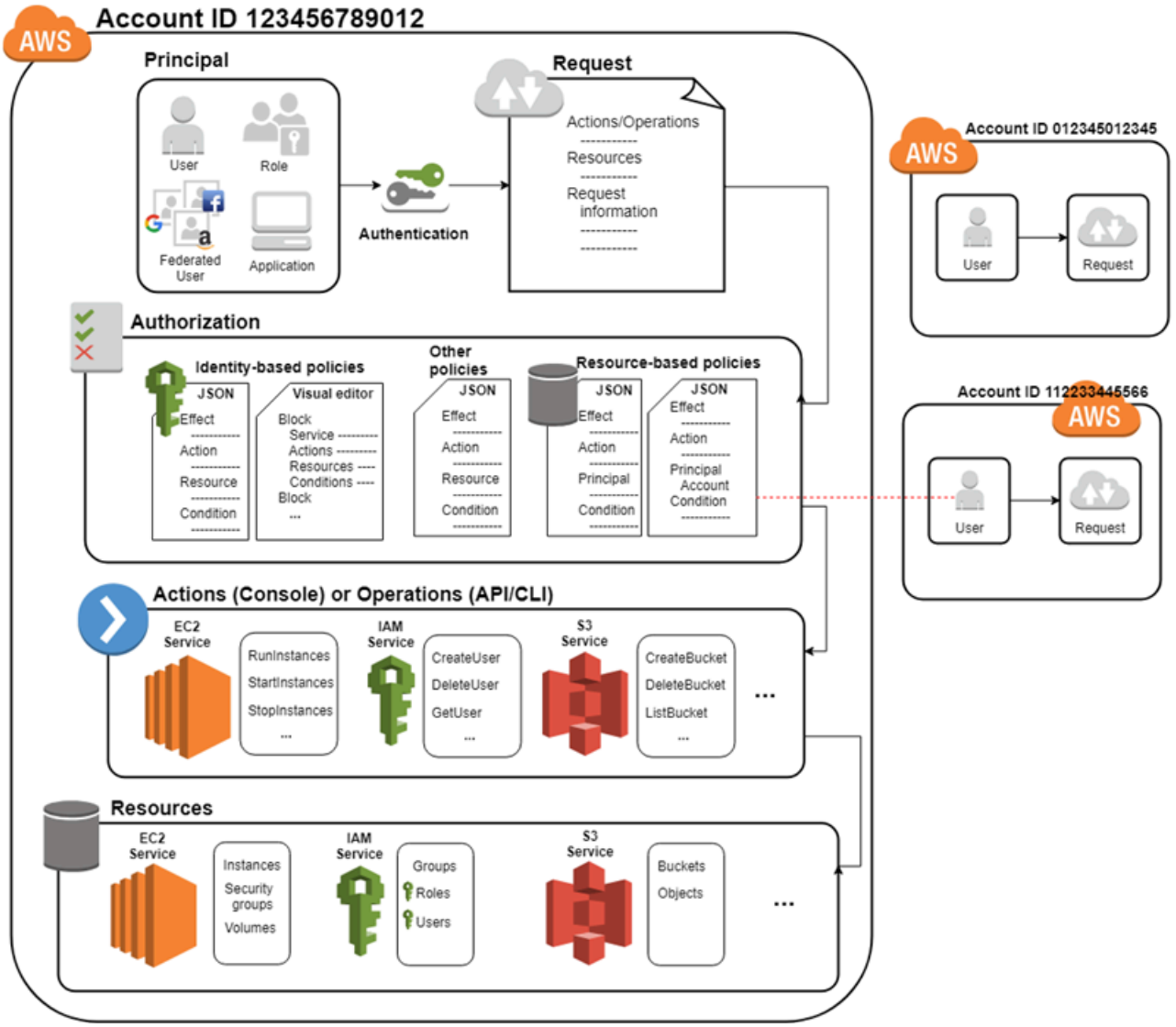
- 信任政策：定義哪些 主體 可以擔任角色 (以及擔任條件)。信任政策是一項專屬於 IAM 角色的資源型政策。一個角色只能由一項信任政策。
- 身分型政策 (內嵌和受管)：這些政策會定義角色使用者能夠執行 (或無法執行) 的許可，以及在哪些資源上執行。

使用 以身分為基礎的 IAM 政策範例 可協助您定義 IAM 身分的許可。在您找到所需的政策後，選擇「查看政策」以查看政策的 JSON。您可以將 JSON 政策文件用作自己政策的範本。

> **Note**
>
> 如果您是使用 IAM Identity Center 來管理使用者，您可以在 IAM Identity Center 指派許可集，而不是將許可政策連接至主體。將權限集指派給群組時 AWS IAM Identity Center 的使用者，或者 IAM Identity Center 會在每個帳戶中建立對應的 IAM 角色，並將權限集中指定的政策附加到這些角色。IAM Identity Center 會負責管理角色，並允許您定義的授權使用者擔任該角色。修改許可集後，IAM Identity Center 會確保有據此更新對應的 IAM 政策和角色。
> 如需有關 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 的運作方式

IAM 提供控制 AWS 帳戶身分驗證和授權所需的基礎設施。下圖會說明 IAM 的基礎設施：



首先，人類使用者或應用程式會使用其登入憑證來向 AWS進行身分驗證。系統會透過將登入憑證與受 AWS 帳戶信任的主體 (IAM 使用者、聯合身分使用者、IAM 角色或應用程式) 進行比對來提供身分驗證。

接下來，系統會提出請求來向主體授與資源的存取權。系統是為了回應授權請求而授與存取權。舉例來說，當您第一次登入主控台且位於主控台首頁頁面時，您並未存取特定服務。選取某項服務時，系統會將授權請求傳送至該服務，該項服務會檢查您的身分是否在授權使用者清單上、強制執行了哪些政策來控制授與的存取層級，以及任何可能生效中的其他政策。授權要求可由您內部的主體 AWS 帳戶 或您信任的其他主 AWS 帳戶 體提出。

獲得授權後，主體即可對您 AWS 帳戶中的資源採取動作或執行操作。例如，主體可以啟動新 Amazon Elastic Compute Cloud 執行個體、修改 IAM 群組成員資格或刪除 Amazon Simple Storage Service 值區。

基本概念

- 條款
- Principal
- 請求
- 身分驗證
- 授權
- 動作或操作
- 資源

條款

使用下列項目時通常使用這些 IAM 術語 AWS：

IAM 資源

IAM 資源儲存在 IAM 中。您可以從 IAM 中新增、編輯和移除這些資源。

- 使用者
- 群組
- role
- 政策
- 身分供應商物件

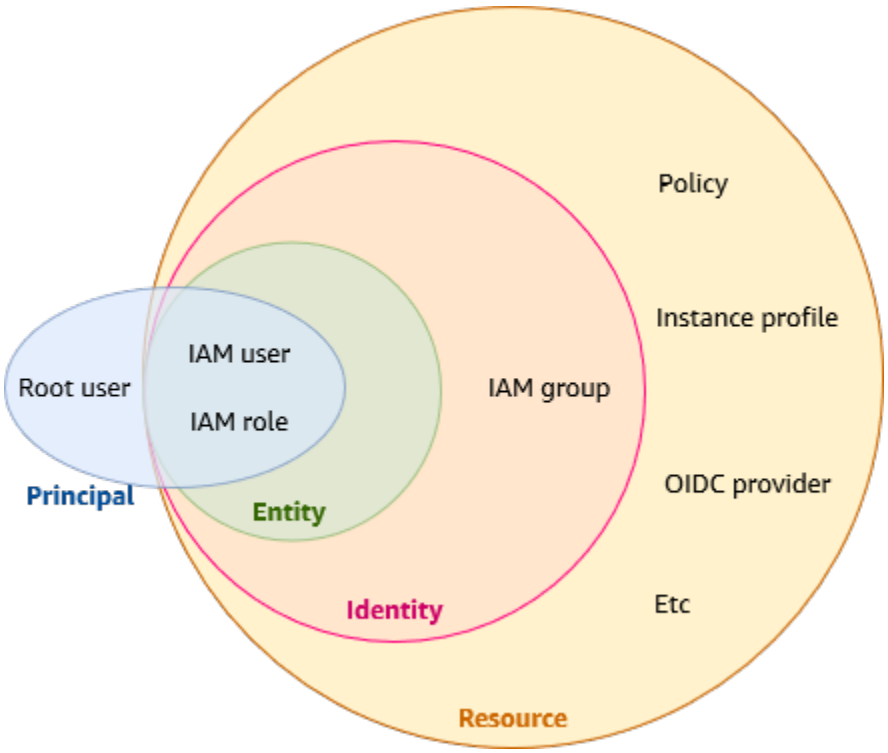
IAM 實體

AWS 用於身份驗證的 IAM 資源。在以資源為基礎的政策中，實體可以指定為主體。

- 使用者
- role

IAM 身分

可在政策中授權以執行動作和存取資源的 IAM 資源。身分包括使用者、群組和角色。



主體

使用 IAM 使用者或 IAM 角色登入並向其提出請求的個人或應用程式 AWS。 AWS 帳戶根使用者主體包括聯合身分使用者及擔任的角色。

人類使用者

也稱為 人類身分；是應用程式的相關人員、管理員、開發人員、操作員和消費者。

工作負載

可提供商業價值的資源和程式碼的集合，例如應用程式或後端程序。可包括應用程式、操作工具和元件。

Principal

主參與者是可以對 AWS 資源的動作或作業請求的人力使用者或工作負載。驗證之後，可以根據主體類型，授與主體要 AWS求的永久或暫時認證來提出要求。IAM 使用者和根使用者被授予永久憑證，而角色則被授予臨時性憑證。最佳做法是，建議您要求人力使用者和工作負載使用臨時登入資料存取 AWS 資源。

請求

當主體嘗試使用 AWS Management Console、 AWS API 或該 AWS CLI主體傳送要求時 AWS。請求包含下列資訊：

- 動作或操作 – 主體想要執行的動作或操作。這可以是中的動作 AWS Management Console，也可以是 AWS CLI 或 AWS API 中的作業。
- 資源 — 執行動作或作業時所依據的 AWS 資源物件。
- 主體 – 使用實體 (使用者或角色) 來傳送請求的人員或應用程式。有關主體的資訊，包括與該主體用於登入之實體相關聯的政策。
- 環境資料 – IP 地址、使用者代理程式、啟用 SSL 的狀態或一天中時間的相關資訊。
- 資源資料 – 與所請求資源相關的資料。這可以包括諸如 DynamoDB 資料表名稱或 Amazon EC2 執行個體上之標籤的資訊。

AWS 將請求信息收集到請求上下文中，該內容用於評估和授權請求。

身分驗證

主體必須使用其認證來驗證 (登入 AWS)，才能將要求傳送至 AWS。某些服務 (例如 Amazon S3 和 AWS STS) 允許來自匿名使用者的一些請求。不過，這些服務是此規則的例外。

若要從主控台通過根使用者的身分驗證，您必須使用您的電子郵件地址和密碼登入。身為聯合身分使用者，您會經由身分提供者驗證，並透過假設 IAM 角色授予對 AWS 資源的存取權。以 IAM 使用者的身分，提供帳戶 ID 或別名，然後您的使用者名稱和密碼。若要從 API 或 AWS CLI對工作負載進行身分驗證，您可能要用指派給角色的臨時性憑證，或者您也可能要透過提供您的存取金鑰和私密金鑰來使用長期憑證。您可能還需要提供額外的安全性資訊。最佳作法是 AWS 建議您使用多重要素驗證 (MFA) 和臨時登入資料來增加帳戶的安全性。若要進一步了解 AWS 可驗證的 IAM 實體，請參閱IAM 使用者和IAM 角色。

授權

您也必須獲得授權 (允許) 以完成您的請求。在授權期間， AWS 使用請求內容的值來檢查套用於請求的政策。接著使用政策以決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中，並指定主體實體的權限。有數種類型的政策會影響是否授權請求。若要為您的使用者提供存取其帳戶中 AWS 資源的權限，您只需要以身分識別為基礎的策略。資源類型政策最常用於授與跨帳戶存取權。其他政策類型是進階功能，應謹慎使用。

AWS 檢查適用於請求內容的每個策略。如果單一權限原則包含拒絕的動作，則會 AWS 拒絕整個要求並停止評估。此稱為明確拒絕。由於預設會拒絕要求，因此只有在適用權限原則允許您要求的每個部分時，才會 AWS 授權您的要求。用於單一帳戶中請求的評估邏輯會遵循下列一般規則：

- 根據預設，所有的請求一律拒絕。(一般而言，一律允許使用帳戶中的資源的 AWS 帳戶根使用者 登入資料提出請求)。
- 任何許可政策中的明確允許 (以身分為基礎或以資源為基礎的政策) 都會覆寫此預設值。
- 如果 Organizations SCP、IAM 許可界限或工作階段政策存在就會覆寫該允許。如果這些政策類型中有一或多個存在，則它們必須全部允許該請求。否則，它會被隱含拒絕。
- 任何政策中的明確拒絕會覆寫任何允許。

若要進一步了解如何評估所有政策類型的詳細資訊，請參閱 政策評估邏輯。如果您需要在不同的帳戶發出請求，在其他帳戶中的政策必須可讓您存取資源，而且您用來發出請求的 IAM 實體必須具備允許該請求的以身分為基礎的政策。

動作或操作

在您的要求經過驗證和授權之後， AWS 核准要求中的動作或作業。操作由服務定義，包括您可以對資源執行的動作，例如查看、建立、編輯和刪除該資源。例如，IAM 支援大約 40 個用於使用者資源的動作，包括下列動作：

- `CreateUser`
- `DeleteUser`
- `GetUser`
- `UpdateUser`

若要允許主體執行操作，您必須在適用於主體或受影響資源的政策中包含必要的動作。若要查看每個服務支援的動作、資源類型和條件索引鍵清單，請參閱服務的動作、資源和條件索 AWS 引鍵。

資源

AWS 核准請求中的作業後，即可對您帳戶內的相關資源執行這些作業。資源是存在於服務內的物件。範例包括 Amazon EC2 執行個體、IAM 使用者和 Amazon S3 儲存貯體。該服務定義了可以在每個資源執行的一組動作。如果您建立了對資源執行不相關動作的請求，則拒絕該請求。例如，如果您請求刪除 IAM 角色但提供 IAM 群組資源，則請求將失敗。若要查看識別哪些資源受動作影響的 AWS 服務表，請參閱AWS 服務的動作、資源和條件索引鍵。

AWS 身分識別管理概觀：使用者

您可以將您的存取權授 AWS 帳戶 予特定使用者，並為他們提供特定的權限，以存取您的 AWS 帳戶. 您可以同時使用 IAM 和建立新 AWS IAM Identity Center 使用者或將現有使用者聯合到 AWS。兩者之間的主要差異在於 IAM 使用者會獲得 AWS 資源的長期登入資料，而 IAM Identity Center 中的使用者則擁有每次使用者登入時建立的臨時登入資料。 AWS最佳做法是要求人類使用者與身分識別提供者使用聯合，以 AWS 使用臨時登入資料而非身分 IAM 使用者存取。IAM 使用者的主要用途是讓無法使用 IAM 角色的工作負載能夠使用 API 或 CLI 向 AWS 服務發出程式設計請求。

主題

僅限第一次存取：您的根使用者憑證

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。僅有組織中的服務控制政策 (SCP) 可以限制授與根使用者的許可。

IAM 使用者和 IAM Identity Center 中的使用者

IAM 使用者不是分開帳戶，他們是您帳戶內的使用者。每個使用者都擁有自己的密碼來存取 AWS Management Console。您也可以為每個使用者建立個別的存取金鑰，讓使用者能利用程式設計方式提出請求，進而能使用您帳戶中的資源。

IAM 使用者會獲得資源的長期登入 AWS 資料。最佳實務是不要為人類使用者建立具有長期憑證的 IAM 使用者。而是要求您的人類使用者在存取時使用臨時認證 AWS。

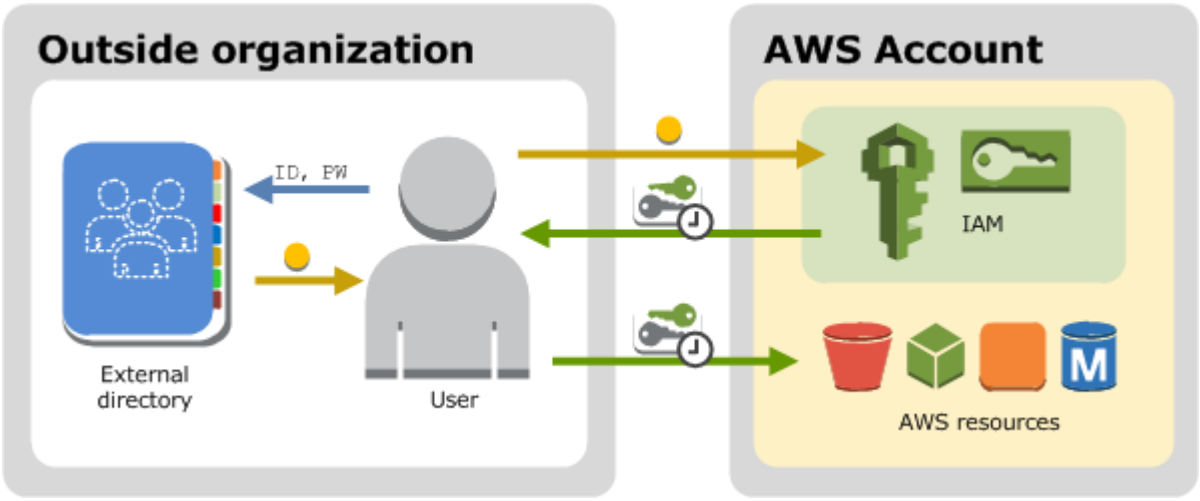
> **Note**
>
> 對於需要具有程式化存取和長期憑證的 IAM 使用者的情況，我們建議您視需要更新存取金鑰。如需詳細資訊，請參閱 更新存取金鑰。

相反地， AWS IAM Identity Center 的使用者 會授予您 AWS 資源的短期憑證。如果是集中式存取管理，我們建議您使用 AWS IAM Identity Center (IAM Identity Center) 管理您帳戶的存取權和這些帳戶內的許可。IAM Identity Center 會自動設定為身分識別中心目錄做為預設身分識別來源，您可以在其中建立使用者和群組，並將其存取層級指派給您的 AWS 資源。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 AWS IAM Identity Center。

聯合現有的使用者

如果組織中的使用者已有進行身分驗證的方式 (例如登入到公司網路)，則您不需要為他們建立個別 IAM 使用者，或建立 IAM Identity Center 使用者。相反地，您可以將這些使用者身分識別聯合到 AWS 使用 IAM 或 AWS IAM Identity Center.

下圖顯示使用者如何取得臨時 AWS 安全登入資料，以存取您的 AWS 帳戶.



在下列情況下，聯合特別有用：

- 您的使用者已存在於公司目錄。

  如果您的公司目錄與安全性宣告標記語言 2.0 (SAML 2.0) 相容，您可以將公司目錄設定為為使用者提供單一登入 (SSO) 存 AWS Management Console 取權。如需詳細資訊，請參閱 暫時性憑證的常見案例。

  如果您的公司目錄與 SAML 2.0 不相容，您可以建立身分識別代理應用程式， AWS Management Console 為您的使用者提供單一登入 (SSO) 存取權。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

  如果您的公司目錄是 Microsoft Active Directory，您可以用 AWS IAM Identity Center 來連接 Active Directory 中的自我管理目錄或中的目錄，以在AWS Directory Service您的公司目錄與您 AWS 帳戶的目錄之間建立信任。

如果您使用外部身分識別提供者 (IdP) (例如 Okta 或 Microsoft Entra) 來管理使用者，您可 AWS IAM Identity Center 以使用在 IdP 與您的. AWS 帳戶如需詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》 中的 連結至外部身分供應商。

- 您的使用者已經有網際網路身分。

  如果您正在建立行動應用程式或 Web 為基礎的應用程式，讓使用者透過網際網路身分提供者 (例如，Login with Amazon、Facebook、Google 或任何與 OpenID Connect (OIDC) 相容的身分提供者) 識別自己的身分，則該應用程式可以使用聯合功能存取 AWS。如需詳細資訊，請參閱 OIDC 聯盟。

  > **Tip**
  >
  > 若要搭配網際網路身分提供者使用聯合身分，建議您使用 Amazon Cognito。

存取控制方法

以下是您可以控制 AWS 資源存取權的方法。

使用者存取類型	為什麼要使用它？	我在哪裡可以獲得更多資訊？
針對人類使用者 (如您的人力使用者)，使用 IAM Identity Center 的單一登入存取來存取 AWS 資源	IAM 身分中心提供集中的位置，可將使用者的管理及其對雲端應用程式的存取 AWS 帳戶 權限整合在一起。 您可以在 IAM Identity Center 中設定身分存放區，或者您也可以使用現有的身分提供者 (IdP) 設定聯合。建議根據需要授予您的人力使用者限制 AWS 資源認證，作為安全性最佳實務。 使用者有更簡單的登入體驗，而您也能從單個系統持續控	如需有關設定 IAM Identity Center 的詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的入門。 如需有關使用 IAM Identity Center 中的 MFA 的詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的多重要素驗證。

使用者存取類型	為什麼要使用它？	我在哪裡可以獲得更多資訊？
	制他們的資源存取權。IAM Identity Center 支援多重要素驗證 (MFA)，以提高對帳戶安全的保護。	
針對人類使用者 (如您的人力使用者)，使用 IAM 身分提供者的聯合存取來存取 AWS 服務	與 OpenID Connect（OIDC）或 SAML 2.0（安全斷言標記語言 2.0）兼容的 IAM 支持 IdPs 。在建立 IAM 身分提供者以後，您必須建立一或多個可被動態指派給聯合身分使用者的 IAM 角色。	如需有關 IAM 身分提供者與聯合的詳細資訊，請參閱 身分提供者與聯合。
跨帳戶存取 AWS 帳戶	您想要與其他使用者共用特定 AWS 資源的存取權 AWS 帳戶。 角色是授予跨帳戶存取的主要方式。不過，一些 AWS 服務允許您直接將政策連接到資源 (而不是使用角色做為代理)。這些政策稱為資源型政策。	如需關於 IAM 角色的詳細資訊，請參閱 IAM 角色。 如需服務連結角色的詳細資訊，請參閱使用服務連結角色。 如需有關哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務。尋找服務連結角色欄中顯示 Yes (是) 的服務。若要檢視該服務的服務連結角色文件，請選擇該欄中與 Yes (是) 相關聯的連結。

使用者存取類型	為什麼要使用它？	我在哪裡可以獲得更多資訊？
指定 IAM 使用者的長期登入資料 AWS 帳戶	您可能有特定使用案例需要 IAM 使用者的長期登入資料 AWS。您可以使用 IAM 在您的中建立這些 IAM 使用者 AWS 帳戶，並使用 IAM 來管理其許可。以下是部分使用案例： • 無法使用 IAM 角色的工作負載 • 需要透過存取金鑰進行程式設計存取的協力廠商 AWS • AWS CodeCommit 或 Amazon Keyspaces 的服務特定登入資料 • AWS IAM Identity Center 您的帳戶無法使用，且您沒有其他身分提供者 對於需要具有程式化存取和長期憑證的 IAM 使用者的情況，我們建議的最佳實務是視需要更新存取金鑰。如需詳細資訊，請參閱 更新存取金鑰。	如需有關設定 IAM 使用者的詳細資訊，請參閱在您的中建立 IAM 使用者 AWS 帳戶。 如需 IAM 使用者存取金鑰的詳細資訊，請參閱管理 IAM 使用者的存取金鑰。 如需有關 AWS CodeCommit 或 Amazon Keyspaces 服務特定登入資料的詳細資訊，請參閱將 IAM 搭配使用 CodeCommit：Git 登入資料、安全殼層金鑰和 AWS 存取金鑰和。使用 IAM 與 Amazon Keyspaces (適用於 Apache Cassandra)

存取管理概述：許可和政策

AWS Identity and Access Management (IAM) 的存取管理部分可協助您定義主體在帳戶中允許執行的動作。主體實體是指使用 IAM 實體 (使用者或角色) 執行身分驗證的人員或應用程式。存取管理通常稱為授權。您可以 AWS 透過建立政策並將其附加到 IAM 身分 (使用者、使用者群組或角色) 或 AWS 資源來管理中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主體使用 IAM 實體 (使用者或角色) 提出要求時，評估這些政策。政策中的許可決定是否允許或拒絕請

求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需有關政策類型及其使用的詳細資訊，請參閱 IAM 中的政策和許可。

政策和帳戶

如果您在中管理單一帳戶 AWS，則您可以使用策略定義該帳戶內的權限。如果您跨多個帳戶管理許可，您的使用者會更難管理許可。對於跨帳戶許可，您可以使用 IAM 角色、以資源為基礎的政策或存取控制清單 (ACL)。但是，如果您擁有多個帳戶，我們建議您使用該 AWS Organizations 服務來幫助您管理這些權限。如需詳細資訊，請參閱什麼是 AWS Organizations？ 在《Organ izations 使用指南》中。

政策與使用者

IAM 使用者是服務中的身分。當您建立 IAM 使用者，在您提供許可前他們無法存取您帳戶中的任何項目。您可以藉由建立連接到使用者或該使用者所屬群組的以身分為基礎的政策，來提供許可給該使用者。下列範例顯示 JSON 政策，其允許使用者執行 `us-east-2` 區域內 `123456789012` 帳戶之 Books 資料表上的所有 Amazon DynamoDB 動作 (`dynamodb:*`)。

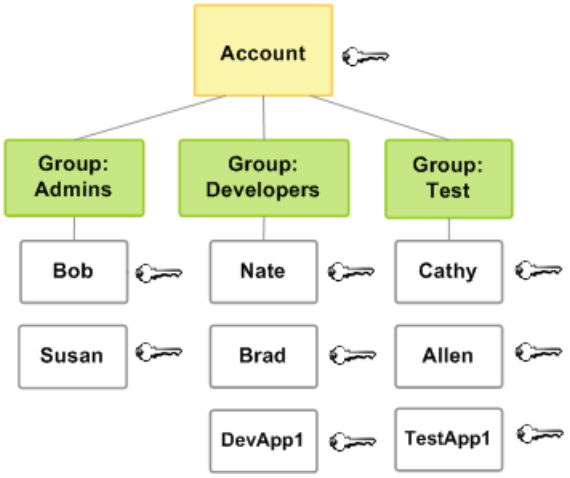
```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"
  }
}
```

在將此政策連接至您的 IAM 使用者後，該使用者僅擁有這些 DynamoDB 許可。大部分使用者有多個政策一起代表該使用者的許可。

根據預設，未明確允許的動作或資源會被拒絕。例如，如果上述政策是連接至使用者的唯一政策，則該使用者只被允許對 Books 資料表執行 DynamoDB 動作。對其他任何表格的動作被禁止。同樣地，使用者不允許在亞馬遜 EC2、Amazon S3 或任何其他 AWS 服務中執行任何動作。原因是政策中並未包含使用這些服務的許可。

政策和群組

您可以將 IAM 使用者組織到 IAM 群組並將政策連接到群組。在這種情況下，各個使用者仍有自己的憑證，但是群組中的所有使用者都擁有連接到群組的許可。使用群組來更輕鬆地進行許可管理，並遵循我們的 IAM 中的安全最佳實務。



使用者或群組可以有多個政策與之連接，分別授與不同的許可。這種情況下，使用者的許可會根據政策組合進行計算。不過基本原則仍然適用：如果未向使用者授與針對動作和資源的明確許可，則使用者無法擁有這些許可。

聯合身分使用者和角色

聯合身分使用者不會像 IAM 使用者那樣擁有永久身分。 AWS 帳戶 若要向聯合身分使用者指派許可，您可以建立稱為角色的實體，並為角色定義許可。聯合身分使用者登入時 AWS，使用者會與角色相關聯，而且會被授與角色中定義的權限。如需詳細資訊，請參閱 針對第三方身分提供者建立角色 (聯合身分)。

以身分為基礎和以資源為基礎的政策

以身分為基礎的政策是指您連接到 IAM 身分的許可政策，這些身分包括像是 IAM 使用者、群組或角色。以資源為基礎的政策是您連接到資源的許可政策，這些資源包括像是 Amazon S3 儲存貯體或 IAM 角色信任政策。

以身分為基礎政策可控制身分在何種條件下對哪些資源執行哪些動作。身分型政策可進一步分類：

- 受管理的策略 — 獨立以身分識別為基礎的策略，您可以將其附加到中的多個使用者、群組和角色。 AWS 帳戶您可以使用兩種類型受管政策：
  - AWS 受管理的策略 — 由建立和管理的受管理策略 AWS。如果您是使用原則的新手，建議您從使用 AWS 受管理的政策開始。
  - 客戶管理政策：您在 AWS 帳戶中建立和管理的受管政策。與受管政策相比 AWS ，客戶管理的政策可提供更精確的控制您的政策。您可以在視覺化編輯器建立和編輯和驗證 IAM 政策，或直接建立 JSON 政策文件。如需更多詳細資訊，請參閱 建立 IAM 政策 及 編輯 IAM 政策。

- 內嵌政策：由您建立和管理並直接嵌入到單一使用者、群組或角色的政策。在大多數情況下，我們建議您不要使用內嵌政策。

以資源為基礎的政策可控制指定的主體在何種條件下對該資源執行哪些動作。資源為基礎的政策是內嵌政策，而且沒有受管的以資源為基礎的政策。若要啟用跨帳戶存取，您可以指定一個完整帳戶或其他帳戶內的 IAM 實體做為以資源為基礎的政策的主體。

IAM 服務僅支援一種稱為角色信任政策的以資源為基礎的政策，其用以連接到 IAM 角色。由於 IAM 角色是支援以資源為基礎之政策的身分也是資源，所以您必須將信任政策和以身分為基礎的政策同時連接到 IAM 角色。信任政策會定義哪些主體實體 (帳戶、使用者、角色和聯合身分使用者) 可擔任該角色。若要了解 IAM 角色與其他以資源為基礎之政策之間的差別，請參閱 IAM 中的跨帳戶資源存取。

若要查看哪些服務支援以資源為基礎的政策，請參閱 AWS 與 IAM 搭配使用的服務。若要進一步了解以資源為基礎的政策的詳細資訊，請參閱 以身分為基礎和以資源為基礎的政策。

ABAC 是做什麼用的 AWS？

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。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 資源，包括 IAM 實體 (使用者或角色) 和 AWS 資源。您可以為您的 IAM 主體建立單一 ABAC 政策或是一組政策。這些 ABAC 政策可以設計成在主體的標籤與資源標籤相符時允許操作。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。

例如，您可以使用 `access-project` 標籤鍵建立三個角色。將第一個角色的鍵值設為 `Heart`，第二個角色的鍵值設為 `Star`，並將第三個角色的鍵值設為 `Lightning`。然後，您可以使用單一政策，在角色和資源針對 `access-project` 使用相同值標記時允許存取。如需示範如何在中使用 ABAC 的詳細教學課程 AWS，請參閱IAM 教學課程：根據標籤定義存取 AWS 資源的許可。若要瞭解支援 ABAC 的服務，請參閱 AWS 與 IAM 搭配使用的服務。

比較 ABAC 與傳統 RBAC 模型

IAM 中使用的傳統授權模型稱為角色類型存取控制 (RBAC)。RBAC 會根據人員的任務角色 (在 AWS 外稱為角色) 來定義許可。在 AWS 角色中通常是指 IAM 角色，這是您可以假設的 IAM 中的身分。IAM 確實包括在 RBAC 模型中將許可對齊到任務功能的用於任務角色的受管政策。

在 IAM 中，您可以為不同的任務角色建立不同的政策，來實作 RBAC。然後，您可以將政策連接到身分 (IAM 使用者、使用者群組或 IAM 角色)。根據最佳實務，您應授與任務角色所需要的最低許可。這稱為授與最低權限。透過列出任務功能所能存取的特定資源，即可執行此作業。使用傳統 RBAC 模型的缺點是當員工新增新的資源時，您必須更新政策以允許存取這些新資源。

例如，假設您有三個員工正在進行的專案，分別名為 Heart、Star 和 Lightning。您為每個專案建立 IAM 角色。您接著將政策連接到各 IAM 角色，定義允許擔任這些角色的人員所能存取的資源。若一名員工在您的公司內變更了職位，您便會將他們指派至不同的 IAM 角色。可將超過一個角色指派給人員或程式。但是，Star 專案可能需要其他資源，例如，新的 Amazon EC2 容器。在這種情況下，您必須更新連接到 Star 角色的政策，指定新的容器資源。否則，Star 專案的成員便無法存取新的容器。

與傳統 RBAC 模型相較，ABAC 提供了下列優點：

- ABAC 許可隨創新擴展。管理員不再需要更新現有政策來允許存取新的資源。例如，假設您使用 access-project 標籤設計您的 ABAC 策略。開發人員以 access-project = Heart 標籤使用角色。當 Heart 專案的人員需要額外 Amazon EC2 資源時，開發人員便可以使用 access-project = Heart 標籤建立新的 Amazon EC2 執行個體。接著，Heart 專案的任何人員便可以開始和停止這些執行個體，因為他們的標籤值相符。
- ABAC 需要的政策較少。由於您不需要為不同的任務功能建立不同政策，您建立的政策也會較少。這些政策較易於管理。
- 使用 ABAC，團隊便可以快速地改變和成長。這是因為新資源會自動根據屬性授與許可。例如，若您的公司已使用 ABAC 支援 Heart 和 Star，新增新的 Lightning 專案也相當容易。IAM 管理員可以使用 access-project = Lightning 標籤建立新的角色。不需要變更政策來支援新的專案。任何具備取得角色許可的人員都能建立和檢視以 access-project = Lightning 標記的執行個體。此外，團隊成員可能會從 Heart 專案移動至 Lightning 專案。IAM 管理員可以將使用者指派至不同的 IAM 角色。而不需要變更許可政策。
- 使用 ABAC 可實現精密許可。在您建立政策時，最佳實務是授與最低權限。使用傳統 RBAC，您必須撰寫政策，只允許存取特定資源。但是，當您使用 ABAC 時，您可以允許在所有資源上進行所有動作，但只有在資源標籤與主體標籤相符時才能進行。
- 搭配 ABAC 使用您企業目錄的員工屬性。您可以將 SAML 或 OIDC 提供者設定為將工作階段標記傳遞給。 AWS當您的員工聯合到時 AWS，他們的屬性會套用至其產生的主參與者。 AWS您接著可以使用 ABAC 來根據這些屬性允許或拒絕許可。

如需示範如何在中使用 ABAC 的詳細教學課程 AWS，請參閱IAM 教學課程：根據標籤定義存取 AWS 資源的許可。

IAM 外部的安全功能

您可以使用 IAM 來控制使用AWS 命令列工具或使用 AWS SDK 服務 API 作業執行之工作的存取。AWS Management Console有些 AWS 產品也有其他方法來保護其資源。以下列表提供了一些範例，但仍有未能列出者。

Amazon EC2

在 Amazon Elastic Compute Cloud 中，需要使用金鑰對 (對於 Linux 執行個體) 或使用使用者名稱和密碼 (對於 Windows 執行個體) 來登入執行個體。

如需詳細資訊，請參閱下列 文件：

- Amazon EC2 使用者指南中的 Amazon EC2 Linux 執行個體入門
- Amazon EC2 使用者指南中的亞 Amazon EC2 視窗執行個體入門

Amazon RDS

在 Amazon Relational Database Service 中，需要使用與資料庫綁定的使用者名稱和密碼來登入資料庫引擎。

如需詳細資訊，請參閱 Amazon RSD 使用者指南中的 Amazon RDS 入門。

Amazon EC2 和 Amazon RDS

在 Amazon EC2 與 Amazon RDS 中，需要使用安全群組來控制發送到執行個體或資料庫的流量。

如需詳細資訊，請參閱下列 文件：

- Amazon EC2 使用者指南中適用於 Linux 執行個體的 Amazon EC2 安全群組
- Amazon EC2 使用者指南中適用於 Windows 執行個體的 Amazon EC2 安全群組
- Amazon RDS 使用者指南中的 Amazon RDS 安全群組

WorkSpaces

在 Amazon 中 WorkSpaces，使用者使用使用者名稱和密碼登入桌面。

如需詳細資訊，請參閱 Amazon WorkSpaces 管理指南 WorkSpaces中的入門指南。

Amazon WorkDocs

在 Amazon 中 WorkDocs，使用者可以使用使用者名稱和密碼登入來存取共用文件。

有關更多信息，請參閱 Amazon WorkDocs 管理指南 WorkDocs中的入門使用 Amazon。

這些存取控制方法不屬於 IAM。IAM 可讓您控制這些 AWS 產品的管理方式，包括建立或終止 Amazon EC2 執行個體、設定新 WorkSpaces桌面等。也就是說，IAM 可幫助您控制對 Amazon Web Services 提出請求來執行的任務，並且可幫助您控制對 AWS Management Console的存取。不過，IAM 無法協助您管理登入作業系統 (Amazon EC2)、資料庫 (Amazon RDS)、桌面 (Amazon) 或協作網站 (Amazon WorkSpaces WorkDocs) 等任務的安全性。

當您使用特定 AWS 產品時，請務必閱讀文件，以瞭解屬於該產品之所有資源的安全性選項。

快速連結到常見任務

使用以下連結來取得與 IAM 相關的常見任務的說明。

登入不同的使用者類型

選擇 IAM 使用者並輸入您的 AWS 帳戶 ID 或帳戶別名，以登入 IAM 主控台。在下一頁中，輸入您的 IAM 使用者名稱和密碼。

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，請參閱使用指南中的登入 AWS 存取入口網站。AWS 登入

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。

如需判斷你的使用者類型和登入頁面，請參閱《AWS 登入 使用者指南》中的什麼是 AWS 登入。

管理 使用者的密碼

您需要密碼才能存取 AWS Management Console，包括帳單資訊的存取權。

對於您的 AWS 帳戶根使用者，請參閱《AWS Account Management 參考指南》 AWS 帳戶根使用者中的更改密碼

對於 IAM 使用者，請參閱 管理 IAM 使用者密碼。

管理 使用者的許可

您可以使用政策將許可授與 AWS 帳戶. IAM 使用者在建立時沒有許可，因此您必須新增許可以允許他們使用 AWS 資源。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 使用者和群組位於 AWS IAM Identity Center：

建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 建立權限合集 說明進行操作。

- 透過身分提供者在 IAM 中管理的使用者：

  建立聯合身分的角色。請按照 IAM 使用者指南 的 為第三方身分提供者 (聯合) 建立角色 中的指示進行操作。

- IAM 使用者：
  - 建立您的使用者可擔任的角色。請按照 IAM 使用者指南 的 為 IAM 使用者建立角色 中的指示進行操作。
  - (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南的新增許可到使用者 (主控台)中的指示。

如需詳細資訊，請參閱 管理 IAM 政策。

列出您的用戶 AWS 帳戶 並獲取有關其憑據的信息

請參閱取得 AWS 帳戶的憑證報告。

新增 多重要素驗證 (MFA)

若要新增虛擬 MFA 裝置，請參閱下列其中一項：

- 針對 AWS 帳戶根使用者 啟用虛擬 MFA 裝置 (主控台)
- 針對 IAM 使用者啟用虛擬 MFA 裝置 (主控台)

若要新增 FIDO 安全性金鑰，請參閱下列其中一項：

- 啟用 AWS 帳戶根使用者 (控制台) 的金鑰或安全金鑰
- 為其他 IAM 使用者 (主控台) 啟用金鑰或安全金鑰

若要新增硬體 MFA 裝置，請參閱下列其中一項：

- 啟用 AWS 帳戶根使用者 （ 控制台 ） 的硬件 TOTP 令牌.
- 啟用另一個 IAM 使用者的硬體 TOTP 權杖 (主控台)

取得存取金鑰

您可以使用存取金鑰來使用 AWS SDK、AWS 命令列工具或 API 作業發出 AWS 要求。

> **Important**
>
> 最佳實務的作法是使用臨時性安全憑證 (如 IAM 角色)，而不是建立長期憑證 (如存取金鑰)。在建立存取金鑰前，請檢閱長期存取金鑰的替代方案。

如需協助您保護存取金鑰的指引，請參閱保護存取金鑰。

若要了解如何管理 IAM 使用者的存取金鑰，請參閱 管理 IAM 使用者的存取金鑰。

如需有關可用於您的安全登入資料的詳細資訊 AWS 帳戶，請參閱AWS 安全性認證。

標記 IAM 資源

您可以標記以下 IAM 資源：

- IAM 使用者
- IAM 角色
- 客戶受管政策
- 身分提供者
- 伺服器憑證
- 虛擬 MFA 裝置

若要進一步了解 IAM 中的標籤，請參閱 標記 IAM 資源。

若要瞭解如何使用標籤來控制 AWS 資源的存取，請參閱使用標籤控制 AWS 資源的存取。

檢視所有服務的動作、資源和條件索引鍵

這組參考文件可協助您撰寫詳細的 IAM 政策。每個 AWS 服務都會定義您可以在 IAM 政策中使用的動作、資源和條件內容索引鍵。若要深入了解，請參閱AWS 服務的動作、資源和條件金鑰。

開始使用所有的 AWS

這組文件主要處理 IAM 服務。若要瞭解如何開始使用 AWS 和使用多項服務來解決問題 (例如建置和啟動第一個專案)，請參閱入門資源中心。

IAM 主控台搜尋

使用 IAM 主控台搜尋頁面做為更快速的 IAM 資源搜索選項。您可以使用主控台搜尋來尋找與帳戶、IAM 實體 (例如使用者、群組、角色、身分識別提供者) 相關的存取金鑰，以及依名稱的政策等。

IAM 主控台搜尋功能可以尋找下列任何內容：

- 符合您的搜尋關鍵字 IAM 實體名稱 (適用於使用者、群組、角色、身分提供者和政策)
- 符合您搜尋關鍵字的任務

IAM 主控台搜尋功能不會傳回 IAM Access Analyzer 的相關資訊。

在搜尋結果的每一列都是作用中的連結。例如，您可以在搜尋結果中選擇使用者名稱，該使用者名稱將帶您前往該使用者的詳細資訊頁面。或者，您可以選擇動作連結 (例如建立使用者) 前往 Create User (建立使用者) 頁面。

> **Note**
>
> 存取金鑰搜尋要求您在搜尋方塊中輸入完整的存取金鑰 ID。搜尋結果顯示與該金鑰關聯的使用者。然後您可以直接導覽至該使用者的頁面，您可以在其中管理存取金鑰。

使用 IAM 主控台搜尋

使用 IAM 主控台中的 Search (搜尋) 頁面找到與該帳戶相關的項目。

在 IAM 主控台中搜尋項目

1. 按照《AWS 登入使用者指南》 如何登入 AWS 主題中適合您使用者類型的登入程序操作。
2. 在 主控台首頁 頁面上選取 IAM 服務。
3. 在導覽窗格中，選擇 Search (搜尋)。
4. 在 Search (搜尋) 方塊中，輸入您的搜尋關鍵字。
5. 在搜尋結果清單中選擇連結來導覽至主控台的相應部分。

在 IAM 主控台搜尋結果中的圖示

以下圖示識別搜尋發現的項目類型：

圖示	描述
	IAM 使用者
	IAM 群組
	IAM 角色

圖示	描述
	IAM 政策
	任務，例如「建立使用者」或「連接政策」
	關鍵字 **delete** 的結果

搜尋字詞範例

您可以在 IAM 搜尋中使用以下字詞。用您想要尋找的實際 IAM 使用者、群組、角色、存取金鑰、政策或身分提供者的名稱取代斜體中的術語。

- ***user_name*** 或 ***group_name*** 或 ***role_name*** 或 ***policy_name*** 或 ***identity_provider_name***
- ***access_key***
- **add user *user_name* to groups** 或 **add users to group *group_name***
- **remove user *user_name* from groups**
- **delete *user_name*** 或 **delete *group_name*** 或 **delete *role_name***、或 **delete *policy_name***、或 **delete *identity_provider_name***
- **manage access keys *user_name***
- **manage signing certificates *user_name***
- **users**
- **manage MFA for *user_name***
- **manage password for *user_name***
- **create role**
- **password policy**
- **edit trust policy for role *role_name***
- **show policy document for role *role_name***
- **attach policy to *role_name***
- **create managed policy**
- **create user**

- **create group**
- **attach policy to *group_name***
- **attach entities to *policy_name***
- **detach entities from *policy_name***

建立 AWS Identity and Access Management 資源 AWS CloudFormation

AWS Identity and Access Management 與整合的服務可協助您建立資源模型並設定資 AWS 源 AWS CloudFormation，以減少建立和管理資源和基礎架構的時間。您可以建立一個範本，說明所需的所有 AWS 資源 (例如存取金鑰、群組、群組原則、執行個體設定檔、受管理原則、OIDC 提供者、內嵌原則、角色原則、SAML 提供者、伺服器憑證、服務連結角色、使用者 (以及將使用者新增至群組)、使用者原則和虛擬 MFA 裝置)，以及 AWS CloudFormation 佈建和設定這些資源給您。

使用時 AWS CloudFormation，您可以重複使用範本，以一致且重複地設定 IAM 資源。描述您的資源一次，然後在多個區域中一遍又一遍地佈建相同 AWS 帳戶 的資源。

IAM 和 AWS CloudFormation 範本

若要佈建和設定 IAM 和相關服務的資源，您必須瞭解AWS CloudFormation 範本。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您要在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，可以使用 AWS CloudFormation 設計工具來協助您開始 AWS CloudFormation 使用範本。如需更多詳細資訊，請參閱 AWS CloudFormation 使用者指南 中的 什麼是 AWS CloudFormation 設計器？。

IAM 支援建立存取金鑰、群組、群組政策、執行個體設定檔、受管政策、OIDC 提供者、內嵌政策、角色、角色政策、SAML 提供者、伺服器憑證、服務連結角色、使用者 (以及將使用者新增至群組)、使用者政策和中的虛擬 MFA 裝置。 AWS CloudFormation如需詳細資訊，包括 IAM 資源的 JSON 和 YAML 範本範例，請參閱AWS CloudFormation 使用者指南中的AWS Identity and Access Management 資源類型參考資料。

您也可以建立建立相關資源的範本，例如角色和受管理的策略。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation 使用者指南
- AWS CloudFormation API 參考
- AWS CloudFormation 指令行介面使用者指南

使用 AWS CloudShell 來使用 AWS Identity and Access Management

AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console 您可以使用偏好的殼層 (Bash PowerShell 或 Z 殼層) 針對 AWS 服務 (包括 AWS Identity and Access Management) 執行 AWS CLI 命令。另外，您無需下載或安裝命令列工具即可執行此操作。

您可以AWS CloudShell 從啟動 AWS Management Console，並且您用來登入主控台的 AWS 認證會自動在新的 shell 工作階段中使用。這種使用 AWS CloudShell 者的預先驗證可讓您在使用 AWS CLI 版本 2 (預先安裝在殼層運算環境中) 與 IAM 等 AWS 服務互動時略過設定登入資料。

取得的 IAM 許可 AWS CloudShell

管理員可以使用提供的存取管理資源將許可授予 IAM 使用者 AWS Identity and Access Management，以便他們可以存取 AWS CloudShell 和使用環境的功能。

系統管理員授與使用者存取權的最快方法是透過 AWS 受管理的原則。AWS 受管政策是由 AWS建立並管理的獨立政策。的下列 AWS 受管政策 CloudShell 可附加至 IAM 身分：

- `AWSCloudShellFullAccess`：授予使用權限 AWS CloudShell 以完全訪問所有功能。

如果您想要限制 IAM 使用者可以執行的動作範圍 AWS CloudShell，您可以建立使用`AWSCloudShellFullAccess`受管政策做為範本的自訂政策。如需有關限制中使用者可使用的動作的詳細資訊 CloudShell，請參閱《使用AWS CloudShell 者指南》中的「使用 IAM 政策管理 AWS CloudShell 存取和使用」。

使用與 IAM 進行互動 AWS CloudShell

AWS CloudShell 從啟動之後 AWS Management Console，您可以立即開始使用命令列介面與 IAM 互動。

> **Note**
>
> AWS CLI 在中使用時 AWS CloudShell，您不需要下載或安裝任何其他資源。此外，因為您已經在 Shell 中驗證身分，因此無需設定憑證即可呼叫。

建立 IAM 群組並使用下列方式將 IAM 使用者新增至群組 AWS CloudShell

下列範例使 CloudShell 用建立 IAM 群組、將 IAM 使用者新增至群組，然後驗證命令是否成功。

1. 從中 AWS Management Console，您可以選擇 CloudShell 導覽列上的下列可用選項來啟動：
   - 選擇圖 CloudShell 示。
   - 開始在搜索框中輸入「cloudshell」，然後選擇該 CloudShell選項。
2. 若要建立 IAM 群組，請在命令列中輸入下列 CloudShell 命令。在這個範例中，我們將群組命名為 east_coast：

   ```
   aws iam create-group --group-name east_coast
   ```

   如果呼叫成功，命令列會顯示類似下列輸出的服務回應：

   ```
   {
      "Group": {
          "Path": "/",
          "GroupName": "east_coast",
          "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
          "Arn": "arn:aws:iam::111122223333:group/east_coast",
          "CreateDate": "2023-09-11T21:02:21+00:00"
       }
   }
   ```

3. 若要將使用者新增至您建立的群組，請使用下列命令，指定群組名稱和使用者名稱。在這個範例中，我們將群組命名為 east_coast，將使用者命名為 johndoe：

   ```
   aws iam add-user-to-group --group-name east_coast --user-name johndoe
   ```

4. 若要驗證使用者是否在群組中，請使用下列命令，指定群組名稱。在這個範例中，我們繼續使用群組 east_coast：

```
aws iam get-group --group-name east_coast
```

如果呼叫成功，命令列會顯示類似下列輸出的服務回應：

```
{
  "Users": [
    {
        "Path": "/",
        "UserName": "johndoe",
        "UserId": "AIDAYBDBW4JBXGEXAMPLE",
        "Arn": "arn:aws:iam::552108220995:user/johndoe",
        "CreateDate": "2023-09-11T20:43:14+00:00",
        "PasswordLastUsed": "2023-09-11T20:59:14+00:00"
    }
  ],
  "Group": {
        "Path": "/",
        "GroupName": "east_coast",
        "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
        "Arn": "arn:aws:iam::111122223333:group/east_coast",
        "CreateDate": "2023-09-11T21:02:21+00:00"
     }
 }
```

搭配 AWS 開發套件使用 IAM

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

SDK 文件	代碼範例
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 程式碼範例的工具
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 IAM 專屬範例，請參閱 使用 AWS 軟體開發套件的 IAM 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。

開始使用 IAM 設定

> **Important**
>
> IAM 最佳實務建議您要求人類使用者與身分識別提供者的聯合使用，以 AWS 使用臨時登入資料存取，而不是使用具有長期登入資料的 IAM 使用者。

AWS Identity and Access Management (IAM) 可協助您安全地控制對 Amazon Web Services (AWS) 和帳戶資源的存取。IAM 還可以讓您的登入認證保持私有。您不需專門註冊即可使用 IAM。使用 IAM 免費。

使用 IAM 為使用者和角色等身分提供您的帳戶中資源的存取權。例如，您可以將 IAM 與公司目錄中的現有使用者搭配使用，這些使用者在外部管理， AWS 也可以在中建立 AWS 使用者 AWS IAM Identity Center。聯合身分會擔任定義的 IAM 角色，以存取他們所需的資源。如需有關 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

> **Note**
>
> IAM 與多種 AWS 產品整合。如需支援 IAM 的服務清單，請參閱 AWS 與 IAM 搭配使用的服務。

主題

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。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。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，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和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 帳戶根使用者

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」。

為最低權限許可做好準備

使用 最低權限許可 是 IAM 的最佳實務建議。最低權限許可的概念是指僅授與使用者執行任務所需的許可，而且無需額外許可。設定完成後，請思考支援最低權限許可的方式。根使用者和管理員使用者皆具備執行日常任務時不需要的強大許可。當您在了解 AWS 並測試不同的服務時，我們建議您在 IAM Identity Center 中建立至少一個額外的使用者，而這些使用者可以在不同情況下使用較少的許可。您可以使用 IAM 政策定義特定條件下可對特定資源採取的動作，然後透過權限較低的帳戶連結至這些資源。

如果您使用的是 IAM Identity Center，請考慮使用 IAM Identity Center 許可集來展開行動。若要了解詳情，請參閱《IAM Identity Center 使用者指南》中的建立許可集。

如果您使用的不是 IAM Identity Center，請使用 IAM 角色為不同的 IAM 實體定義許可。如需進一步了解，請參閱建立 IAM 角色。

IAM 角色和 IAM 身分中心許可集都可以根據工作職能使用 AWS 受管政策。如需這些政策所授與權限的詳細資訊，請參閱 AWS 受管理的工作職能政策。

> **Important**
>
> 請記住， AWS 受管理的政策可能不會針對您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。完成設定後，建議您使用 IAM Access Analyzer 來根據記錄

在 AWS CloudTrail的存取活動產生最低權限政策。如需政策產生的詳細資訊，請參閱 IAM Access Analyzer 政策產生。

IAM 管理方法

您可以使用 AWS 主控台、 AWS 指令列介面或透過相關 SDK 中的應用程式介面 (API) 來管理 IAM。在設定時，請考慮您想要支援哪些方法以及您計劃如何支援不同的使用者。

主題

- AWS 控制台
- AWS 命令列介面 (CLI) 和軟體開發套件 (SDK)

AWS 控制台

AWS 管理主控台是一種 Web 應用程式，其中包含並參照用於管理 AWS 資源的廣泛服務主控台集合。若是首次登入，這時主控台頁面將會顯示。首頁可讓您存取每個服務主控台，並提供單一位置來存取執行 AWS 相關工作的資訊。登入主控台後，您可以使用哪些服務和應用程式，取決於您有權存取的 AWS 資源。您可以透過擔任某個角色、加入已取得許可的群組或明確取得相關許可，從而獲得對資源的許可。對於獨立 AWS 帳戶，根使用者或 IAM 管理員會設定資源的存取權限。對於 AWS Organizations，管理帳戶或委派的系統管理員可設定資源的存取權。

如果您打算讓人員使用 AWS 管理主控台來管理 AWS 資源，建議您使用臨時登入資料來設定使用者，做為安全性最佳作法。已擔任角色的 IAM 使用者、聯合身分使用者和 IAM Identity Center 中的使用者有臨時憑證，而 IAM 使用者和根使用者有長期憑證。根使用者憑證提供對 AWS 帳戶的完整存取權，但其他使用者所擁有的憑證讓他們可以存取由 IAM 政策授予的資源。

登入體驗會因不同類型的 AWS Management Console 使用者而有所不同。

- IAM 使用者和根使用者從主要 AWS 登入網址 (https://signin.aws.amazon.com) 登入。一旦登入，他們便可存取已取得許可的帳戶中的資源。

  若要以根使用者身分登入，您必須有根使用者電子郵件地址和密碼。

  若要以 IAM 使用者身分登入，您必須具有 AWS 帳戶 編號或別名、IAM 使用者名稱和 IAM 使用者密碼。

  我們建議您將帳戶中的 IAM 使用者限定於需要長期憑證的特定情形，例如緊急存取，而且您只會在需要根使用者憑證的任務中使用根使用者。

為了方便起見，AWS 登入頁面會使用瀏覽器 Cookie 來記住 IAM 使用者名稱和帳戶資訊。下次使用者前往中的任何頁面時 AWS Management Console，主控台會使用 Cookie 將使用者重新導向至帳戶登入頁面。

完成您的工作階段後，請登出主控台以防止重複使用您先前的登入。

- IAM 身分中心使用者使用其組織專屬的特定 AWS 存取入口網站登入。登入後，他們可以選擇要存取的帳戶或應用程式。如果選擇存取某個帳戶，他們會選擇想要在管理工作階段中使用哪個許可集。
- 外部身分提供者管理的聯合身分使用者會連結至使用自訂企業存取入口網站的 AWS 帳戶 登入。聯合身分使用者可用的 AWS 資源取決於其組織選取的策略。

> **Note**
>
> 為了提供額外的安全性，IAM 身分中心的根使用者、IAM 使用者和使用者在授予資源存取權 AWS 之前，可以先通過驗證多重要 AWS 素身份驗證 (MFA)。當啟用 MFA 時，您還必須能夠存取要登入的 MFA 裝置。

若要深入瞭解不同使用者如何登入管理主控台，請參閱登入使用者指南中的AWS 登入 AWS 管理主控台。

AWS 命令列介面 (CLI) 和軟體開發套件 (SDK)

當 IAM Identity Center 和 IAM 使用者透過相關 SDK 中的 CLI 或應用程式介面 (API) 驗證身分時，他們會使用不同方法來驗證其憑證。

認證和組態設定位於多個位置，例如系統或使用者環境變數、本機 AWS 組態檔案，或在命令列上明確宣告為參數。某些位置的優先順序高於其他位置。

IAM Identity Center 和 IAM 均提供可與 CLI 或 SDK 搭配使用的存取金鑰。IAM Identity Center 存取金鑰是可自動整理的臨時憑證，並且優於與 IAM 使用者相關聯的長期存取金鑰。

要 AWS 帳戶 使用 CLI 或 SDK 管理您可以 AWS CloudShell 從瀏覽器中使用。如果您使 CloudShell 用執行 CLI 或 SDK 命令，您必須先登入主控台。存取資 AWS 源的權限取決於您用來登入主控台的認證。根據您的經驗，您可能會發現 CLI 是管理 AWS 帳戶的更有效率的方法。

對於應用程式開發，您可以將 CLI 或 SDK 下載到您的電腦，然後從命令提示或 Docker 視窗進行登入。在這種情況下，您要將身分驗證和存取憑證設定為 CLI 指令碼或 SDK 應用程式的一部分。您可以採用不同的方式設定對資源的程式設計存取，具體取決於適用於您的環境和存取權。

- 建議使用 AWS 服務驗證本機程式碼的選項是 IAM 身分中心和 IAM 角色隨處
- 驗證在 AWS 環境中執行的程式碼的建議選項是使用 IAM 角色或使用 IAM 身分中心登入資料。

如果您使用 IAM Identity Center，可在 AWS 存取入口網站的開始頁面取得短期憑證，並在該頁面上選擇您的許可集。這些憑證具有定義的持續時間，不會自動重新整理。如果您想要使用這些認證，請在登 AWS 入入口網站後選擇， AWS 帳戶 然後選擇權限集。選取命令列或程式設計存取，以檢視可用來以程式設計方式或從 CLI 存取 AWS 資源的選項。如需有關這些方法的詳細資訊，請參閱《IAM Identity Center 使用者指南》中的取得與重新整理臨時憑證。這些憑證經常在應用程式開發期間使用，以便快速測試程式碼。

我們建議您使用 IAM 身分中心登入資料，以便在自動存取 AWS 資源時自動重新整理。如果您已在 IAM Identity Center 中設定使用者和許可集，您可以使用 `aws configure sso` 命令以便利用命令列精靈來協助您識別可用的憑證，並且將它們存放在設定檔中。如需有關設定您的設定檔的更多資訊，請參閱《AWS 命令列介面使用者指南 (適用於版本 2)》中的使用 `aws configure sso` 精靈設定您的設定檔。

> **Note**
>
> 許多範例應用程式使用與 IAM 使用者或根使用者相關聯的長期存取金鑰。您應該只在沙盒環境中使用長期憑證，這是您學習練習的一部分。檢閱長期存取金鑰的替代方案並制定轉換程式碼計畫，使其儘快使用替代憑證，例如 IAM Identity Center 憑證或 IAM 角色。在轉換程式碼後，請刪除存取金鑰。

若要進一步了解有關設定 CLI 的資訊，請參閱《AWS 命令列介面使用者指南》第 2 版的《安裝或更新最新版的 AWS CLI》和《AWS 命令列介面使用者指南》中的驗證和存取認證

若要詳細了解如何設定軟體開發套件，請參閱《AWS SDK 和工具參考指南》中的 IAM Identity Center 身分驗證和《AWS 軟體開發套件和工具參考指南》中的 IAM Roles Anywhere。

您的 AWS 帳戶 ID 及其別名

帳戶中的 IAM 使用者使用包含帳戶別名或帳戶 ID 的 Web URL 登入。如果您沒有 URL， AWS 登入頁面會要求您提供 AWS 帳戶 別名或帳戶 ID。

如果您不知道自己的帳戶 ID 或別名：

- 檢查瀏覽器歷史記錄。如果您之前已登入，帳戶 ID 或別名可能儲存在您最近瀏覽的網站中。

- 如果您已使用帳戶憑據配置 AWS CLI 或 AWS SDK，則可以從配置文件中獲取您的帳戶 ID。
- 詢問您的本機系統管理員或帳戶擁有者， AWS 無法提供帳戶 ID 給使用者。

> **Tip**
>
> 要在網頁瀏覽器為您的帳戶登入頁面建立書籤，您應該在書籤項目手動輸入登入 URL。請勿使用網頁瀏覽器的「將此頁面加入書籤」功能，因為這會擷取您目前瀏覽器工作階段的特定資訊，進而對日後造訪登入頁面造成干擾。

主題

檢視您的 AWS 帳戶 身份證

您可以使用下列方法檢視您 AWS 帳戶 的帳號 ID。

使用主控台檢視您的帳戶 ID

帳戶 ID 會顯示在 AWS 帳戶 區段的 IAM 儀表板上。視您的使用者類型而定，還有其他方法可以在主控台中檢視您的帳戶 ID。如果您已擔任角色，則無法使用安全認證。

使用者類型	程序
根使用者	在右上角的導覽列中，選擇您的使用者名稱，然後選擇 [安全認證]。帳戶號碼會顯示在帳戶識別碼下方。
IAM 使用者	在右上角的導航欄中，選擇您的用戶名，帳戶 ID 顯示在您的用戶名上方。選擇 Security credentials (安全登入資料)。帳戶號碼會顯示在帳戶詳細資下方。

使用者類型	程序
聯合身分使用者	在右上角的導航欄中，選擇您的用戶名，帳戶 ID 顯示在您的用戶名上方。
擔任的角色	在右上角的導覽列中，選擇「Sup port」圖示，然後從清單中選擇「Sup port 中心」。您目前登入的 12 位數帳戶號碼 (ID) 會顯示在 Support Center (支援中心) 導覽窗格。

檢視您的帳戶 ID AWS CLI

使用以下命令來檢視您的使用者 ID、帳戶 ID 和您的使用者 ARN：

- AWS get-caller-identity

使用 API 檢視您的帳戶 ID

使用以下 API 來檢視您的使用者 ID、帳戶 ID 及您的使用者 ARN：

- GetCallerIdentity

關於帳戶別名

如果您希望登入頁面的 URL 包含您的公司名稱 (或其他易記識別碼)，而非您的 AWS 帳戶 ID，您可以建立帳戶別名。本節提供 AWS 帳戶 別名的相關資訊，並列出您用來建立別名的 API 作業。

預設情況下，您的登入頁面 URL 有以下格式。

```
https://Your_Account_ID.signin.aws.amazon.com/console/
```

如果您為 AWS 帳戶 ID 建立 AWS 帳戶 別名，您的登入頁面 URL 看起來如下範例所示。

```
https://Your_Account_Alias.signin.aws.amazon.com/console/
```

考量事項

- 您只 AWS 帳戶 能有一個別名。如果為您的 AWS 帳戶建立新的別名，則新別名將覆寫之前的別名，並且包含之前別名的 URL 將停止運作。
- 帳戶別名必須僅包含數字、小寫字母和連字號。如需 AWS 帳戶實體限制的詳細資訊，請參閱IAM 和 AWS STS 配額。
- 帳戶別名在指定網路分割區內所有 Amazon Web Services 產品中必須是唯一的。

  分割區是一組 AWS 區域。每個 AWS 帳戶的範圍都限定在一個分割區。

  以下是支援的分割區：

  - `aws`- AWS 地區
  - `aws-cn` - 中國區域
  - `aws-us-gov`- AWS GovCloud (US) 地區

建立、刪除和列出 AWS 帳戶 別名

您可以使 AWS Management Console用 IAM API 或命令列介面來建立或刪除 AWS 帳戶 別名。

> **Note**
>
> 帳戶別名並非秘密，它們會顯示在您的面向公眾的登入頁面 URL 中。請勿在您的帳戶別名中包含任何敏感資訊。
> 包含您 AWS 帳戶 ID 的原始 URL 會保持作用中，並可在您建立 AWS 帳戶 別名後使用。

建立或編輯帳戶別名 (主控台)

您可以從 AWS Management Console建立、編輯和刪除帳戶別名。

> **最低許可**
>
> 若要執行下列步驟，您至少必須擁有下列 IAM 許可：
>
> - `iam:ListAccountAliases`
> - `iam:CreateAccountAlias`

建立或編輯帳戶別名 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Dashboard (儀表板)。
3. 在 AWS 帳戶區段中，選擇帳戶別名旁邊的建立。如果別名已存在，請選擇 Edit (編輯)。
4. 在對話方塊中，輸入要用作別名的名稱，然後選擇儲存變更。

> **Note**
>
> 您一次只能有一個與您關聯 AWS 帳戶 的別名。如果您建立新別名，則會移除先前的別名，且與先前別名相關聯的登入 URL 會停止運作。

刪除帳戶別名 (主控台)

您可以從 AWS Management Console刪除帳戶別名。

> **最低許可**
>
> 若要執行下列步驟，您至少必須擁有下列 IAM 許可：
>
> - `iam:ListAccountAliases`
> - `iam:CreateAccountAlias`
> - `iam:DeleteAccountAlias`

刪除帳戶別名 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Dashboard (儀表板)。
3. 在 AWS 帳戶區段中，選擇帳戶別名旁邊的刪除。

> **Note**
>
> 帳戶的唯一登入 URL 取決於帳戶 ID。連線至別名 URL 的任何嘗試都不會重新導向。

建立、刪除和列出別名 (AWS CLI)

> **Note**
>
> 若要使用下列命令，您至少必須擁有下列 IAM 許可：
>
> - `iam:ListAccountAliases`
> - `iam:CreateAccountAlias`
> - `iam:DeleteAccountAlias`

若要為您的 AWS Management Console 登入頁面 URL 建立別名，請執行下列命令：

- `aws iam create-account-alias`

若要刪除 AWS 帳戶 ID 別名，請執行下列命令：

- `aws iam delete-account-alias`

若要顯示您的 AWS 帳戶 ID 別名，請執行下列命令：

- `aws iam list-account-aliases`

Example 別名命令

若要顯示您的 AWS 帳戶 ID 別名，請執行下列命令。

```
$ aws iam list-account-aliases
{
    "AccountAliases": [
        "myaccountalias"
    ]
}
```

若要為您的 AWS Management Console 登入建立別名，請執行下列命令：

```
$ aws iam create-account-alias \
    --account-alias myaliasname
```

此命令如果成功就不會產生輸出。

若要刪除 AWS 帳戶 ID 別名，請執行下列命令。

```
$ aws iam delete-account-alias \
--account-alias myaliasname
```

此命令如果成功就不會產生輸出。

建立、刪除和列出別名 (AWS API)

> **Note**
>
> 若要使用下列 API 操作，您至少必須擁有下列 IAM 許可：
>
> - `iam:ListAccountAliases`
> - `iam:CreateAccountAlias`
> - `iam:DeleteAccountAlias`

若要為您的 AWS Management Console 登入頁面 URL 建立別名，請呼叫下列作業：

- `CreateAccountAlias`

若要刪除 AWS 帳戶 ID 別名，請呼叫下列作業：

- `DeleteAccountAlias`

若要顯示您的 AWS 帳戶 ID 別名，請呼叫下列作業：

- `ListAccountAliases`

IAM 入門

使用此教學課程可開始使用 AWS Identity and Access Management (IAM)。您將了解如何使用 AWS Management Console建立角色、使用者及政策。

AWS Identity and Access Management 是您 AWS 帳戶 提供的一項功能，不收取額外費用。您只需支付 IAM 使用者使用其他 AWS 產品的費用。如需其他 AWS 產品定價的相關資訊，請參閱 Amazon Web Services 定價頁面。

> **Note**
>
> 這組文件主要處理 IAM 服務。若要瞭解如何開始使用 AWS 和使用多項服務來解決問題 (例如建置和啟動第一個專案)，請參閱入門資源中心。

目錄

必要條件

開始之前，請務必先完成 開始使用 IAM 設定 中的步驟。本教學課程會使用您在該程序中建立的管理員帳戶。

建立第一位 IAM 使用者

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。使用者可以安排至共用相同許可的群組。

> **Note**
>
> 我們的安全最佳實務是建議您透過聯合身分來提供資源存取權限，而不是建立 IAM 使用者。若要了解需要 IAM 使用者的特定情形，請參閱建立 IAM 使用者 (而非角色) 的時機。

為讓您熟悉建立 IAM 使用者的程序，本教學課程將逐步引導您建立 IAM 使用者和群組，以應對緊急存取狀況。

建立第一位 IAM 使用者

1. 按照《AWS 登入使用者指南》 如何登入 AWS 主題中適合您使用者類型的登入程序操作。
2. 在 主控台首頁 頁面上選取 IAM 服務。
3. 在導覽窗格中選取 使用者，然後選取 新增使用者。

   > **Note**
   >
   > 如果您已啟用 IAM 身分中心，則 AWS Management Console 會顯示提醒您最好在 IAM 身分中心管理使用者存取權限。在本教學課程中，您建立的 IAM 使用者僅會在 IAM Identity Center 憑證中的使用者無法使用時才會派上用場。

4. 在 User name (使用者名稱) 中輸入 **`EmergencyAccess`**。名稱不可含有空格。
5. 選取 [提供使用者存取 AWS Management Console— 選用] 旁邊的核取方塊，然後選擇 [我要建立 IAM 使用者]。
6. 在 主控台密碼 下選取 自動產生的密碼。
7. 清除 使用者必須在下次登入時建立新密碼 (建議) 旁的核取方塊。由於此 IAM 使用者是用於緊急存取，因此受信任的管理員會保留密碼，並僅在必要時提供密碼。
8. 在 設定許可 頁面的 許可選項 下方選取 新增使用者至群組。然後，在 使用者群組 下方選取 建立群組。
9. 在 建立使用者群組 頁面的 使用者群組名稱 中輸入**`EmergencyAccessGroup`**。然後，在 [權限原則] 下，選取AdministratorAccess。
10. 選取 建立使用者群組 以返回 設定許可 頁面。
11. 在 使用者群組 下方選取先前建立的 **`EmergencyAccessGroup`** 的名稱。
12. 選取 下一步 以繼續前往 檢閱和建立 頁面。
13. 在 檢閱和建立 頁面上檢閱要新增至新使用者的使用者群組成員資格清單。準備好繼續時，請選取 建立使用者。

14. 在 擷取密碼 頁面上選取 下載 .csv 檔案 以儲存含有使用者憑證資訊 (連線 URL、使用者名稱和密碼) 的 .csv 檔案。
15. 如果您需要登入 IAM 且無法存取聯合身分供應商，請儲存此檔案以供使用。

新的 IAM 使用者會顯示在 使用者 清單中。選取 使用者名稱 連結以檢視使用者詳細資訊。在 摘要 下方將使用者的 ARN 複製到剪貼簿。將 ARN 貼到文字文件中，供下一個程序使用。

建立第一個角色

IAM 角色是為您信任的實體授予許可的安全方式。IAM 角色與 IAM 使用者有些相似處。角色和使用者都是具備許可政策的主體，可決定身分在 AWS中可執行和不可執行的操作。但是，角色的目的是讓需要它的任何人可代入，而不是單獨地與某個人員關聯。此外，角色沒有與之關聯的標準長期憑證，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。使用角色有助於您遵循 IAM 最佳實務。您可以使用角色執行以下動作：

- 啟用員工身分識別和啟用身分識別中心的應用程式存取 AWS Management Console 使用 AWS IAM Identity Center。
- 將權限委派給 AWS 服務，以代表您執行動作。
- 啟用在 Amazon EC2 執行個體上執行應用程式的程式碼，以存取或修改 AWS 資源。
- 授予存取權給另一個人 AWS 帳戶。

> **Note**
>
> 您可以使用「隨處 AWS Identity and Access Management 角色」來授予機器身分識別的存取權。在任何地方使用 IAM 角色意味著您不需要管理在外部執行的工作負載的長期登入資料 AWS。如需詳細資訊，請參閱《AWS Identity and Access Management Roles Anywhere 使用者指南》中的 What is AWS Identity and Access Management Roles Anywhere? (什麼是 Roles Anywhere？)。

IAM 身分中心和其他 AWS 服務會自動為其服務建立角色。如果您使用的是 IAM 使用者，建議您為使用者建立角色，以便對方在登入後擔任該角色。這樣一來，他們便會在工作階段期間獲得臨時許可，而不是長期許可。

根據您是為 IAM 使用者、 AWS 服務還是聯合身分使用者建立角色， AWS Management Console 精靈會引導您完成建立角色的步驟，所顯示的步驟略有不同。應使用聯合存取來提供組織 AWS 帳戶 內

的定期存取權限。如果您出於特定目的 (例如緊急存取或以程式設計方式存取) 而建立 IAM 使用者，請僅授予這些 IAM 使用者擔任角色的權限，並將這些 IAM 使用者放入角色專屬群組。

在此程序中，您會建立為 EmergencyAccess IAM 使用者提供 SupportUser 存取權的角色。開始此程序前，請先將 IAM 使用者的 ARN 複製到剪貼簿。

為 IAM 使用者建立角色

1. 按照《AWS 登入使用者指南》 如何登入 AWS 主題中適合您使用者類型的登入程序操作。
2. 在 主控台首頁 頁面上選取 IAM 服務。
3. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
4. 選擇 AWS 帳戶 角色類型。
5. 在 選取信任的實體 的 信任的實體類型 下方選擇 自訂信任政策。
6. 在 自訂信任政策 區段中檢閱基本信任政策。這是我們將用於此角色的政策。使用 編輯陳述式 編輯器來更新信任政策：

   1. 在 為 STS 新增動作 中選取 擔任角色。
   2. 選取 新增主體 旁的 新增。新增主體 視窗即會開啟。

      在 主體類型 下方選取 IAM 使用者。

      在 ARN 下方將您複製的 IAM 使用者 ARN 貼到剪貼簿。

      選取 新增主體。
   3. 確認信任政策中的 `Principal` 行目前含有您指定的 ARN：

      ```
      "Principal": { "AWS": "arn:aws:iam::123456789012:user/username" }
      ```
7. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。
8. 在 新增許可 中選取要套用的許可政策旁的核取方塊。在本教程中，我們將選擇SupportUser信任策略。然後，您可以使用此角色來疑難排解和解決 AWS 帳戶 與開啟的支援案例相關的問題 AWS。我們目前不會設定 許可界限。
9. 選擇下一步。
10. 在 命名、檢閱和建立 中完成下列設定：

    - 在角色名稱中，輸入可識別此角色的名稱，例如SupportUserRole。
    - 在 描述 中說明角色的預定用途。

由於其他 AWS 資源可能會參照該角色，因此您無法在建立角色之後編輯該角色的名稱。

11. 選取 建立角色。

    建立角色後，請與需要該角色的人員分享角色資訊。您可以透過下列方式分享角色資訊：

    - 角色連結：向使用者傳送連結，讓他們進入已填寫所有詳細資訊的 Switch Role (切換角色) 頁面。
    - 帳戶 ID 或別名：為每個使用者提供角色名稱以及帳戶 ID 號碼或帳戶別名。使用者接著前往 Switch Role (切換角色) 頁面，然後手動新增詳細資訊。
    - 儲存角色連結資訊以及 EmergencyAccess 使用者認證。

    如需詳細資訊，請參閱 提供資訊給使用者。

建立第一項 IAM 政策

IAM 政策會連接到 IAM 身分 (使用者、使用者群組或角色) 或 AWS 資源。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。

建立第一項 IAM 政策

1. 按照《AWS 登入使用者指南》 如何登入 AWS 主題中適合您使用者類型的登入程序操作。
2. 在 主控台首頁 頁面上選取 IAM 服務。
3. 在導覽窗格中，選擇政策。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
4. 選擇 Create policy (建立政策)。
5. 在建立政策頁面上，選擇動作，然後選擇匯入政策。
6. 在匯入政策視窗中的尋找政策方塊中輸入 **`power`**，以精簡政策清單。選取PowerUserAccess策略。
7. 選取匯入政策。政策會顯示在 JSON 標籤中。
8. 選擇下一步。

9. 在檢視與建立頁面上，在政策名稱欄位中輸入 **PowerUserExamplePolicy**。針對 Description (描述)，輸入 **Allows full access to all services except those for user management**。然後選擇 建立政策 來儲存政策。

您可以將此政策連接至角色，以便為擔任該角色的使用者提供與此政策相關聯的許可。
該PowerUserAccess策略通常用於為開發人員提供訪問權限。

程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於存取的使用者類型：

- 如果您在 IAM 身分中心管理身分識別， AWS API 需要設定檔，且 AWS Command Line Interface 需要設定檔或環境變數。
- 如果您有 IAM 使用者，則 AWS API 和 AWS Command Line Interface 需要存取金鑰。如有可能，請建立臨時憑證，其中包含存取金鑰 ID、私密存取金鑰，以及指出憑證何時到期的安全記號。

若要授予使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用短期認證簽署 AWS CLI 或 AWS API 的程式設計要求 (直接或使用 AWS SDK)。	請依照您要使用的介面遵循指示： • 對於 AWS CLI，請遵循AWS IAM Identity Center 使用者指南中獲取用於 CLI 存取的 IAM 角色登入資料中的說明進行操作。 • 對於 AWS API，請按照 AWS SDK 和工具參考指南中 SSO 認證中的說明進行操作。

哪個使用者需要程式設計存取權？	到	By
IAM	使用短期認證簽署 AWS CLI 或 AWS API 的程式設計要求 (直接或使用 AWS SDK)。	遵循使用臨時認證與 AWS 資源中的說明進行操作。
IAM	使用長期認證來簽署 AWS CLI 或 AWS API 的程式設計要求 (直接或使用 AWS SDK)。 (不建議使用)	請遵循 為 IAM 使用者管理存取金鑰 中的指示操作。

AWS Identity and Access Management中的安全最佳實務和使用案例

AWS Identity and Access Management (IAM) 在開發和實作自己的安全政策時，提供許多安全性功能供您考慮。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

若要取得 IAM 的最大好處，請花些時間了解所建議的最佳實務。其中一個方法是，了解 IAM 搭配其他 AWS 服務用於真實世界的案例。

主題

IAM 中的安全最佳實務

Follow us on Twitter

最 AWS Identity and Access Management 佳做法已於 2022 年 7 月 14 日更新。

為了協助保護您的 AWS 資源，請遵循下列 AWS Identity and Access Management (IAM) 的最佳做法。

主題

- 使用 IAM Access Analyzer 根據存取活動產生最低權限政策
- 定期檢閱並移除未使用的使用者、角色、許可、政策和憑證
- 使用 IAM 政策中的條件進一步限制存取權
- 使用 IAM Access Analyzer 驗證對資源的公開與跨帳戶存取權
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作
- 建立跨多個帳戶的許可防護機制
- 使用許可界限委派帳戶內的許可管理

要求人類使用者使用與身分識別提供者的同盟，才能 AWS 使用臨時登入資料

人類使用者具有人類身分，是應用程式的相關人員、管理員、開發人員、操作員和消費者。他們必須具有身份才能訪問您的 AWS 環境和應用程序。擁有您組織成員身分的人類使用者，也具有人力身分。人類使用者也可以是您與之協作並與您的 AWS 資源互動的外部使用者。他們也可以透過 Web 瀏覽器、用戶端應用程式、行動應用程式或互動式命令列工具進行此項工作。

要求您的人類使用者在存取時使用臨時登入資料 AWS。您可以為您的人類使用者使用身分識別提供者， AWS 帳戶 藉由假設角色 (提供臨時認證) 來提供聯合存取權。如果是集中式存取管理，我們建議您使用 AWS IAM Identity Center (IAM Identity Center) 管理您帳戶的存取權和這些帳戶內的許可。您可以使用 IAM Identity Center 管理使用者身分，或從外部身分提供者管理 中使用者身分的存取許可。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 AWS IAM Identity Center。

如需角色的詳細資訊，請參閱角色術語和概念。

要求工作負載使用臨時登入資料搭配 IAM 角色來存取 AWS

工作負載是可提供商業價值的資源和程式碼的集合，例如應用程式或後端程序。您的工作負載會有需要身分才能向 AWS 服務提出請求 (例如請求讀取資料) 的應用程式、操作工具和元件。這些身分包括在您的 AWS 環境中執行的機器，例如 Amazon EC2 執行個體或 AWS Lambda 函數。

您可以管理需要存取權的外部當事人的機器身分。若要將存取權授予機器身分，您可以使用 IAM 角色。IAM 角色具有特定許可，並透 AWS 過仰賴具有角色工作階段的臨時安全登入資料，提供存取方式。此外，您可能有其他電腦需 AWS 要存取您的 AWS 環境。對於在您以外執行的電腦，可 AWS 以使用任何地方的AWS Identity and Access Management 角色。如需角色的詳細資訊，請參閱IAM 角色。如需如何使用角色委派存取權的詳細資訊 AWS 帳戶，請參閱IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶。

需要多重要素驗證 (MFA)

我們建議將 IAM 角色用於會存取您 AWS 資源的人類使用者和工作負載，以便他們使用暫時性憑證。但是，對於帳戶中需要 IAM 使用者或根使用者的情況，則需要 MFA 提供額外的安全性。使用 MFA，使用者便可擁有一個裝置，針對身分驗證查問產生回應。擁有每位使用者的憑證及裝置產生的回應，才能完成登入程序。如需詳細資訊，請參閱 在中使用多因素身份驗證（MFA）AWS。

如果您使用 IAM 身分中心為人類使用者進行集中式存取管理，則當您的身分識別來源設定為 IAM 身分中心身分存放區、 AWS 受管 Microsoft AD 或 AD Connector 時，可以使用 IAM 身分中心 MFA 功能。如需有關 IAM Identity Center 中的 MFA 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證。

對於需要長期憑證的使用案例，請視需要更新存取金鑰

我們建議您盡可能依賴暫時性憑證，而不要建立長期憑證，例如存取金鑰。但是，對於需要具有程式化存取和長期憑證的 IAM 使用者的情況，我們建議您視需要更新存取金鑰，如在員工離職時。我們建議您使用 IAM 存取上次使用的資訊，以便安全地更新和移除存取金鑰。如需詳細資訊，請參閱 更新存取金鑰。

某些特定使用案例需要長期憑證和 AWS中的 IAM 使用者。以下是部分使用案例：

- 無法使用 IAM 角色的工作負載 - 您可以從需要存取 AWS的位置執行工作負載。在某些情況下，您無法使用 IAM 角色提供臨時登入資料，例如 WordPress 外掛程式。在這些情況下，請將 IAM 使用者長期存取金鑰用於該工作負載，對 AWS進行身分驗證。
- 第三方用 AWS 戶端 — 如果您使用的工具不支援 IAM 身分中心的存取權，例如第三方用 AWS 戶端或未託管的廠商 AWS，請使用 IAM 使用者長期存取金鑰。
- AWS CodeCommit 存取 — 如果您使 CodeCommit 用儲存程式碼，您可以使用具有 SSH 金鑰或服務特定登入資料的 IAM 使用者，對儲存庫 CodeCommit 進行驗證。除了將 IAM Identity Center 中的使用者用於一般身分驗證之外，我們也建議您這樣做。IAM 身分中心的使用者是員工中需要存取您的雲端應用程式 AWS 帳戶 或雲端應用程式的人員。若要讓使用者在不設定 IAM 使用者的情況下 CodeCommit 存取您的儲存庫，您可以設定git-remote-codecommit公用程式。如需 IAM 和的詳細資訊 CodeCommit，請參閱將 IAM 搭配使用 CodeCommit：Git 登入資料、安全殼層金鑰和 AWS 存取金鑰。如需有關設定公用git-remote-codecommit程式的詳細資訊，請參閱《AWS CodeCommit 使用指南》中的使用旋轉認證連線到 AWS CodeCommit 儲存庫。
- Amazon Keyspaces (適用於 Apache Cassandra) 存取 – 在無法使用 IAM Identity Center 中的使用者的情況下，例如為了測試 Cassandra 相容性，您可以將 IAM 使用者搭配服務特定憑證使用，以便使用 Amazon Keyspaces 進行身分驗證。IAM 身分中心的使用者是員工中需要存取您的雲端應用程式

AWS 帳戶 或雲端應用程式的人員。您也可以使用暫時性憑證連線到 Amazon Keyspaces。如需詳細資訊，請參閱《Amazon Keyspaces (適用於 Apache Cassandra) 開發人員指南》中的透過 IAM 角色和 SigV4 外掛程式使用暫時性憑證連線到 Amazon Keyspaces。

遵循最佳實務以保護您的根使用者憑證

當您建立時 AWS 帳戶，您會建立根使用者認證以登入 AWS Management Console. 如同保護其他敏感的個人資訊那樣，保護您的根使用者憑證。若要更好地了解如何保護和擴展根使用者程序，請參閱 適用於 AWS 帳戶的根使用者最佳實務。

套用最低權限許可

設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。探索工作負載或使用案例所需的許可時，您可能會從廣泛許可開始。隨著使用案例的成熟，您可以設法減少授予的許可，以便朝向最低權限的目標邁進。如需使用 IAM 套用許可的詳細資訊，請參閱 IAM 中的政策和許可。

開始使用 AWS 受管理的原則，並邁向最低權限權限

若要開始授予許可給使用者和工作負載，請使用 AWS 受管政策，為許多常見使用案例授予許可。它們可用在您的 AWS 帳戶. 請記住， AWS 受管理的政策可能不會針對您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。因此，我們建議您定義使用案例的客戶管理政策，以便進一步減少許可。如需詳細資訊，請參閱 AWS 受管理政策。如需針對特定工作職能所設計之 AWS 受管理原則的詳細資訊，請參閱AWS 受管理的工作職能政策。

使用 IAM Access Analyzer 根據存取活動產生最低權限政策

若只授予執行任務所需的許可，您可以根據在 AWS CloudTrail中記錄的存取活動產生政策。IAM Access Analyzer 會分析您 IAM 角色使用的服務和動作，然後產生您可以使用的精細政策。測試產生的每個政策後，您可以將政策部署到生產環境。這可確保您僅授予所需的許可給工作負載。如需政策產生的詳細資訊，請參閱 IAM Access Analyzer 政策產生。

定期檢閱並移除未使用的使用者、角色、許可、政策和憑證

您的 AWS 帳戶中可能存在不再需要的 IAM 使用者、角色、許可、政策或憑證。IAM 會提供上次存取的資訊，協助您識別不再需要的使用者、角色、許可、政策和憑證，以便您移除這些資料。這可協助您減少必須監控的使用者、角色、許可、政策和憑證數量。您也可以使用此資訊來精簡 IAM 政策，以便更完善地遵循最低權限許可。如需詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

使用 IAM 政策中的條件進一步限制存取權

您可以根據生效的政策陳述式指定條件。如此，您便可以授予對動作和資源的存取權，前提是存取請求符合特定條件。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與服務動作的存取權，但前提是透過特定 AWS 服務動作 (例如) 使用這些動作 AWS CloudFormation。如需詳細資訊，請參閱 IAM JSON 政策元素：Condition。

使用 IAM Access Analyzer 驗證對資源的公開與跨帳戶存取權

在中授與公用或跨帳戶存取權限之前 AWS，建議您先確認是否需要此類存取權。您可以使用 IAM Access Analyzer，協助自己預覽和分析所支援資源類型的公開和跨帳戶存取權。為此，您可以檢閱 IAM Access Analyzer 產生的問題清單。這些問題清單有助於您驗證資源存取控制是否授予您預期的存取權。此外，隨著您更新公開和跨帳戶許可，您可以在將新的存取控制部署到資源前驗證變更的影響。IAM Access Analyzer 也會持續監控支援的資源類型，並為允許公開或跨帳戶存取權的資源產生問題清單。如需詳細資訊，請參閱使用 IAM Access Analyzer API 預覽存取。

使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作

驗證您建立的政策，確保它們遵守IAM 政策語言 (JSON) 和 IAM 最佳實務。您可以使用 IAM Access Analyzer 政策驗證來驗證自己的政策。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。您在主控台中編寫新政策或編輯現有政策時，IAM Access Analyzer 會提供建議，協助您在儲存政策前完善和驗證政策。此外，我們建議您檢閱並驗證所有現有政策。如需詳細資訊，請參閱 IAM Access Analyzer 政策驗證。如需 IAM Access Analyzer 所提供政策檢查的詳細資訊，請參閱《IAM Access Analyzer 政策檢查參考》。

建立跨多個帳戶的許可防護機制

當您擴展工作負載時，請使用管理的多個帳戶將它們分開 AWS Organizations。我們建議您使用 Organizations 服務控制政策 (SCP) 建立許可防護機制，以便控制您帳戶之間所有 IAM 使用者和角色的存取權。SCP 是一種組織原則類型，可用來在組織、OU 或帳戶層級管理 AWS 組織中的權限。您建立的許可防護機制會套用至所涵蓋帳戶內的所有使用者和角色。不過，僅有 SCP 並不足以授予許可給您組織中的帳戶。若要實現此目的，管理員仍必須將身分型或資源型政策連接到 IAM 使用者、IAM 角色或您帳戶中的資源。如需詳細資訊，請參閱 AWS Organizations、帳戶和 IAM 防護機制。

使用許可界限委派帳戶內的許可管理

在某些情況下，您可能想要將帳戶內的許可管理委派給其他人。例如，您可以允許開發人員為其工作負載建立和管理角色。委派許可給其他人時，請使用許可界限來設定您委派許可的上限。許可界限是一種

進階功能，可供您使用受管政策來設定身分型政策可以授予 IAM 角色的許可上限。許可界限不會自行授予許可。如需更多詳細資訊，請參閱 IAM 實體的許可界限。

適用於 AWS 帳戶的根使用者最佳實務

當您第一次建立時 AWS 帳戶，您會從一組預設的認證開始，並可完整存取您帳戶中的所有 AWS 資源。此身分稱為 AWS 帳戶 根使用者。強烈建議您不要存取 AWS 帳戶 root 使用者，除非您有需要 root 使用者認證的工作。您需要保護根使用者憑證和帳戶復原機制，以協助確保不會公開具有高權限的憑證以供未經授權使用。

並非存取根使用者，而是建立日常任務的系統管理使用者。

- 若要取得單一獨立的 AWS 帳戶，請參閱〈〉建立具有管理權限的使用者。
- 如需透過 AWS 帳戶 管理的多個人資訊 AWS Organizations，請參閱為 AWS 帳戶 IAM 身分中心管理使用者設定存取權限。

然後，您可以使用管理使用者為需要存取 AWS 帳戶中資源的使用者建立其他身分。強烈建議您要求使用者在存取時使用臨時登入資料進行驗證 AWS。

- 對於單一、獨立 AWS 帳戶，使用IAM 角色在您的帳戶中建立具有特定權限的身分識別。角色旨在可由任何需要它的使用者擔任。此外，角色沒有與之關聯的標準長期憑證，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。與 IAM 角色不同，IAM 使用者 擁有長期憑證，例如密碼和存取金鑰。最佳實務建議盡可能依賴暫時憑證，而不是建立擁有長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。
- 對於透過 Organizations AWS 帳戶 管理的多個人，請使用 IAM 身分中心人力使用者。透過 IAM 身分中心，您可以集中管理各個帳戶的使用者 AWS 帳戶 和這些帳戶內的許可。可使用 IAM Identity Center 或外部身分供應商來管理使用者身分。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 AWS IAM Identity Center。

主題

- 保護您的根使用者憑證，以防止未經授權使用
- 使用強式根使用者密碼來協助保護存取
- 使用多重要素驗證 (MFA) 保護您的根使用者登入
- 不要為根使用者建立存取金鑰
- 盡可能為根使用者登入使用多人核准

- 使用群組電子郵件地址作為根使用者憑證
- 限制存取帳戶復原機制
- 保護您的 Organizations 帳戶根使用者憑證
- 監控存取和用量

保護您的根使用者憑證，以防止未經授權使用

保護您的根使用者憑證，僅將其用於需要它們的任務。為了防止未經授權的使用，請勿與任何人共用您的 root 使用者密碼、MFA、 CloudFront 存取金鑰、金鑰配對或簽署憑證，除非具有嚴格業務需求才能存取 root 使用者的憑證。

不要將 root 用戶密碼與依賴於使用相同密碼訪問的帳戶 AWS 服務 中的工具存儲在一起。如果您遺失或忘記 root 使用者密碼，將無法存取這些工具。我們建議您優先考慮彈性，並考慮要求兩個或更多人授權存取儲存位置。應記錄並監控任何對密碼的存取或其儲存位置。

使用強式根使用者密碼來協助保護存取

我們建議您使用高強度且唯一的密碼。具有強密碼產生演算法的密碼管理器之類的工具可協助您實現這些目標。 AWS 要求您的密碼必須符合下列條件：

- 它必須至少有 8 個字元，最多 128 個字元。
- 它至少混用 3 種下列類型字元：大寫、小寫、數字和 ! @ # $ % ^ & * () <> [] {} | _+-= 符號。
- 它不得與您的 AWS 帳戶 姓名或電子郵件地址相同。

如需詳細資訊，請參閱 更改密碼 AWS 帳戶根使用者。

使用多重要素驗證 (MFA) 保護您的根使用者登入

因為根使用者可以執行特權動作，因此除了將電子郵件地址和密碼作為登入憑證之外，還要將根使用者的 MFA 新增為第二個身分驗證因素。我們強烈建議您為根使用者憑證啟用多個 MFA，以在安全性策略中提供額外的靈活性和彈性。您可以以目前受支援 MFA 類型的任意組合為您的 AWS 帳戶 根使用者註冊最多八台 MFA 裝置。

- 由第三方供應商提供經 FIDO 認證的硬體安全金鑰。如需詳細資訊，請參閱啟用 AWS 帳戶 根使用者的 FIDO 安全性金鑰。
- 一種在以時間為基礎的一次性密碼 (TOTP) 演算法的基礎上產生六位數字程式碼的硬體裝置。如需詳細資訊，請參閱為 AWS 帳戶 root 使用者啟用硬體 TOTP 權杖。

- 在手機或其他裝置上執行並模擬實體裝置的虛擬驗證器應用程式。如需詳細資訊，請參閱為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置。

不要為根使用者建立存取金鑰

存取金鑰可讓您在 AWS 命令列介面 (AWS CLI) 中執行命令，或從其中一個 AWS SDK 使用 API 作業。強烈建議您不要為 root 使用者建立存取金鑰配對，因為 root 使用者擁有帳戶中所有 AWS 服務 資源的完整存取權，包括帳單資訊。

由於只有少數工作需要 root 使用者，而且您通常不常執行這些工作，因此建議您登入以執行 root 使用者工作。 AWS Management Console 在建立存取金鑰前，請檢閱長期存取金鑰的替代方案。

盡可能為根使用者登入使用多人核准

請考慮使用多人核准，以確保沒有任何人可以同時存取根使用者的 MFA 和密碼。有些公司會設定可存取密碼的管理員群組，以及另一個可存取 MFA 的管理員群組，藉此增加額外的安全性層級。每個群組必須各有一位成員共同作為根使用者登入。

使用群組電子郵件地址作為根使用者憑證

使用由企業管理的電子郵件地址，直接將收到的訊息轉寄給使用者群組。如果 AWS 必須聯繫帳戶的所有者，這種方法可以降低響應延遲的風險，即使個人休假，生病或已經離開了業務。用於根使用者的電子郵件地址不得用於其他目的。

限制存取帳戶復原機制

務必制定程序來管理根使用者憑證復原機制，以防在緊急情況下需要存取該機制，例如接管您的管理帳戶。

- 確保您有權存取根使用者電子郵件收件匣，以便可以重設遺失或忘記的根使用者密碼。
- 如果 AWS 帳戶 根使用者的 MFA 遺失、損壞或無法運作，您可以使用向相同根使用者認證註冊的另一個 MFA 登入。如果您無法訪問所有 MFA，則需要註冊帳戶時使用的電話號碼和電子郵件地址，以保持最新狀態並可以訪問以恢復您的 MFA。如需詳細資訊，請參閱復原根使用者 MFA 裝置。
- 如果您選擇不儲存根使用者密碼和 MFA，則可以使用帳戶中註冊的電話號碼作為復原根使用者憑證的替代方法。確保您可以存取聯絡人電話號碼，保持更新此電話號碼，並且限制可以存取以管理電話號碼的使用者。

應沒有任何人有權同時存取電子郵件收件匣和電話號碼，因為兩者都是復原根使用者密碼的驗證渠道。務必安排兩組人員管理這些渠道。一個群組可以存取您的主要電子郵件地址，另一個群組可以存取主要電話號碼，從而復原以根使用者身分存取您帳戶的許可。

保護您的 Organizations 帳戶根使用者憑證

當您使用 Organizations 轉向多帳戶策略時，您的每個人都 AWS 帳戶 有自己的 root 使用者憑證，以確保您需要保護這些憑證。您用來建立組織的帳戶是管理帳戶，組織中的其餘帳戶則為成員帳戶。

保護成員帳戶的根使用者憑證

如果您使用 Organizations 來管理多個帳戶，可以採取兩種策略來保護 Organizations 中的根使用者存取權。

- 使用 MFA 保護 Organizations 帳戶的根使用者憑證。
- 請勿為您的帳戶重設根使用者密碼，只有在需要時才使用密碼重設程序復原存取權限。當您在組織中建立成員帳戶時，Organizations 會自動在成員帳戶中建立 IAM 角色，該角色允許管理帳戶暫時存取成員帳戶。

如需詳細資訊，請參閱 Organizations 使用者指南中的存取組織中的成員帳戶。

使用服務控制政策 (SCP) 設定 Organizations 中的預防安全控制措施

如果您使用 Organizations 管理多個帳戶，可以套用 SCP 來限制對成員帳戶根使用者的存取。拒絕成員帳戶中的所有根使用者動作 (除了某些僅限根的動作)，這樣有助於防止未經授權的存取。有關詳細資訊，請參閱使用 SCP 來限制成員帳戶中根使用者可執行的動作。

監控存取和用量

我們建議您使用目前的追蹤機制來監控、警示和報告根使用者憑證的登入和使用情況，包括宣告根使用者登入和用量的警示。下列服務可協助確保追蹤根使用者憑證用量，並且執行安全性檢查，以防止未經授權的使用。

- 如果您希望收到有關帳戶中根使用者登入活動的通知，可以利用 Amazon CloudWatch 建立事件規則，以偵測何時使用 root 使用者登入資料，並向安全管理員觸發通知。如需詳細資訊，請參閱監控 AWS 帳戶 根使用者活動並通知。
- 如果您想要設定通知以提醒您核准的根使用者動作，可以利用 Amazon EventBridge 和 Amazon SNS 撰寫 EventBridge 規則來追蹤特定動作的根使用者使用情況，並使用 Amazon SNS 主題通知您。如需範例，請參閱建立 Amazon S3 物件時傳送通知。

- 如果您已經用 GuardDuty 作威脅偵測服務，您可以擴充其功能，以便在帳戶中使用 root 使用者認證時通知您。

此警示應包含但不得限於根使用者的電子郵件地址。制定如何回應警示的適當程序，從而收到根使用者存取警示的人員可了解如何驗證預期的根使用者存取權，以及如果他們認為安全事件正在進行中該如何上報。有關如何設定警示的範例，請參閱監控和通知 AWS 帳戶 根使用者活動。

評估根使用者 MFA 合規

- AWS Config 使用規則來協助強制執行 root 使用者的最佳作法。您可以使用 AWS 受管規則來要求 root 使用者啟用多因素驗證 (MFA)。 AWS Config 也可以識別 root 使用者的存取金鑰。
- Security Hub 為您提供中安全性狀態的全面檢視， AWS 並協助您根據安全性產業標準和最佳做法來評估您的 AWS 環境，例如針對 root 使用者擁有 MFA，而且沒有 root 使用者存取金鑰。如需有關可用規則的詳細資訊，請參閱 Security Hub 使用者指南中的 AWS Identity and Access Management 控制項。
- Trusted Advisor 提供安全性檢查，以便您知道根使用者帳戶是否未啟用 MFA。如需詳細資訊，請參閱 AWS 支援使用者指南中的根帳戶上的 MFA。

如果您需要回報帳戶的安全性問題，請參閱回報可疑電子郵件或弱點回報。或者，您可以聯絡 AWS 以取得協助和其他指導。

適用於 IAM 的商業使用案例

IAM 的簡單業務使用案例可協助您了解實作服務的基本方式，以控制使用者擁有的 AWS 存取權。使用案例是在一般條款中說明，並不包含您如何使用 IAM API 來達到您想要的結果的機制。

此使用案例示範稱為 Example Corp 的虛構公司可以兩種典型的方式使用 IAM。第一個案例會考慮 Amazon Elastic Compute Cloud (Amazon EC2)。第二個則考慮 Amazon Simple Storage Service (Amazon S3)。

如需將 IAM 與其他服務搭配使用的詳細資訊 AWS，請參閱AWS 與 IAM 搭配使用的服務。

主題

範例企業的初始設定

尼克·沃爾夫和馬特奧·傑克遜是實例公司的創始人在創辦公司後，他們創建了一個 AWS 帳戶 並建立了 AWS IAM Identity Center（IAM 身份中心）來創建與他們的資源一起使用的管理帳戶。 AWS 當您為管理使用者設定帳戶存取權時，IAM Identity Center 會建立對應的 IAM 角色。此角色由 IAM 身分中心控制，會在相關資訊中建立 AWS 帳戶，並將AdministratorAccess權限集中指定的政策附加至該角色。

因為他們現在有管理員帳戶，Nikki 和 Mateo 不再需要使用其根使用者來存取他們的 AWS 帳戶。他們計劃僅將根使用者用於完成只能由根使用者執行的任務。在檢閱安全最佳實務之後，他們要為根使用者憑證設定多重要素驗證 (MFA)，並決定如何保護其根使用者憑證的安全。

隨著公司的發展，他們聘請員工擔任開發人員、管理員、測試人員、經理和系統管理員。Nikki 負責營運，而 Mateo 管理工程團隊。他們要設定 Active Directory Domain Server 來管理員工帳戶並管理對內部公司資源的存取權。

為了讓員工能夠存取 AWS 資源，他們使用 IAM 身分中心將公司的 Active Directory 連線到他們的身分識別中心 AWS 帳戶。

因為他們將 Active Directory 連接到 IAM Identity Center，使用者、群組和群組成員資格將會被同步並且得到定義。他們必須將權限集和角色指派給不同的群組，以便為使用者提供 AWS 資源的正確存取層級。他們使用AWS 受管理的工作職能政策中 AWS Management Console 建立下列權限集：

- 管理員
- 帳單
- 開發人員
- 網路管理員
- 資料庫管理員
- 系統管理員
- 支援使用者

然後，他們要指派這些許可集合到被指派給其 Active Directory 群組的角色。

如需 IAM 身分中心初始設定的 step-by-step 指南，請參閱AWS IAM Identity Center 使用者指南中的入門指南。如需有關佈建 IAM Identity Center 使用者存取權的詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的單一登入存取 AWS 帳戶。

搭配 Amazon EC2 的 IAM 使用案例

與 Example Corp 類似的公司通常會使用 IAM 與像是 Amazon EC2 的服務互動。若要了解此部分的使用案例，您需要對 Amazon EC2 有基本的理解。如需有關 Amazon EC2 的詳細資訊，請參閱 Amazon EC2 使用者指南。

使用者群組的 Amazon EC2 權限

若要提供「周邊」控制項，Nikki 會將原則附加至 AllUsers 使用者群組。如果原始 IP 位址位於 Example Corp 的公司網路之外，此原則會拒絕使用者的任何 AWS 要求。

在 Example Corp，不同使用者群組需要不同的許可：

- System administrators (系統管理員) – 需要許可才能建立及管理受管 AMI、執行個體、快照、磁碟區、安全群組等。Nikki 會將受`AmazonEC2FullAccess` AWS 管政策附加到使用 SysAdmins 者群組，讓群組成員可以使用所有 Amazon EC2 動作。
- Developers (開發人員) – 僅需要使用執行個體的能力。Nikki 因此建立政策並將其連接到 Developers 使用者群組，讓開發人員能夠呼叫 `DescribeInstances`、`RunInstances`、`StopInstances`、`StartInstances` 和 `TerminateInstances`。

  > **Note**
  >
  > Amazon EC2 使用 SSH 金鑰、Windows 密碼及安全群組控制擁有特定 Amazon EC2 執行個體作業系統存取權限的對象。IAM 系統中沒有方法，可允許或拒絕存取特定執行個體的作業系統。

- Support 使用者 – 除了列出目前可用的 Amazon EC2 資源之外，不應擁有執行任何 Amazon EC2 動作的許可。因此，Nikki 建立政策並將其連接到 Support 使用者群組，只讓他們呼叫 Amazon EC2「描述」API 操作。

如需這些政策可能外觀的範例，請參閱 Amazon EC2 使用者指南中的以身分為基礎的 IAM 政策範例和使用 AWS 身分和存取管理。

使用者的任務職能變更

在某個時間點，其中一名開發人員 Paulo Santos 變更了工作職能成為經理。身為經理，Paulo 成為 Support 使用者群組的一員，因此他可以為其開發人員開啟支援案例。Mateo 將 Paulo 從 Developers

使用者群組移到 Support 使用者群組。其結果是，他與 Amazon EC2 執行個體互動的能力受到限制。他無法啟動或開始執行個體。他也無法停止或終止現有的執行個體，即使他是啟動或開始執行個體的使用者。他只能列出 Example Corp 使用者啟動的執行個體。

搭配 Amazon S3 的 IAM 使用案例

與 Example Corp 相似的公司通常也會搭配 Amazon S3 使用 IAM。John 已為公司建立名為 aws-s3-bucket 的 Amazon S3 儲存貯體。

建立其他使用者和使用者群組

員工 Zhang Wei 和 Mary Major 每個都需要能夠在公司的儲存貯體中建立自己的資料。他們也需要讀取和寫入所有開發人員共用的資料。為啟用此功能，Mateo 依據邏輯使用 Amazon S3 金鑰字首 (key prefix) 機制在 aws-s3-bucket 中安排資料，如下圖所示。

```
/aws-s3-bucket
    /home
        /zhang
        /major
    /share
        /developers
        /managers
```

Mateo 為每個員工將 `/aws-s3-bucket` 區分為一組主目錄，以及為開發人員和管理員的群組一個共用的區域。

現在 Mateo 建立一組政策以指派許可給使用者和使用者群組：

- Zhang 的主目錄存取權 – Mateo 將政策連接到 Wei，讓他可以讀取、寫入及列出具有 Amazon S3 金鑰字首 (key prefix) `/aws-s3-bucket/home/zhang/` 的任何物件
- Major 的主目錄存取權 – Mateo 將政策連接到 Mary，讓她可以讀取、寫入及列出具有 Amazon S3 金鑰字首 (key prefix) `/aws-s3-bucket/home/major/` 的任何物件
- 開發人員使用者群組的共享目錄存取權 – Mateo 將政策連接到使用者群組，讓開發人員可以讀取、寫入及列出 `/aws-s3-bucket/share/developers/` 中的任何物件
- 管理員群組的共享目錄存取權 – Mateo 將政策連接到使用者群組，可讓管理員讀取、寫入及列出 `/aws-s3-bucket/share/managers/` 中的物件

> **Note**
>
> Amazon S3 不會自動授予建立儲存貯體或物件的使用者，在該儲存貯體或物件上執行其他動作的許可。因此，在您的 IAM 政策中，您必須明確地提供使用者許可，以使用他們建立的 Amazon S3 資源。

如需這些政策的範例，請參閱 Amazon Simple Storage Service 使用者指南中的存取控制。如需執行時間期間評估政策方式的資訊，請參閱政策評估邏輯。

使用者的任務職能變更

在某個時間點，其中一名開發人員 Zhang Wei 變更了工作職能成為經理。我們假設他不再需要存取 `share/developers` 目錄中的文件。Mateo 身為管理員，將 Wei 移到 `Managers` 使用者群組並移出 `Developers` 使用者群組。只需簡單的重新指派，Wei 就會自動取得授予到 `Managers` 使用者群組的所有許可，但不再存取 `share/developers` 目錄中的資料。

與第三方企業整合

組織通常會與合作夥伴公司、顧問和承包商合作。Example Corp 有合作夥伴稱為 Widget Company 的公司，而 Widget Company 雇用名為 Shirley Rodriguez 的員工，其需要將資料放入儲存貯體中供 Example Corp 使用。Nikki 會建立名為的使用者群組，WidgetCo並將使用者命名為，`Shirley`並將 Shirley 新增至 WidgetCo 使用者群組。Nikki 也建立名為 aws-s3-bucket1 的特殊儲存貯體，供 Shirley 使用。

Nikki 更新現有政策或新增新的政策，以包含合作夥伴公司 Widget Company。例如，Nikki 可以建立新政策，拒絕使用 WidgetCo者群組的成員使用除寫入以外的任何動作。此政策只在有廣泛的政策可讓所有使用者存取多種 Amazon S3 動作時才必要。

IAM 教學課程

下列教學課 end-to-end 程提供 AWS Identity and Access Management (IAM) 一般工作的完整程序。它們適用於實驗室類型的環境，包括虛構公司名稱、使用者名稱等。他們的目的是提供一般指導。他們不能直接用於您的生產環境，需要仔細審查更動以符合組織環境的獨特需求。

教學課程

- IAM 教學課程：將存取權授予帳單主控台
- IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶
- IAM 教程：建立並連接您的第一個客戶受管政策
- IAM 教學課程：根據標籤定義存取 AWS 資源的許可
- IAM 教學課程：允許使用者管理其憑證和 MFA 設定

IAM 教學課程：將存取權授予帳單主控台

AWS 帳戶 擁有者 (AWS 帳戶根使用者) 可以授與 IAM 使用者和角色對其資 AWS Billing and Cost Management 料的存取權 AWS 帳戶。本教學課程中的說明可協助您設定已預先測試的案例。此案例可協助您獲得設定帳單許可的實作經驗，不需擔心影響您的主要 AWS 生產帳戶。

必要條件

在執行本教學課程中的步驟之前，請進行以下準備工作：

- 創建一個測試 AWS 帳戶。
- 以 root 使用者身分登入您的測試 AWS 帳戶 。
- 記錄測試帳戶的 AWS 帳戶 編號，以便您可以在教程中使用它。在本教學課程中，我們使用的是範例帳戶號碼 111122223333。每當有步驟使用該帳號時，請將其替換為您的測試帳戶號碼。

步驟 1：啟用對測試 AWS 帳戶帳單資訊的 IAM 存取權

在這種情況下，您以根使用者 AWS 帳戶 身分登入測試，以授予帳單資訊的 IAM 存取權。當您授予帳單資訊的 IAM 存取權時，它允許 IAM 使用者和角色存取 AWS Billing and Cost Management 主控台。這項設定並不會授予 IAM 使用者和角色這些主控台頁面的必要許可，而是會讓擁有必要 IAM 政策的 IAM 使用者或角色有權存取。如果政策已附加至 IAM 使用者或角色，但未啟用此設定，則這些政策授予的許可不會生效。

> **Note**
>
> AWS 帳戶 使用建立時 AWS Organizations ，預設為啟用帳單資訊的 IAM 存取權。

步驟 2：建立測試使用者和群組

在此案例中，您將授予 IAM 使用者帳單主控台的存取權，並建立兩個使用者：

- Pat Candella

  Pat 是財務部門的成員，負責計費和付款。Pat 需要完全存取您的帳單資訊 AWS 帳戶。

- Terry Whitlock

  Terry 是您的 IT 支援部門成員。在大多數情況下，Terry 不需要存取帳單主控台，但有時需要存取權，以回答財務部門員工的問題。

步驟 3：建立角色以授予 AWS Billing 主控台存取權

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。IAM 角色類似於 IAM 使用者，因為同樣是 AWS 身分，所以也有許可政策可決定該身分在 AWS中可執行和不可執行的操作。但是，角色的目的是讓需要它的任何人可代入，而不是單獨地與某個人員關聯。此外，角色沒有與之關聯的標準長期登入資料，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。您可以使用角色將存取權委派給通常無法存取 AWS 資源的使用者、應用程式或服務。在此案例中，您將建立 Terry Whitlock 可以擔任以存取帳單主控台的角色。

步驟 4：測試主控台存取權

完成核心任務之後，便可開始測試政策。測試的用意是為確保政策以您想要的運作方式執行。您可以藉由測試每個使用者的存取權來比較使用者體驗。

必要條件

在執行本教學課程中的步驟之前，請進行以下準備工作：

- 創建一個測試 AWS 帳戶。
- 以 root 使用者身分登入您的測試 AWS 帳戶 。
- 記錄測試帳戶的 AWS 帳戶 編號，以便您可以在教程中使用它。在本教學課程中，我們使用的是範例帳戶號碼 111122223333。每當有步驟使用該帳號時，請將其替換為您的測試帳戶號碼。

步驟 1：啟用對測試 AWS 帳戶帳單資訊的 IAM 存取權

在這種情況下，您以根使用者 AWS 帳戶 身分登入測試，以授予帳單資訊的 IAM 存取權。授予帳單資訊的存取權時，它允許 IAM 使用者和角色存取 AWS Billing and Cost Management 主控台。這項設定並不會授予 IAM 使用者和角色這些主控台頁面的必要許可，而是僅會讓擁有必要 IAM 政策的 IAM 使用者或角色有權存取。

> **Note**
>
> AWS 帳戶 使用建立時 AWS Organizations ，預設為啟用帳單資訊的 IAM 存取權。

若要啟用 IAM 使用者和角色對 Billing and Cost Management 主控台的存取

1. 使用您的 root 使 AWS Management Console 用者認證 (特別是您用來建立 AWS 帳戶的電子郵件地址和密碼) 登入。
2. 在導覽列上選取您的帳戶名稱，然後選取帳戶。
3. 向下捲動頁面，直至找到帳單資訊的 IAM 使用者和角色存取權部分，然後選取編輯。
4. 選取 Activate IAM Access (啟用 IAM 存取) 核取方塊，以啟用對 Billing and Cost Management 主控台頁面的存取。
5. 選擇 Update (更新)。

   頁面會顯示已啟用帳單資訊的 IAM 使用者/角色存取權訊息。

在本教學課程的下一步驟中，您將附加 IAM 政策，以授予或拒絕特定帳單功能的存取權。

步驟 2：建立測試使用者和群組

除了 root 用 AWS 戶外，您的測試帳戶沒有定義任何身份。為了提供帳單資訊的存取權，我們會建立額外的身分識別，以向其授予帳單資訊的存取權。

建立測試使用者和群組

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 IAM 主控台。在下一頁中，輸入您的密碼。

> **Note**
>
> 根使用者無法登入 以 IAM 使用者身分登入 頁面。如果您看到 以 IAM 使用者身分登入 頁面，請選擇頁面底部附近的 使用根使用者電子郵件登入。如需以 root 使用者身分登入的說明，請參閱《使用指南》中的「以 root 使用者身分登入」AWS 登入 。 AWS Management Console

2. 在導覽窗格中選取 使用者，然後選取 新增使用者。

> **Note**
>
> 如果您已啟用 IAM 身分中心，則 AWS Management Console 會顯示提醒您最好在 IAM 身分中心管理使用者存取權限。在本教學課程中，我們建立的 IAM 使用者將學習如何提供帳單資訊的存取權。如果您已在 IAM Identity Center 建立使用者，請使用 IAM Identity Center 而非 IAM 將帳單許可集指派給這些使用者或群組。

3. 在 User name (使用者名稱) 中輸入 **`pcandella`**。名稱不可含有空格。
4. 選取 [為使用者提供 AWS Management Console— 選用的存取權限] 旁邊的選取方塊，然後選擇要建立 IAM 使用者。
5. 在 主控台密碼 下選取 自動產生的密碼。
6. 清除位於使用者必須在下次登入時建立新密碼 (建議)旁的選取方塊，然後選取下一步。由於此 IAM 使用者用於測試，因此我們將下載密碼，以供驗證程序使用。
7. 在 設定許可 頁面的 許可選項 下方選取 新增使用者至群組。然後，在 使用者群組 下方選取 建立群組。
8. 在 建立使用者群組 頁面的 使用者群組名稱 中輸入**`BillingGroup`**。然後，在 [權限] 原則下，選取 AWS 受管理的工作功能原則 [計費]。
9. 選取 建立使用者群組 以返回 設定許可 頁面。
10. 在使用者群組下方，選取先前建立之 **`BillingGroup`** 的選取方塊。
11. 選取 下一步 以繼續前往 檢閱和建立 頁面。
12. 在檢閱和建立頁面上，檢閱新使用者的使用者群組成員資格清單。準備好繼續時，請選取 建立使用者。
13. 在擷取密碼頁面上，選取下載 .csv 檔案，以儲存含有使用者登入資訊 (連線 URL、使用者名稱和密碼) 的 .csv 檔案。

    以此 IAM 使用者身分登入時，儲存此檔案以 AWS 作為參考

14. 選取返回使用者清單
15. 使用下列修改內容重複此程序，以建立 Terry Whitlock 的使用者和支援使用者群組。

    a. 在步驟 3 中，針對使用者名稱，輸入 **twhitlock**。
    b. 在步驟 8 中，針對使用者群組名稱，輸入 **SupportGroup**。然後，在 [權限] 原則下，選取 [受 AWS 管理的工作功能] 原則。SupportUser

您可以在主控台清單中檢閱新的 IAM 使用者、群組和角色。針對您建立的每個項目，您可以選取名稱，以檢視其詳細資訊。當您檢視使用者詳細資料時，主控台會顯示 [帳單] 列在 [權限] 政策下，**pcandella**並SupportUser列在的 [權限] 政策下方**twhitlock**。

如需使用政策授予 IAM 使用者存取 AWS Billing and Cost Management 功能的詳細資訊，請參閱《AWS Billing 使用者指南》中的對 AWS Billing 使用以身分為基礎的政策 (IAM 政策)。

步驟 3：建立角色以授予 AWS Billing 主控台存取權

您可以使用角色來授予 IAM 使用者帳單主控台的存取權。角色會提供使用者在需要時可以使用的臨時認證。在本教學課程中，財務部門的支援請求要求使用者 **twhitlock** 調查問題時，其必須能夠存取帳單資訊。

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 IAM 主控台。在下一頁中，輸入您的密碼。

   > **Note**
   >
   > 根使用者無法登入 以 IAM 使用者身分登入 頁面。如果您看到 以 IAM 使用者身分登入 頁面，請選擇頁面底部附近的 使用根使用者電子郵件登入。如需以 root 使用者身分登入的說明，請參閱《使用指南》中的「以 root 使用者身分登入」AWS 登入 。 AWS Management Console

2. 在導覽窗格中，選取使用者，然後選取 **twhitlock** 使用者，以便檢視使用者詳細資訊。將 **twhitlock** 使用者的 ARN 複製到剪貼簿。
3. 在導覽窗格中，選取角色，然後選取建立角色。
4. 在選取可信任實體頁面上，選取自訂信任政策，然後在編輯陳述式下完成下列項目：

   - 為 STS 新增動作-確認AssumeRole已選取。
   - 新增主體 – 選取新增可顯示新增主體對話方塊。對於主體類型，選取 IAM 使用者，然後針對 ARN 貼上您在步驟 16 中複製到剪貼簿的 twhitlock 使用者的 ARN。然後選取新增主體。

5. 選取下一步可前往新增許可頁面。
6. 在篩選方塊中的 [權限] 原則下，輸入**Billing**並選取受 AWS 管理工作功能原則 [帳單]。
7. 選取下一步可前往命名、檢閱和建立頁面。在角色名稱下，輸入 **TempBillingAccess**，然後選取建立角色。

   系統會通知您已建立角色。檢視角色，以顯示有關角色的詳細資訊。在摘要區段中，請注意下列資訊：

   - 根據預設，工作階段最長持續時間為 1 小時。在此之後，擔任該角色的使用者將恢復為其基本帳戶許可。如果使用者想繼續使用該角色許可，則必須再次切換角色。您可以編輯角色，以增加持續時間上限。最長的工作階段持續時間可能是 12 小時。
   - 在主控台中切換角色的連結。您可以複製連結，將其直接提供給在信任政策中新增為主體的使用者。您可以從信任關係索引標籤檢視和編輯信任政策。

步驟 4：測試主控台存取權

建議您以測試使用者的身分登入來測試存取權，以了解使用者可能經歷的情況。使用下列步驟，用兩個測試帳戶登入，查看不同存取權之間的差異。

以兩個測試使用者登入的方式測試帳單存取權

1. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 IAM 主控台。

   > **Note**
   >
   > 為方便起見， AWS 登入頁面會使用瀏覽器 Cookie 來記住您的 IAM 使用者名稱和帳戶資訊。如果您先前以不同的使用者身分登入，請選擇在頁面底部附近的 Sign in to a different account (登入不同的帳戶)，返回主要登入頁面。您可以在該處輸入帳 AWS 戶 ID 或帳戶別名，以重新導向至帳戶的 IAM 使用者登入頁面。

2. 使用以下提供的步驟登入每個使用者，讓您得以比較不同的使用者體驗。

   完整存取

   a. 以使用者 AWS 帳戶 身分登入**pcandella**。

   b. 在導覽列上，選擇 pcandella@111122223333，然後選擇帳單儀表板。

   c. 瀏覽各個頁面，然後選擇各種按鈕，以確保您有完整的修改許可。

沒有存取權

a. 以使用者 AWS 帳戶 身分登入**twhitlock**。
b. 在導覽列上，選擇 twhitlock@111122223333，然後選擇帳單儀表板。
c. 會顯示一則訊息，說明您需要權限。無可見帳單資料。

切換角色以提升存取權

a. 以使用者 AWS 帳戶 身分登入**twhitlock**。
b. 在導覽列上，選擇 twhitlock@111122223333，然後選擇切換角色。

   切換角色頁面隨即開啟。依照以下方式填寫資訊：

   - 帳戶 - 111122223333
   - 角色 - **TempBillingAccess**

   選擇切換角色

   或者，您可以使用在主控台中切換角色的連結中提供的 URL 來開啟切換角色頁面。
c. 控制台會顯示「AWS Billing 儀表板」，導覽列會顯示 TempBillingAccess@111122223333。

Summary

您現在已完成提供 IAM 使用者存取 AWS Billing 主控台之權限的必要步驟。因此，您可以第一手瞭解使用者帳單主控台體驗的情況。您現在可以隨時繼續在生產環境中實作此邏輯。

相關資源

如需 AWS Billing 使用者指南中的相關資訊，請參閱下列資源：

- 啟用 AWS Billing 主控台的存取權
- AWS 帳單政策範例
- 使用身分型政策 (IAM 政策) 進行帳單 AWS
- 移轉存取控制 AWS Billing

如需與 IAM 使用者指南中的相關資訊，請參閱下列資源：

- 受管政策與內嵌政策
- 控制 IAM 使用者對 AWS Management Console的存取
- 將政策連接至 IAM 使用者群組

IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶

此教學課程教導您如何使用角色將存取許可委派給您所擁有不同 AWS 帳戶 中的資源 (生產和開發)。您與不同帳戶中的使用者分享一個帳戶中的資源。透過以這種方式設定跨帳戶存取，您不需要在每個帳戶中建立個別的 IAM 使用者。此外，使用者不需要為了存取不同 AWS 帳戶中的資源，登出一個帳戶然後登入另一個帳戶。設定角色之後，您會看到如何使用 AWS Management Console AWS CLI、和 API 中的角色。

> **Note**
>
> IAM 角色和資源型政策只會在單一分割內跨帳戶委派存取許可。例如，假設您在標準 aws 分割區的美國西部 (加利佛尼亞北部) 中有一個帳戶。您在 aws-cn 分割區的中國 (北京) 中也有一個帳戶。您不能使用中國 (北京) 中帳戶的 Amazon S3 資源型政策，對標準 aws 帳戶中的使用者允許存取許可。

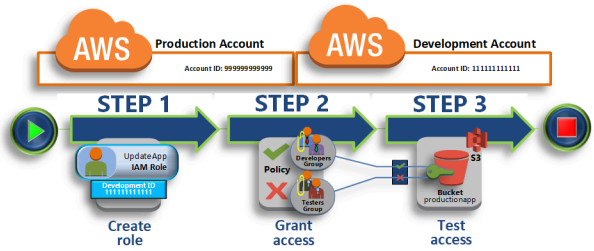
在本教學中，生產帳戶管理即時應用程式。開發人員和測試人員可以將開發帳戶作為自由測試應用程式的沙盒。在每個帳戶中，應用程式資訊存放在 Amazon S3 儲存貯體中。您管理開發帳戶中的 IAM 使用者，在該帳戶中您有兩個 IAM 使用者群組：開發人員和測試人員。兩個群組中的使用者均具有許可能在開發帳戶中工作和存取資源。有時候，開發人員必須在生產帳戶中更新即時應用程式。開發人員將這些應用程式存放在稱為 productionapp 的 Amazon S3 儲存貯體中。

在本教學的結尾，您會擁有：

- 在開發帳戶 (受信任帳戶) 中允許擔任生產帳戶中特定角色的使用者。
- 在生產帳戶 (信任帳戶) 中允許存取特定 Amazon S3 儲存貯體的角色。
- 生產帳戶中的 productionapp 儲存貯體。

開發人員可以使用中的角色 AWS Management Console 來存取生產帳戶中的productionapp值區。他們也可以透過使用 API 呼叫來存取儲存貯體，而該呼叫是由角色提供的暫時性憑證進行身分驗證。測試人員進行類似嘗試來使用角色會失敗。

此工作流程有三個基本步驟：



在生產帳戶中建立角色

首先，您可 AWS Management Console 以使用在生產帳戶 (識別碼編號 999999999999) 和開發帳戶 (識別碼編號 1111111111) 之間建立信任。首先，您可以建立名為的 IAM 角色UpdateApp。建立角色後，您可以定義開發帳戶為受信任的實體並指定許可政策，該政策允許受信任的使用者更新 `productionapp` 儲存貯體。

授予角色存取權

在本節中，您修改 IAM 使用者群組政策，拒絕測試人員存取 `UpdateApp` 角色。因為測試人員在這種情況下有 PowerUser 訪問權限，你必須明確拒絕使用該角色的能力。

透過切換角色測試存取

最後，作為開發人員，您使用 `UpdateApp` 角色來更新生產帳戶中的 `productionapp` 儲存貯體。您會看到如何透過 AWS 主控台 AWS CLI、和 API 存取角色。

必要條件

此教學課程假設您已備妥下列項目：

- 您可以使用兩個單獨 AWS 帳戶 的帳戶，一個代表開發帳戶，另一個代表生產帳戶。
- 開發帳戶中使用者和使用者群組的建立和設定，如下所示：

使用者	使用者群組	許可
David	開發人員	兩個使用者都可以登入並使用開發帳戶 AWS Management Console 中的。
Jane	測試人員	

- 您不需要在生產帳戶中建立有任何使用者或使用者群組。
- 在生產帳戶中建立一個 Amazon S3 儲存貯體。您可在此教學課程中將其稱為 `ProductionApp`，但因為 S3 儲存貯體名稱必須是全域唯一，所以您必須使用不同名稱的儲存貯體。

在生產帳戶中建立角色

您可以允許其中一個 AWS 帳戶 使用者存取另一個資源 AWS 帳戶。若要執行此作業，請建立一個角色並定義誰可以存取該角色，以及角色可授與哪些許可給切換至該角色的使用者。

在教學課程的此步驟中，您在生產帳戶中建立角色，並指定開發帳戶作為受信任的實體。您也限制角色的許可為僅擁有讀取和寫入 `productionapp` 儲存貯體的存取許可。被授予許可以使用角色的任何人都可以讀取和寫入 `productionapp` 儲存貯體。

在您可以創建角色之前，您需要開發的帳戶 ID AWS 帳戶。每個都 AWS 帳戶 有指派給它的唯一帳戶 ID 識別碼。

若要取得開發 AWS 帳戶 ID

1. 以開發帳戶的管理員身分登入，然後在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。 AWS Management Console
2. 在導覽列中，選擇 Support (支援)，然後選擇 Support Center (支援中心)。您目前登入的 12 位數帳戶號碼 (ID) 會顯示在 Support Center (支援中心) 導覽窗格。在此案例中，您可以使用帳戶 ID 111111111111 表示開發帳戶。不過，如果您在測試環境中使用此案例，則應該使用有效的帳戶 ID。

在生產帳戶中建立開發帳戶可以使用的角色

1. 以生產帳戶的管理員身分登入，然後開啟 IAM 主控台。 AWS Management Console
2. 建立角色之前，請準備定義角色所需許可的受管政策。您可以在稍後的步驟將此政策連接至角色。

   您想要設定 `productionapp` 儲存貯體的讀取和寫入存取許可。雖然 AWS 提供了一些 Amazon S3 受管政策，但沒有提供對單一 Amazon S3 儲存貯體的讀取和寫入存取權限。您可以改為建立自己的政策。

   在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
3. 選擇 JSON 標籤並從下列 JSON 政策文件複製文字。將此文字貼到 JSON 文字方塊中，以您 Amazon S3 儲存貯體的真實 ARN 替換資源 ARN (`arn:aws:s3:::productionapp`)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
       ],
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

`ListAllMyBuckets` 動作授予許可，可列出已經過身分驗證的請求發送者擁有的所有儲存貯體 `ListBucket` 許可允許使用者檢視 `productionapp` 儲存貯體中的物件。`GetObject`、`PutObject`、`DeleteObject` 許可允許使用者檢視、更新和刪除 `productionapp` 儲存貯體中的內容。

4. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

5. 在檢視與建立頁面上，針對政策名稱輸入 **`read-write-app-bucket`**。檢視政策授與的許可，然後選擇建立政策來儲存您的工作。

   新的政策會出現在受管政策清單中。

6. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。

7. 選擇 AWS 帳戶 角色類型。
8. 針對 Account ID (帳戶 ID)，輸入開發帳戶 ID。

   此教學課程對於開發帳戶使用範例帳戶 ID **111111111111**。您應該使用有效的帳戶 ID。如果您使用無效的帳戶 ID (例如 **111111111111**)，IAM 不會讓您建立新的角色。

   現在您不需要要求外部 ID，或要求使用者要有多重要素驗證 (MFA) 才能擔任角色。請保持不選取這些選項。如需更多詳細資訊，請參閱 在中使用多因素身份驗證（MFA）AWS。
9. 選擇 Next: Permissions (下一步：許可)，以設定與角色建立關聯的許可。
10. 選擇您之前建立的政策旁的核取方塊。

    > **秘訣**
    >
    > 針對 Filter (篩選條件)，選擇 Customer managed (客戶受管) 以篩選清單，使其僅包含建立的政策。這會隱藏 AWS 建立的政策並讓找出所需項目更容易。

    然後選擇下一步。
11. (選用) 藉由連接標籤作為鍵值對，將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
12. (選用) 在 Description (說明) 中，輸入新角色的說明。
13. 檢閱角色之後，選擇 Create role (建立角色)。

    UpdateApp 角色會顯示在角色清單中。

現在，您必須取得角色的 Amazon Resource Name (ARN)，這是該角色的唯一識別碼。當您修改開發人員和測試人員使用者群組的政策時，您將指定角色的 ARN 以授予或拒絕許可。

若要取得 ARN，請執行下列動 UpdateApp

1. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
2. 在角色清單中，選擇 UpdateApp 角色。
3. 在詳細資訊窗格的 Summary (摘要) 區段中，複製 Role ARN (角色 ARN) 值。

生產帳戶具有帳戶 ID 999999999999，所以角色 ARN 為 `arn:aws:iam::999999999999:role/UpdateApp`。請確定您提供生產帳戶的真實 AWS 帳戶 ID。

此時，您已建立生產和開發帳戶之間的信任。完成上述作業的方法是在生產帳戶中建立角色，可將開發帳戶識別為受信任主體。您也定義切換到 UpdateApp 角色的使用者可以執行什麼動作。

接下來，修改使用者群組的許可。

授予角色存取權

此時，測試人員和開發人員使用者群組成員都具有允許他們自由在開發帳戶中測試應用程式的許可。使用以下所需的步驟以新增各種許可允許切換到該角色。

若要修改開發人員使用者群組，以允許他們切換至 UpdateApp角色

1. 以開發帳戶管理員身分登入，並開啟 IAM 主控台。
2. 選擇 User Groups (使用者群組)，然後選擇 Developers (開發人員)。
3. 選擇 Permissions (許可) 標籤，選擇 Add permissions (新增許可)，然後選擇 Attach policy (連接政策)。
4. 請選擇 JSON 標籤。
5. 新增以下政策陳述式，以允許生產帳戶中的 UpdateApp 角色進行 `AssumeRole` 動作。請務必將`Resource`元素中的*`#### ID`* 變更為生產帳戶的實際 AWS 帳戶 ID。

   ```
   {
     "Version": "2012-10-17",
     "Statement": {
       "Effect": "Allow",
       "Action": "sts:AssumeRole",
       "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"
     }
   }
   ```

   `Allow` 效果明確地允許開發人員群組存取生產帳戶中的 UpdateApp 角色。任何嘗試存取角色的開發人員都會成功。
6. 選擇 Review policy (檢閱政策)。
7. 輸入名稱，例如 **`allow-assume-S3-role-in-production`**。

8. 選擇 Create policy (建立政策)。

在大多數環境中，可能不需要以下程序。但是，如果您使用 PowerUserAccess 權限，則某些群組可能已經能夠切換角色。下列程序顯示如何將新增 "Deny" 許可到測試人員群組，以確保他們無法擔任角色。如果在您的環境中並不需要此程序，我們建議您不要新增。"Deny" 許可會讓整體許可情況變得更複雜，不易管理和了解。只在沒有更好的選項時使用 "Deny" 許可。

若要修改測試人員使用者群組以拒絕擔任 **UpdateApp** 角色的許可

1. 選擇 User Groups (使用者群組)，然後選擇 Testers (測試人員)。
2. 選擇 Permissions (許可) 標籤，選擇 Add permissions (新增許可)，然後選擇 Attach policy (連接政策)。
3. 請選擇 JSON 標籤。
4. 新增以下政策陳述式以拒絕 UpdateApp 角色的 AssumeRole 動作。請務必將Resource元素中的*#### ID* 變更為生產帳戶的實際 AWS 帳戶 ID。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/UpdateApp"
  }
}
```

   Deny 效果明確地拒絕讓測試人員群組存取生產帳戶中的 UpdateApp 角色。任何嘗試存取角色的測試人員都會收到存取遭拒的訊息。
5. 選擇 Review policy (檢閱政策)。
6. 輸入名稱，例如 **deny-assume-S3-role-in-production**。
7. 選擇 Create policy (建立政策)。

開發人員使用者群組現在具有許可，可使用生產帳戶中的 UpdateApp 角色。測試人員使用者群組就無法使用 UpdateApp 角色。

接下來，您可以了解開發人員 David 如何存取生產帳戶中的 productionapp 儲存貯體。大衛可以從 AWS Management Console AWS CLI、或 AWS API 存取值區。

透過切換角色測試存取

完成此教學課程的前兩個步驟後，您會有一個角色可授與存取生產帳戶中的資源。您也會在開發帳戶中有一個使用者群組，其中的使用者被允許使用該角色。此步驟討論如何測試從 AWS Management Console AWS CLI、和 AWS API 切換到該角色。

> **Important**
>
> 只有在您以 IAM 使用者或聯合身分使用者登入之後，才可以切換到角色。此外，如果您啟動 Amazon EC2 執行個體以執行應用程式，應用程式可透過其執行個體設定檔擔任角色。以 AWS 帳戶根使用者身分登入時無法切換到特定角色。

切換角色 (主控台)

如果 David 需要在中的生產環境中工作 AWS Management Console，他可以使用「切換角色」來執行此操作。他會指定帳戶 ID 或別名和角色名稱，而其許可會立即切換到角色所允許的項目。然後他可以利用主控台來使用 `productionapp` 儲存貯體，但無法使用生產中的任何其他資源。當 David 使用角色時，他也無法運用他在開發帳戶中的進階使用者許可。那是因為一次只有一組許可有效。

> **Important**
>
> AWS Management Console 僅使用切換角色適用於不需要`ExternalId`. 例如，假設您將您帳戶的存取權授與第三方，並在許可政策的 `Condition` 元素中需要 `ExternalId`。在這種情況下，第三方只能通過使用 AWS API 或命令行工具訪問您的帳戶。第三方無法使用主控台，因為它無法提供 `ExternalId` 的值。如需有關此案例的詳細資訊將 AWS 資源存取權授予第三方時，如何使用外部 ID，請參閱 AWS 安全性部落格 AWS Management Console中的和如何啟用跨帳戶存取。

David 可以使用 IAM 提供的兩種方式來進入 Switch Role (切換角色) 頁面：

- David 從他們的管理員那裡收到指向預先定義的 Switch Role (切換角色) 組態的連結。該連結是在 Create Role (建立角色) 精靈的最終頁面上，或在跨帳戶角色的 Role Summary (角色摘要) 頁面上，提供給管理員。選擇此連結會帶領 David 前往已填寫 Account ID (帳戶 ID) 和 Role name (角色名稱) 欄位的 Switch Role (切換角色) 頁面。David 需要做的只有選擇 Switch Role (切換角色)。
- 管理員不會在電子郵件中傳送連結，但會改為傳送 Account ID (帳戶 ID) 號碼和 Role Name (角色名稱) 的值。若要切換角色，大衛必須手動輸入該值。下列程序中會以圖表說明：

擔任角色

1. David 在開發 AWS Management Console 用戶組中登錄了使用他的普通用戶。
2. 他們選擇管理員電郵給他們的連結。這會將 David 導航到 Switch Role (切換角色) 頁面，其中帳戶 ID 或別名以及角色名稱資訊已填寫。

   —或—

   David 在導覽列上選擇他們的名稱 (Identity (身分) 選單)，然後選擇 Switch Roles (切換角色)。

   如果這是 David 第一次以這種方式嘗試存取 Switch Role (切換角色) 頁面，他首先停留在初次執行的 Switch Role (切換角色) 頁面。此頁面提供有關切換角色如何讓使用者跨 AWS 帳戶管理資源的額外資訊。David 必須在此頁面上選擇 Switch Role (切換角色)，以完成此程序的剩餘部分。
3. 接下來，為了存取角色，David 必須手動輸入生產帳戶 ID 號碼 (999999999999) 和角色名稱 (`UpdateApp`)。

   此外，David 想要監控 IAM 中目前作用中的角色 (以及相關聯的許可)。為了追蹤此資訊，他在 Display Name (顯示名稱) 文字方塊中輸入 `PRODUCTION`、選擇紅色選項，然後選擇 Switch Role (切換角色)。
4. David 現在可以使用 Amazon S3 主控台來使用 `UpdateApp` 角色所具有許可的 Amazon S3 儲存貯體或任何其他資源。
5. 完成後，David 可以回到他們的原始許可。為了執行此作業，他們在導覽列上選擇 PRODUCTION (生產) 角色顯示名稱，然後選擇 Back to David @ 111111111111 (返回 David @ 111111111111)。
6. 下次 David 想要切換角色和在導覽列中選擇 Identity (身分) 選單時，他會看到上次的 PRODUCTION 項目仍在那裡。他可以只要選擇該項目即可立即切換角色，而無需重新輸入帳戶 ID 和角色名稱。

切換角色 (AWS CLI)

如果 David 需要在命令列於生產環境中工作，則可以使用 AWS CLI。他執行 `aws sts assume-role` 命令，並傳遞角色 ARN 以取得該角色的暫時性安全憑證。然後，他會在環境變數中設定這些認證，以便後續 AWS CLI 命令使用角色的權限來運作。雖然 David 使用該角色，他無法在開發帳戶中使用他的進階使用者許可，因為一次只能有一組許可有效。

請注意，所有存取金鑰和權杖僅為範例，不能如下所示般使用。以您實際環境中的適當值取代。

擔任角色

1. David 開啟命令提示字元視窗，並透過執行命令來確認 AWS CLI 用戶端正在運作：

```
aws help
```

> **Note**
>
> David 的預設環境使用其預設設定檔的 David 使用者憑證，這是使用 aws configure 命令建立的。如需詳細資訊，請參閱 AWS Command Line Interface 使用者指南中的設定 AWS Command Line Interface。

2. 他透過執行以下命令開始切換角色程序，以切換到生產帳戶中的 UpdateApp 角色。他從建立角色的管理員角色收到角色 ARN。該命令也會要求您提供工作階段名稱，您可以選擇您想要的任何文字。

```
aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/UpdateApp" --role-
session-name "David-ProdUpdate"
```

David 然後會在輸出中看到以下：

```
{
    "Credentials": {
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLE
CvSRyh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDy
EXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPyOj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3Uuysg
sKdEXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLEsnf87e
NhyDHq6ikBQ==",
        "Expiration": "2014-12-11T23:08:07Z",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

3. David 在輸出的 Credentials (憑證) 區段看到他們需要的三個部分。

- `AccessKeyId`
- `SecretAccessKey`
- `SessionToken`

David 需要配置 AWS CLI 環境，以便在後續調用中使用這些參數。如需設定憑證各種方式的資訊，請參閱設定 AWS Command Line Interface。您不能使用 `aws configure` 命令，因為它不支援擷取工作階段權杖。不過，您可以將資訊手動輸入到組態檔案中。由於這些是暫時性憑證，其過期時間相對較短，最簡單的方式是將它們新增至您目前命令列工作階段的環境。

4. 為新增三個值到環境，David 剪下並貼上前一個步驟的輸出到以下命令中。建議您剪下並貼上到簡單的文字編輯器，來處理工作階段權杖輸出中的換行問題。它必須新增為單一長字串，即使在這裡為清楚起見以換行方式顯示。

> **Note**
>
> 以下範例顯示 Windows 環境中指定命令，其中 "set" 是用來建立環境變數的命令。在 Linux 或 macOS 電腦上，請改為使用命令 "export"。範例的所有其他部分在所有三個環境中均為有效。
> 如需使用 Windows Powershell 工具的詳細資訊，請參閱切換至 IAM 角色 (適用於視窗的工具 PowerShell)

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvS
Ryh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDyEXA
MPLEKEY9/
g7QRUhZp4bqbEXAMPLENwGPyOj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UusKd
EXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLENhykxiHen
DHq6ikBQ==
```

此時，任何下列命令在這些憑證所識別角色的許可下執行。在 David 的案例中，為 `UpdateApp` 角色。

5. 執行命令來存取生產帳戶中的資源。在這個範例中，David 只需要使用下列命令列出他們的 S3 儲存貯體的內容。

```
aws s3 ls s3://productionapp
```

由於 Amazon S3 儲存貯體名稱是全域唯一，所以不需要指定擁有該儲存貯體的帳戶 ID。若要存取其他 AWS 服務的資源，請參閱該服務的 AWS CLI 文件，以取得參考其資源所需的命令和語法。

使用 AssumeRole 應用AWS 程式介面

當 David 需要從程式碼進行生產帳戶的更新時，他進行 AssumeRole 呼叫以擔任 UpdateApp 角色 。此呼叫會傳回他可用來存取生產帳戶中 productionapp 儲存貯體的暫時性憑證。David 可以使用這些憑證，來進行 API 呼叫以更新 productionapp 儲存貯體。不過，他無法進行 API 呼叫以存取生產帳戶中任何其他資源，即使他在開發帳戶中具有進階使用者許可。

擔任角色

1. David 呼叫 AssumeRole 作為應用程式的一部分。他們必須指定 UpdateApp ARN：arn:aws:iam::999999999999:role/UpdateApp。

   來自 AssumeRole 呼叫的回應包含具有 AccessKeyId 和 SecretAccessKey 的暫時性憑證。也包含 Expiration 時間，指出憑證何時過期以及必須請求新憑證的時間。

2. David 使用暫時性憑證，進行 s3:PutObject 呼叫以更新 productionapp 儲存貯體。他們會將憑證作為 AuthParams 參數傳遞到 API 呼叫。由於暫時性角色憑證只有 productionapp 儲存貯體的讀取和寫入存取權，在生產帳戶中的任何其他動作會被拒絕。

如需程式碼範例 (使用 Python)，請參閱 切換到身分與存取權管理角色 (AWS API)。

相關資源

- 如需有關 IAM 使用者和使用者群組的詳細資訊，請參閱 IAM 身分 (使用者、使用者群組和角色)。
- 如需使用 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。
- 若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。

Summary

您已完成跨帳戶 API 存取教學課程。您建立了一個角色，以建立與另一個帳戶的信任，並定義了受信任實體可採取的動作。然後，您修改了群組政策，以控制哪個 IAM 使用者可存取角色。因此，開發帳戶的開發人員可以透過使用暫時性憑證，來更新生產帳戶中的 `productionapp` 儲存貯體。

IAM 教程：建立並連接您的第一個客戶受管政策

在本教學課程中，您可 AWS Management Console 以使用建立客戶受管政策，然後將該政策附加到 AWS 帳戶. 您建立的政策允許 IAM 測試使用者使用唯讀權限直接登入。 AWS Management Console

此工作流程有三個基本步驟：

步驟 1：建立政策

根據預設，IAM 使用者沒有執行任何動作的許可。它們無法存取 AWS 管理主控台或管理其中的資料，除非您允許。在此步驟中，您建立客戶受管政策，其允許任何連接的使用者登入主控台。

步驟 2：連接政策

當您將政策連接到使用者時，使用者會繼承與該政策相關聯的所有存取許可。在此步驟中，您會將新的政策連接到測試使用者。

步驟 3：測試使用者存取許可

一旦連接政策，便可以使用者身分登入並測試政策。

必要條件

若要執行此教學課程中的步驟，您需具備以下內容：

- 您可以使用具有管理許可的 IAM 使用者身分登入。 AWS 帳戶
- 未擁有指派許可或群組成員資格的測試 IAM 使用者，如下所示：

使用者名稱	群組	許可
PolicyUser	<無>	<無>

步驟 1：建立政策

在此步驟中，您會建立客戶受管政策，允許任何附加的使用者以唯讀存取權限登入 IAM 資料。 AWS Management Console

為您的測試使用者建立政策

1. 以具有管理員許可的使用者身分登入 IAM 主控台 (https://console.aws.amazon.com/iam/)。
2. 在導覽窗格上選擇 Policies (政策)。
3. 在內容窗格中，選擇 Create policy (建立政策)。
4. 選擇 JSON 選項，並從下列 JSON 政策文件中複製文字。將此文字貼上至 JSON 文字方框中。

```
{
    "Version": "2012-10-17",
    "Statement": [ {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateCredentialReport",
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    } ]
}
```

5. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行更改或在 Visual editor (視覺編輯工具) 索引標籤中選擇 Review policy (檢閱政策)，IAM 可能會調整您的政策結構以針對視覺編輯工具進行最佳化。如需詳細資訊，請參閱 政策結構調整。

6. 在檢視與建立頁面上，針對政策名稱輸入 **`UsersReadOnlyAccessToIAMConsole`**。檢視政策授與的許可，然後選擇建立政策來儲存您的工作。

   新的政策會出現在受管政策清單中，並且已準備好連接。

步驟 2：連接政策

接下來，將您剛建立的政策連接到測試 IAM 使用者。

連接政策到您的測試使用者

1. 在 IAM 主控台的導覽窗格中，選擇 Policies (政策)。
2. 在政策清單頂端的搜尋方塊中，開始輸入 **UsersReadOnlyAccesstoIAMConsole** 直到您可以看到您的政策。然後選擇列表中 UsersReadOnlyAccessToIamConsole 旁邊的單選按鈕。
3. 選擇 Actions (動作) 按鈕，然後選擇 Attach (連接)。
4. 在 IAM 實體中，選擇篩選使用者的選項。
5. 在搜尋方塊中，開始輸入 **PolicyUser** 直到該使用者顯示在清單上。然後勾選清單中該使用者旁的方塊。
6. 選擇連接政策。

您可以連接政策到 IAM 測試使用者，這表示使用者現在擁有唯讀存取 IAM 主控台的許可。

步驟 3：測試使用者存取許可

對於本教學，建議您以測試使用者身分登入來測試存取許可，如此才能了解使用者可能經歷的情況。

以測試使用者登入來測試存取許可

1. 以 `PolicyUser` 測試使用者身分登入位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 瀏覽主控台頁面並嘗試建立新的使用者或群組。請注意，`PolicyUser` 可以顯示資料，但無法建立或修改現有 IAM 資料。

相關資源

如需相關資訊，請參閱下列資源：

- 受管政策與內嵌政策
- 控制 IAM 使用者對 AWS Management Console的存取

Summary

現在您已成功完成所有建立和連接客戶受管政策的必要步驟。因此，您可以利用您的測試帳戶登入 IAM 主控台，以查看使用者的體驗。

IAM 教學課程：根據標籤定義存取 AWS 資源的許可

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。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 資源，包括 IAM 實體 (使用者或角色) 和 AWS 資源。您可以定義使用標籤條件金鑰的政策，以根據主體的標籤來授與許可。當您使用標籤來控制對 AWS 資源的存取時，您可以透過更少的 AWS 原則變更，讓您的團隊和資源成長。ABAC 政策比傳統政 AWS 策更靈活，因為它要求您列出每個單獨的資源。如需 ABAC 及其優於傳統政策的詳細資訊，請參閱 ABAC 是做什麼用的 AWS？。

> **Note**
>
> 您必須為每個工作階段標籤傳遞單一值。 AWS Security Token Service 不支援多值工作階段標籤。

主題

教學課程概觀

本教學說明如何建立和測試政策，以允許具有主體標籤的 IAM 角色存取具有相符標籤的資源。當主體向 AWS發出請求時，會根據主體和資源標籤是否相符來授與其許可。此策略可讓個人只檢視或編輯其工作所需的 AWS 資源。

案例

假設您是 Example Corporation 這間大型公司的首席開發人員，也是經驗豐富的 IAM 管理員。您很熟悉如何建立及管理 IAM 使用者、角色和政策。您想要確保您的開發工程師和品質保證團隊成員可以存取所需資源。您也需要可隨著公司成長而擴展的策略。

您選擇使用 AWS 資源標籤和 IAM 角色主要標籤，為支援它的服務實作 ABAC 策略，從開始 AWS Secrets Manager。若要了解哪些服務支援以標籤為基礎的授權，請參閱 AWS 與 IAM 搭配使用的服務。若要瞭解您可以在原則中與每個服務的動作和資源搭配使用哪些標記條件金鑰，請參閱服AWS 務的動作、資源和條件金鑰。您可以設定您的 SAML 類型或 Web 身分提供者，將工作階段標籤傳遞至 AWS。當您的員工聯合到時 AWS，他們的屬性會套用至其產生的主參與者。 AWS您接著可以使用 ABAC 來根據這些屬性允許或拒絕許可。若要了解搭配 SAML 聯合身分使用工作階段標籤與本教學的不同，請參閱 IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記。

您的工程和品質保證團隊成員負責 Pegasus 或 Unicorn 專案。您可以選擇下列 3 個字元的專案和團隊標籤值：

- `access-project` = `peg` 適用於 Pegasus 專案
- `access-project` = `uni` 適用於 Unicorn 專案
- `access-team` = `eng` 適用於工程團隊
- `access-team` = `qas` 適用於品質保證團隊

此外，您還可以選擇要求成`cost-center`本分配標籤來啟用自訂 AWS 帳單報告。如需詳細資訊，請參閱 AWS Billing and Cost Management 使用者指南中的使用成本分配標籤。

重要決策摘要

- 員工使用 IAM 使用者憑證登入，然後擔任其團隊和專案的 IAM 角色。如果您的公司有專屬身分系統，您可以設定聯合以允許員工在沒有 IAM 使用者的情況下擔任角色。如需更多詳細資訊，請參閱 IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記。
- 相同的政策會連接至所有角色。動作會根據標籤而受允許或拒絕。

- 員工可以建立新的資源，但僅限於將相同的標籤連接至套用到其角色的資源時。這可確保員工在建立資源之後能夠檢視資源。管理員不再需要使用新資源的 ARN 來更新政策。
- 員工可以讀取其團隊所擁有的資源，不論專案為何。
- 員工可以更新及刪除自己團隊和專案所擁有的資源。
- IAM 管理員可以新增新專案的角色。他們可以建立並標記新的 IAM 使用者，以允許存取適當的角色。管理員不需要編輯政策來支援新的專案或團隊成員。

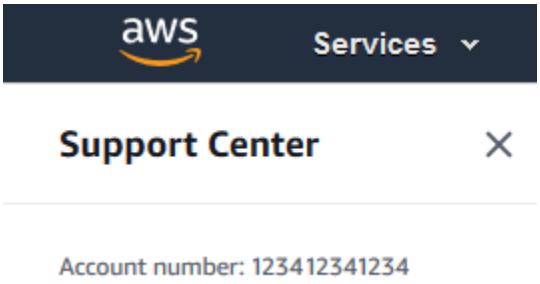
在此教學中，您將為每個資源加上標籤、為您的專案角色加上標籤並將政策新增至角色，以允許前述的行為。產生的政策允許角色 `Create`、`Read`、`Update` 和 `Delete` 存取以相同專案和團隊標籤所標記的資源。政策也允許跨專案 `Read` 存取以相同團隊標記的資源。

必要條件

若要執行此教學課程中的步驟，您必須具備以下內容：

- 您可 AWS 帳戶 以以具有管理權限的使用者身分登入。
- 您的 12 位數帳戶 ID，用於在步驟 3 中建立角色。

  若要使用尋找您的 AWS 帳戶 ID 號碼 AWS Management Console，請選擇右上角導覽列上的 [Support 援]，然後選擇 [Sup port 中心]。帳戶號碼 (ID) 會顯示在左側導覽窗格。



- 體驗在 AWS Management Console中建立並編輯 IAM 使用者、角色和政策。不過，如果您需要記住 IAM 管理程序的協助，本教學課程會提供可讓您檢視 step-by-step 指示的連結。

步驟 1：建立測試使用者

針對測試，請建立四個 IAM 使用者，並授與其許可擔任具有相同標籤的角色。這可讓您更輕鬆地新增更多使用者到您的團隊。當您標記使用者時，使用者會自動取得擔任正確角色的存取權。如果使用者只在一個專案和團隊中工作，則您不必將使用者新增至角色的信任政策。

1. 建立下列名為 access-assume-role 的客戶受管政策。如需如何建立 JSON 政策的詳細資訊，請參閱 建立 IAM 政策。

   ABAC 政策：擔任任何 ABAC 角色，但僅限使用者和角色標籤相符時

   下列政策允許使用者擔任您帳戶中名稱字首為 access- 的任何角色。角色也必須以與使用者相同的專案、團隊和成本中心標籤來標記。

   若要使用此政策，請將斜體預留位置文字取代為您的帳戶資訊。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TutorialAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::account-ID-without-hyphens:role/access-*",
            "Condition": {
                "StringEquals": {
                    "iam:ResourceTag/access-project": "${aws:PrincipalTag/access-
project}",
                    "iam:ResourceTag/access-team": "${aws:PrincipalTag/access-
team}",
                    "iam:ResourceTag/cost-center": "${aws:PrincipalTag/cost-
center}"
                }
            }
        }
    ]
}
```

   若要將本教學擴展為大量使用者，您可以將政策連接至群組，並將每個使用者新增至群組。如需更多詳細資訊，請參閱 建立 IAM 使用者群組 及 在 IAM 使用者群組中新增和移除使用者。

2. 建立下列 IAM 使用者、連接 access-assume-role 許可政策。務必選取為使用者提供 AWS Management Console的存取權限，然後新增下列標籤。如需建立和標記新使用者的詳細資訊，請參閱 建立 IAM 使用者 (主控台)。

ABAC 使用者

使用者名稱	使用者標籤索引鍵	使用者標籤值
access-Arnav-peg-eng	access-project access-team cost-center	peg eng 987654
access-Mary-peg-qas	access-project access-team cost-center	peg qas 987654
access-Saanvi-uni-eng	access-project access-team cost-center	uni eng 123456
access-Carlos-uni-qas	access-project access-team cost-center	uni qas 123456

步驟 2：建立 ABAC 政策

建立下列名為 **access-same-project-team** 的政策。您會在後續步驟將此政策新增至角色。如需如何建立 JSON 政策的詳細資訊，請參閱 建立 IAM 政策。

如需您可以針對本教學進行調整的其他政策，請參閱下列頁面：

- 控制 IAM 主體的存取權限
- Amazon EC2：允許以程式設計方式和在主控台中開始或停止使用者已標記的 EC2 執行個體
- EC2：依據比對主體和資源的標籤啟動或停止執行個體
- EC2：依據標籤啟動或停止執行個體

- IAM：擔任具有特定標籤的角色

ABAC 政策：僅在主體與資源標籤相符時，才存取 Secrets Manager 資源

下列政策允許主體建立、讀取、編輯及刪除資源，但僅限資源受標記為與主體相同的鍵值對時。當主體建立資源時，必須新增 access-project、access-team、和 cost-center 標籤，其值符合主體的標籤。此政策也允許新增選用的 Name 或 OwnedBy 標籤。

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Sid": "AllActionsSecretsManagerSameProjectSameTeam",
         "Effect": "Allow",
         "Action": "secretsmanager:*",
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                 "aws:ResourceTag/access-project": "${aws:PrincipalTag/access-
project}",
                 "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}",
                 "aws:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"
             },
             "ForAllValues:StringEquals": {
                 "aws:TagKeys": [
                     "access-project",
                     "access-team",
                     "cost-center",
                     "Name",
                     "OwnedBy"
                 ]
             },
             "StringEqualsIfExists": {
                 "aws:RequestTag/access-project": "${aws:PrincipalTag/access-project}",
                 "aws:RequestTag/access-team": "${aws:PrincipalTag/access-team}",
                 "aws:RequestTag/cost-center": "${aws:PrincipalTag/cost-center}"
             }
         }
     },
     {
         "Sid": "AllResourcesSecretsManagerNoTags",
         "Effect": "Allow",
         "Action": [
```

```
                "secretsmanager:GetRandomPassword",
                "secretsmanager:ListSecrets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ReadSecretsManagerSameTeam",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:Describe*",
                "secretsmanager:Get*",
                "secretsmanager:List*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}"
                }
            }
        },
        {
            "Sid": "DenyUntagSecretsManagerReservedTags",
            "Effect": "Deny",
            "Action": "secretsmanager:UntagResource",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "aws:TagKeys": "access-*"
                }
            }
        },
        {
            "Sid": "DenyPermissionsManagement",
            "Effect": "Deny",
            "Action": "secretsmanager:*Policy",
            "Resource": "*"
        }
    ]
}
```

此政策的功能為何？

- `AllActionsSecretsManagerSameProjectSameTeam` 陳述式允許在所有相關資源上執行此服務的所有動作，但僅限於資源標籤符合主體標籤時。透過將 `"Action": "secretsmanager:*"` 新增至政策，政策會隨著 Secrets Manager 成長而成長。如果 Secrets Manager 新增新的 API 操作，您不需要將該動作新增至陳述式。陳述式使用三個條件區塊來實作 ABAC。只有在三個區塊全部傳回 true 時，才會允許此請求。
  - 如果資源上有指定的標籤鍵，而且其值符合主體的標籤，則此陳述式的第一個條件區塊會傳回 true。此區塊會為不相符的標籤或不支援資源標記的動作傳回 false。若要瞭解此區塊不允許哪些動作，請參閱的動作、資源和條件索引鍵 AWS Secrets Manager。該頁面顯示對 Secret (秘密) 資源類型執行的動作支援 `secretsmanager:ResourceTag/tag-key` 條件索引鍵。部分 Secrets Manager 動作不支援該資源類型，包括 `GetRandomPassword` 和 `ListSecrets`。您必須建立額外的陳述式以允許這些動作。
  - 如果請求中傳遞的每個標籤鍵都包含在指定清單中，則第二個條件區塊會傳回 true。此動作是使用 `ForAllValues` 搭配 `StringEquals` 條件運算子來完成。如果未傳遞金鑰或金鑰集的子集，則條件會傳回 true。這允許了不允許在請求中傳遞標籤的 `Get*` 操作。如果申請者包含了不在清單中的標籤鍵，則條件會傳回 false。在請求中傳遞的每個標籤鍵，都必須符合此清單的成員。如需詳細資訊，請參閱 多值內容索引鍵。
  - 如果請求支援傳遞標籤 (如果這三個標籤都存在)，且符合主體標籤值，則第三個條件區塊會傳回 true。如果請求不支援傳遞標籤，此區塊也會傳回 true。這要歸功於條件運算子中的 `...IfExists`。如果在支援該標籤的動作期間沒有傳遞任何標籤，或標籤鍵和值不相符，則區塊會傳回 false。
- `AllResourcesSecretsManagerNoTags` 陳述式允許第一個陳述式不允許的 `GetRandomPassword` 和 `ListSecrets` 動作。
- 如果主體所標記的存取團隊標籤與資源相同，則 `ReadSecretsManagerSameTeam` 陳述式允許唯讀操作。無論專案或成本中心標籤為何，都允許此操作。
- `DenyUntagSecretsManagerReservedTags` 陳述式拒絕從 Secrets Manager 移除鍵開頭為 "access-" 之標籤的請求。這些標籤用於控制對資源的存取，因此移除標籤可能會移除許可。
- `DenyPermissionsManagement` 陳述式拒絕建立、編輯或刪除 Secrets Manager 以資源為基礎的政策的存取。這些政策可用來變更秘密的許可。

> **Important**
>
> 此政策使用策略來允許服務的所有動作，但會明確拒絕更改許可的動作。拒絕動作會覆寫任何其他允許主體執行該動作的政策。這可能會產生意外的結果。根據最佳實務，只有在該動作無

論何種情況都不應允許的情況下，才使用明確拒絕。否則，請允許包含個別動作的清單，且將不需要的動作預設拒絕。

步驟 3：建立角色

建立下列 IAM 角色，並連接您在先前步驟中建立的 **access-same-project-team** 政策。如需建立 IAM 角色的詳細資訊，請參閱 建立角色以委派許可給 IAM 使用者。如果您選擇使用聯合，而非 IAM 使用者和角色，請參閱 IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記。

ABAC 角色

任務職能	角色名稱	角色標籤	角色描述
Pegasus 工程專案	access-peg-engineering	access-project = peg access-team = eng cost-center = 987654	允許工程師讀取所有工程資源，並建立和管理 Pegasus 工程資源。
Pegasus 品質保證專案	access-peg-quality-assurance	access-project = peg access-team = qas cost-center = 987654	允許 QA 團隊讀取所有 QA 資源，並建立和管理所有 Pegasus QA 資源。
Unicorn 工程專案	access-uni-engineering	access-project = uni	允許工程師讀取所有工程資源，並建立和管理 Unicorn 工程資源。

任務職能	角色名稱	角色標籤	角色描述
		access-team = eng cost-center = 123456	
Unicorn Quality Assurance 專案	access-uni-quality-assurance	access-project = uni access-team = qas cost-center = 123456	允許 QA 團隊讀取所有 QA 資源，並建立和管理所有 Unicorn QA 資源。

步驟 4：測試建立秘密

連接至角色的許可政策允許員工建立秘密。只有在秘密以其專案、團隊和成本中心來標記時，才允許此操作。以使用者身分登入來確認您的許可如預期般運作且擔任正確角色，並測試 Secrets Manager 中的活動。

測試使用和不使用必要的標籤來建立秘密

1. 在您的主要瀏覽器視窗中，保持以管理員使用者的身分登入，讓您可以檢閱 IAM 中的使用者、角色和政策。使用瀏覽器無痕式視窗或個別的瀏覽器進行測試。前往以下位置以 access-Arnav-peg-eng IAM 使用者身分登入 Secrets Manager 主控台：https://console.aws.amazon.com/secretsmanager/。
2. 嘗試切換到 access-uni-engineering 角色。

   此操作會失敗，因為 access-project 和 cost-center 標籤值與 access-Arnav-peg-eng 使用者和 access-uni-engineering 角色不相符。

   如需「」中切換角色的詳細資訊 AWS Management Console，請參閱 切換到角色 (主控台)

3. 切換到 access-peg-engineering 角色。
4. 使用下列資訊存放新的秘密。若要了解如何存放秘密，請參閱 AWS Secrets Manager 使用者指南中的建立基本秘密。

> **Important**
>
> Secrets Manager 會顯示警示，指出您沒有與 Secrets Manager 搭配使用之其他 AWS 服務的許可。例如，若要建立 Amazon RDS 資料庫的憑證，您必須擁有描述 RDS 執行個體、RDS 叢集和 Amazon Redshift 叢集的許可。您可以忽略這些警示，因為您在本教學課程中並未使用這些特定 AWS 服務。

1. 在 Select secret type (選取秘密類型) 區段中，選擇 Other type of secrets (其他類型的秘密)。在兩個文字方塊中，輸入 test-access-key 和 test-access-secret。
2. 針對 Secret name (秘密名稱) 欄位輸入 test-access-peg-eng。
3. 從下表新增不同的標籤組合，並檢視預期的行為。
4. 選擇 Store (存放) 以嘗試建立秘密。在儲存失敗時，返回先前的 Secrets Manager 主控台頁面，並使用下表中的下一個標籤集。最後一個標籤集會受到允許，且將成功建立秘密。

test-access-peg-eng 角色的 ABAC 標籤組合

access-project 標籤值	**access-team** 標籤值	**cost-center** 標籤值	其他標籤	預期行為
(無)	(無)	(無)	(無)	已拒絕，因為 access-project 標籤值不符合角色的值 peg。
uni	eng	987654	(無)	已拒絕，因為 access-project 標籤值不符合角色的值 peg。
peg	qas	987654	(無)	已拒絕，因為 access-team 標籤值不符合角色的值 eng。
peg	eng	123456	(無)	已拒絕，因為 cost-center 標籤值不符合角色的值 987654。

access-project 標籤值	**access-team** 標籤值	**cost-center** 標籤值	其他標籤	預期行為
peg	eng	987654	owner = Jane	已拒絕，因為政策不允許額外的標籤 owner，即使所有三個必要標籤都存在且其值符合角色的值。
peg	eng	987654	Name = Jane	已允許，因為三個必要標籤都存在，且其值符合角色的值。您也可以包含選用的 Name 標籤。

5. 登出並針對每個下列角色和標籤值來重複此程序的前三個步驟。在此程序的第四個步驟中，測試任何遺漏的標籤、選用標籤、不允許的標籤，和您選擇的無效標籤值。然後，使用必要標籤來建立具有下列標籤和名稱的秘密。

   ABAC 角色和標籤

使用者名稱	角色名稱	秘密名稱	秘密標籤
access-Mary-peg-qas	access-peg-quality-assurance	test-access-peg-qas	access-project = peg access-team = qas cost-center = 987654
access-Saanvi-uni-eng	access-uni-engineering	test-access-uni-eng	access-project = uni access-team = eng cost-center = 123456

使用者名稱	角色名稱	秘密名稱	秘密標籤
access-Carlos-uni-qas	access-uni-quality-assurance	test-access-uni-qas	access-project = uni access-team = qas cost-center = 123456

步驟 5：測試檢視秘密

您連接到每個角色的政策允許員工檢視以其團隊名稱標記的任何秘密，不論他們的專案為何。在 Secrets Manager 中測試您的角色，確認您的許可如預期般運作。

測試檢視包含和不含必要標籤的秘密

1. 使用下列其中一個 IAM 使用者身分登入:
   - access-Arnav-peg-eng
   - access-Mary-peg-qas
   - access-Saanvi-uni-eng
   - access-Carlos-uni-qas
2. 切換至相符的角色：
   - access-peg-engineering
   - access-peg-quality-assurance
   - access-uni-engineering
   - access-uni-quality-assurance

   如需「」中切換角色的詳細資訊 AWS Management Console，請參閱切換到角色 (主控台)。
3. 在左側的導覽窗格中，選擇選單圖示以展開選單，然後選擇 Secrets (秘密)。
4. 無論您目前的角色為何，您應該都會看到資料表中的全部四個秘密。這是預期的結果，因為名為 access-same-project-team 的政策允許對所有資源執行 secretsmanager:ListSecrets 動作。

5. 選擇其中一個秘密的名稱。
6. 在秘密的詳細資訊頁面上，角色的標籤將決定您是否可以檢視頁面內容。將您的角色名稱與秘密名稱進行比較。如果這兩個名稱具有相同團隊名稱，則 `access-team` 標籤會相符。如果不相符，則會拒絕存取。

   每個角色的 ABAC 秘密檢視行為

角色名稱	秘密名稱	預期行為
access-peg-engineering	test-access-peg-eng	允許
	test-access-peg-qas	已拒絕
	test-access-uni-eng	允許
	test-access-uni-qas	已拒絕
access-peg-quality-assurance	test-access-peg-eng	已拒絕
	test-access-peg-qas	允許
	test-access-uni-eng	已拒絕
	test-access-uni-qas	允許
access-uni-engineering	test-access-peg-eng	允許
	test-access-peg-qas	已拒絕
	test-access-uni-eng	允許
	test-access-uni-qas	已拒絕
access-uni-quality-assurance	test-access-peg-eng	已拒絕
	test-access-peg-qas	允許
	test-access-uni-eng	已拒絕
	test-access-uni-qas	允許

7. 從頁面頂端的頁面導覽路徑中，選擇 Secrets (秘密) 以返回秘密清單。使用不同角色重複此程序中的步驟，以測試您是否可以檢視每個秘密。

步驟 6：測試可擴展性

偏好以屬性為基礎的存取控制 (ABAC) 多過以角色為基礎的存取控制 (RBAC) 的重要原因是可擴展性。當您的公司新增專案、團隊或人員時 AWS，您不需要更新 ABAC 驅動的原則。例如，假設 Example Company 為名為 Centaur 的新專案提供資金。一位名為 Saanvi Sarkar 的工程師將擔任 Centaur 的主要工程師，同時繼續進行 Unicorn 專案。Sanvi 還將檢閱 Peg 專案的工作。另外還有幾位新聘的工程師，包括 Nikhil Jayashankar，只負責 Centaur 專案。

若要將新專案加入至 AWS

1. 以 IAM 管理原使用者身分登入，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。
2. 在左側的導覽窗格中，選擇 Roles (角色)，然後新增名為 `access-cen-engineering` 的 IAM 角色 將 **`access-same-project-team`** 許可政策連接至角色，並新增下列角色標籤：

   - `access-project` = `cen`
   - `access-team` = `eng`
   - `cost-center` = `101010`
3. 在左側導覽窗格中，選擇 Users (使用者)。
4. 新增名為 `access-Nikhil-cen-eng` 的新使用者、連接名為 `access-assume-role` 的政策，並新增以下使用者標籤。

   - `access-project` = `cen`
   - `access-team` = `eng`
   - `cost-center` = `101010`
5. 使用 步驟 4：測試建立秘密 和 步驟 5：測試檢視秘密 中的程序。在另一個瀏覽器視窗中測試 Nikhil 是否只能建立 Centaur 工程秘密，並且可以檢視所有工程秘密。
6. 在您以管理員身分登入的主要瀏覽器視窗中，選擇使用者 `access-Saanvi-uni-eng`。
7. 在 [權限] 索引標籤上，移除access-assume-role權限原則。
8. 新增下列名為 `access-assume-specific-roles` 的內嵌政策 。如需將內嵌政策新增至使用者的詳細資訊，請參閱 為使用者或角色嵌入內嵌政策 (主控台)。

   ABAC 政策：僅擔任特定角色

此政策允許 Saanvi 擔任 Pegasus 和 Centaur 專案的工程角色。您必須建立此自訂政策，因為 IAM 不支援多值標籤。您無法以 `access-project = peg` 和 `access-project = cen` 來標記 Saanvi 的使用者。此外， AWS 授權模型不能匹配兩個值。如需詳細資訊，請參閱 IAM 和標記的規則 AWS STS。您必須改為手動指定她可以擔任的兩個角色。

若要使用此政策，請將斜體預留位置文字取代為您的帳戶資訊。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TutorialAssumeSpecificRoles",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::account-ID-without-hyphens:role/access-peg-
engineering",
                "arn:aws:iam::account-ID-without-hyphens:role/access-cen-
engineering"
            ]
        }
    ]
}
```

9. 使用 步驟 4：測試建立秘密 和 步驟 5：測試檢視秘密 中的程序。在另一個瀏覽器視窗中，確認 Saanvi 可以擔任這兩個角色。根據角色的標籤，檢查她是否只能為她的專案、團隊和成本中心建立秘密。同時確認她可以檢視工程團隊所擁有之任何秘密的詳細資訊，包括她剛建立的秘密。

步驟 7：測試更新和刪除秘密

連接至角色的 `access-same-project-team` 政策允許員工更新和刪除任何以其專案、團隊和成本中心加上標籤的秘密。在 Secrets Manager 中測試您的角色，確認您的許可如預期般運作。

測試更新和刪除含有與不含必要標籤的秘密

1. 使用下列其中一個 IAM 使用者身分登入:

   - `access-Arnav-peg-eng`
   - `access-Mary-peg-qas`

- access-Saanvi-uni-eng
- access-Carlos-uni-qas
- access-Nikhil-cen-eng

2. 切換至相符的角色：

   - access-peg-engineering
   - access-peg-quality-assurance
   - access-uni-engineering
   - access-peg-quality-assurance
   - access-cen-engineering

   如需「」中切換角色的詳細資訊 AWS Management Console，請參閱切換到角色 (主控台)。

3. 針對每個角色嘗試更新秘密描述，然後嘗試刪除下列秘密。如需詳細資訊，請參閱AWS Secrets Manager 使用者指南中的修改秘密以及刪除與還原秘密。

   每個角色的 ABAC 秘密更新和刪除行為

角色名稱	秘密名稱	預期行為
access-peg-engineering	test-access-peg-eng	允許
	test-access-uni-eng	已拒絕
	test-access-uni-qas	已拒絕
access-peg-quality-assurance	test-access-peg-qas	允許
	test-access-uni-eng	已拒絕
access-uni-engineering	test-access-uni-eng	允許
	test-access-uni-qas	已拒絕
access-peg-quality-assurance	test-access-uni-qas	允許

Summary

您現在已成功完成使用標籤進行以屬性為基礎的存取控制 (ABAC) 所需的所有步驟。您已學會如何定義標記策略。您已將該策略套用到主體和資源。您已建立並套用了實施 Secrets Manager 策略的政策。您也已了解當您新增專案和團隊成員時，ABAC 可以輕鬆擴展。因此，您可以使用測試角色登入 IAM 主控台，並體驗如何在 AWS中使用 ABAC 標籤。

> **Note**
>
> 您已新增僅在特定條件下允許執行動作的政策。如果您將不同的政策套用至具有更廣泛許可的使用者或角色，則動作可能不會受限為需要標記。例如，如果您使用受管理的原則授與使用者完整的系統`AdministratorAccess` AWS 管理權限，則這些原則不會限制該存取權。更多涉及多個政策時如何決定許可的詳細資訊，請參閱 決定是否允許或拒絕帳戶中的請求。

相關資源

如需相關資訊，請參閱下列資源：

- ABAC 是做什麼用的 AWS？
- AWS 全域條件內容索引鍵
- 建立 IAM 使用者 (主控台)
- 建立角色以委派許可給 IAM 使用者
- 標記 IAM 資源
- 使用標籤控制 AWS 資源的存取
- 切換到角色 (主控台)
- IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記

若要了解如何監控帳戶中的標籤，請參閱使用無伺服器工作流程和 Amazon E CloudWatch vents 監控 AWS 資源上的標籤變更。

IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記

屬性為基礎的存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 資源，包括 IAM 實體 (使用者或角色) 以及 AWS 資源。使用實體向其提出要求時 AWS，它們會成為主參與者，而這些主參與者包含標籤。

您也可以在擔任角色或與使用者聯合身分時傳遞工作階段標籤。然後您可以定義使用標籤條件索引鍵的政策，以根據主體的標籤來授與許可。當您使用標籤來控制對 AWS 資源的存取 ，您就是允許團隊和資源能夠用更少的 AWS 政策變更來增長。ABAC 政策比傳統政 AWS 策更靈活，因為它要求您列出每個單獨的資源。如需 ABAC 及其優於傳統政策的詳細資訊，請參閱 ABAC 是做什麼用的 AWS？。

如果您的公司使用 SAML 類型的身分提供者 (IdP) 來管理企業使用者身分，您可以將 SAML 屬性用於AWS中精細的存取控制。屬性可包含成本中心識別符、使用者電子郵件地址、部門分類和專案指派。當您將這些屬性做為工作階段標籤傳遞時，您接著便可以根據這些工作階段標籤來控制對 AWS 的存取。

如要將 SAML 屬性傳遞到您的工作階段主體來完成 ABAC 教學，請使用本主題中包含的變更，完成 IAM 教學課程：根據標籤定義存取 AWS 資源的許可 中的任務。

必要條件

如要執行步驟以使用 ABAC 的 SAML 工作階段標籤，您必須已具備以下項目：

- 對 SAML 類型 IdP 的存取，可讓您使用特定屬性建立測試使用者。
- 以具有管理許可的使用者身分登入的能力。
- 體驗在 AWS Management Console中建立並編輯 IAM 使用者、角色和政策。不過，如果您需要協助記住 IAM 管理程序，ABAC 教學課程會提供可讓您檢視 step-by-step 指示的連結。
- 在 IAM 中體驗設定 SAML 類型的 IdP。如要檢視更多詳細資訊及更詳細 IAM 文件的連結，請參閱 使用 AssumeRoleWith SAML 傳遞工作階段標記。

步驟 1：建立測試使用者

請跳過 步驟 1：建立測試使用者 中的說明。由於您的身分是定義在您的供應商中，您不需要為您的員工新增 IAM 使用者。

步驟 2：建立 ABAC 政策

遵循 步驟 2：建立 ABAC 政策 中的說明，在 IAM 中建立指定受管政策。

步驟 3：建立和設定 SAML 角色

當您使用適用於 SAML 的 ABAC 教學課程時，您必須執行其他步驟來建立角色、設定 SAML IdP 以及啟用存取權。 AWS Management Console 如需詳細資訊，請參閱 步驟 3：建立角色。

步驟 3A：建立 SAML 角色

建立單一角色，該角色信任您的 SAML 身分提供者，以及您在步驟 1 中建立的 test-session-tags 使用者。ABAC 教學會使用具備不同角色標籤的不同角色。因為您是從您的 SAML IdP 傳遞工作階段標籤，您只需要一個角色。若要了解如何建立 SAML 類型的角色，請參閱 為 SAML 2.0 聯合 (主控台) 建立角色。

將角色命名為 access-session-tags。將 access-same-project-team 許可政策連接到角色。編輯角色信任政策，以使用以下政策。如需如何編輯角色信任關係的詳細說明，請參閱 修改角色 (主控台)。

以下角色信任政策會允許您的 SAML 身分提供者和 test-session-tags 使用者擔任角色。當 SAML 身分提供者和該使用者擔任角色時，必須傳遞三個指定工作階段標籤。sts:TagSession 動作是允許傳遞工作階段標籤的必要項目。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSamlIdentityAssumeRole",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRoleWithSAML",
                "sts:TagSession"
            ],
            "Principal": {"Federated":"arn:aws:iam::123456789012:saml-
provider/ExampleCorpProvider"},
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/cost-center": "*",
                    "aws:RequestTag/access-project": "*",
                    "aws:RequestTag/access-team": [
                        "eng",
                        "qas"
                    ]
                },
                "StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/saml"}
            }
        }
    ]
}
```

此AllowSamlIdentityAssumeRole陳述式可讓「工程」與「品質保證」團隊的成員在 AWS 從「範例公司 IdP」聯盟時擔任此角色。ExampleCorpProvider SAML 供應商是在 IAM 中定義。管理員已設定 SAML 聲明，傳遞三個必要的工作階段標籤。聲明可以傳遞其他標籤，但必須要有這三個標籤。身分的屬性可針對 cost-center 和 access-project 標籤具備任何值。但是，access-team 屬性值必須與 eng 或 qas 相符，才能指出身分是位於 Engineering 或 Quality Assurance 團隊。

步驟 3B：設定 SAML IdP

設定您的 SAML IdP，將 cost-center、access-project 和 access-team 屬性做為工作階段標籤傳遞。如需更多詳細資訊，請參閱 使用 AssumeRoleWith SAML 傳遞工作階段標記。

若要將這些屬性做為工作階段標籤傳遞，請在您的 SAML 聲明中包含下列元素。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:cost-center">
  <AttributeValue>987654</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-project">
  <AttributeValue>peg</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-team">
  <AttributeValue>eng</AttributeValue>
</Attribute>
```

步驟 3C：啟用主控台存取

為您的聯合 SAML 使用者啟用主控台存取。如需更多詳細資訊，請參閱 啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console。

步驟 4：測試建立秘密

聯合到 AWS Management Console 使用access-session-tags角色。如需詳細資訊，請參閱 啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console。然後遵循 步驟 4：測試建立秘密 中的說明來建立秘密。搭配屬性使用不同的 SAML 身分，來匹配 ABAC 教學中指出的標籤。如需更多詳細資訊，請參閱 步驟 4：測試建立秘密。

步驟 5：測試檢視秘密

遵循 步驟 5：測試檢視秘密 中的說明來檢視您在上一個步驟中建立的秘密。搭配屬性使用不同的 SAML 身分，來匹配 ABAC 教學中指出的標籤。

步驟 6：測試可擴展性

遵循 步驟 6：測試可擴展性 中的說明來測試可擴展性。請使用下列屬性在您的 SAML 類型 IdP 中新增新的身分，來執行此作業：

- `cost-center = 101010`
- `access-project = cen`
- `access-team = eng`

步驟 7：測試更新和刪除秘密

遵循 步驟 7：測試更新和刪除秘密 中的說明來更新和刪除秘密。搭配屬性使用不同的 SAML 身分，來匹配 ABAC 教學中指出的標籤。

> **Important**
>
> 刪除您建立的所有秘密來避免產生費用。如需關於 Secrets Manager 定價的詳細資訊，請參閱 AWS Secrets Manager 定價。

Summary

您現在已成功完成使用 SAML 工作階段標籤和資源標籤進行許可管理必要的所有步驟。

> **Note**
>
> 您已新增僅在特定條件下允許執行動作的政策。如果您將不同的政策套用至具有更廣泛許可的使用者或角色，則動作可能不會受限為需要標記。例如，如果您使用受管理的原則授與使用者完整的系統`AdministratorAccess` AWS 管理權限，則這些原則不會限制該存取權。更多涉及多個政策時如何決定許可的詳細資訊，請參閱 決定是否允許或拒絕帳戶中的請求。

IAM 教學課程：允許使用者管理其憑證和 MFA 設定

您可以在 [安全性認證] 頁面上允許使用者管理自己的多重要素驗證 (MFA) 裝置和認證。您可以使用 AWS Management Console 為使用者設定憑證 (存取金鑰、密碼、簽章憑證和 SSH 公有金鑰)、刪除或停用不需要的憑證，以及啟用 MFA 裝置。當使用者人數較少時，它十分好用。但隨著使用者人數增

加，這個任務很快會變得非常耗時。本教程是為您介紹如何實現這些最佳實務，而不給您的管理員帶來負擔。

本教學課程說明如何允許使用者存取 AWS 服務，但只有在使用 MFA 登入時才能存取服務。如果未使用 MFA 裝置登入，則使用者無法存取其他服務。

此工作流程有三個基本步驟。

步驟 1：建立政策以實施 MFA 登入

建立一個客戶受管政策，其禁止除少數 IAM 動作之外的所有動作。這些例外允許使用者變更自己的認證，並在 [安全性認證] 頁面上管理其 MFA 裝置。如需存取該頁面的詳細資訊，請參閱 IAM 使用者如何變更他們自己的密碼 (主控台)。

步驟 2：將政策連接到您的測試使用者群組

建立一個使用者群組，其成員在使用 MFA 登入時擁有對所有 Amazon EC2 動作的完全存取權。若要建立這類使用者群組，請附加呼叫的 AWS 受管理原則`AmazonEC2FullAccess`和您在第一個步驟中建立的客戶管理政策。

步驟 3：測試您的使用者存取權限

以測試使用者身分登入，驗證在使用者建立 MFA 裝置之前對 Amazon EC2 的存取被封鎖。之後使用者可以使用該裝置登入。

必要條件

若要執行此教學課程中的步驟，您必須具備以下內容：

- 您可以使用具有管理許可的 IAM 使用者身分登入。 AWS 帳戶
- 您在步驟 1 中輸入政策的帳戶 ID 號碼。

  若要尋找您的帳戶 ID 編號，請在頁面頂部的導覽列上，選擇 Support (支援)，然後選擇 Support Center (支援中心)。您可以在此頁面的 Support (支援) 選單下尋找您的帳戶 ID。
- 虛擬 (軟體式) MFA 裝置、FIDO 安全性金鑰，或硬體式 MFA 裝置。
- 一個做為群組成員的測試 IAM 使用者，如下所示：

建立使用者		建立和設定使用者群組帳戶		
使用者名稱	其他說明	使用者群組名稱	新增使用者為成員	其他說明
MFAUser	僅選擇 啟用主控台存取 (選用) 選項，並指定一組密碼。	EC2MFA	MFAUser	請勿連接任何政策或者授與許可給此使用者群組。

步驟 1：建立政策以實施 MFA 登入

首先建立一個 IAM 客戶受管政策，除 IAM 使用者對管理其自有憑證和 MFA 裝置所需的許可權外，該政策拒絕其他所有許可。

1. 以具有系統 AWS 管理員認證的使用者身分登入管理主控台。若要遵守 IAM 最佳做法，請勿使用您的登入 AWS 帳戶根使用者 資料登入。

   > **Important**
   >
   > IAM 最佳實務建議您要求人類使用者與身分識別提供者的聯合使用，以 AWS 使用臨時登入資料存取，而不是使用具有長期登入資料的 IAM 使用者。

2. 在 https://console.aws.amazon.com/iam/ 中開啟 IAM 主控台。

3. 在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。

4. 選擇 JSON 標籤並從下列 JSON 政策文件複製文字：AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

5. 將此政策文字貼上至 JSON 文字方框中。解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇下一步。

   > **Note**
   >
   > 您可以隨時切換視覺化編輯器與 JSON 選項。但是，上方的政策會包含 `NotAction` 元素，其不受視覺化編輯器支援。針對此政策，您會在 Visual editor (視覺化編輯器) 標籤上看到一個通知。返回 JSON 以繼續處理此政策。

本範例政策不允許使用者在首次登入 AWS Management Console 時重設密碼。在新使用者登入並重設密碼之前，建議您不要將許可授予給他們。

6. 在檢視與建立頁面上，針對政策名稱輸入 **Force_MFA**。針對政策說明，在標籤區域中輸入 **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.**，您可以選擇性地新增標籤鍵值對至客戶管理政策。檢視政策授與的許可，然後選擇建立政策來儲存您的工作。

   新的政策會出現在受管政策清單中，並且已準備好連接。

步驟 2：將政策連接到您的測試使用者群組

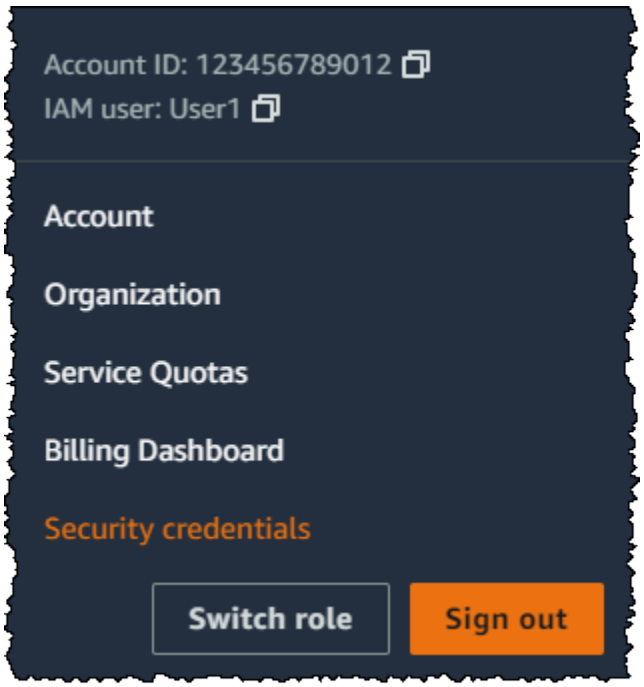
接下來，您將兩個政策連接到 IAM 使用者 群組，將使用該群組授與以 MFA 提供保護的許可。

1. 在導覽窗格中，選擇 User groups (使用者群組) 。
2. 在搜尋方塊中，輸入 **EC2MFA**，然後在清單中選擇群組名稱 (非核取方塊)。
3. 選擇 許可 標籤、選擇 新增許可，然後選擇 連接政策。
4. 在 Attach permission policies to EC2MFA group (將許可政策連接到 EC2MFA 群組) 頁面上，在搜尋方塊中輸入 **EC2Full**。然後選取清單FullAccess中亞馬遜 EC2 旁邊的核取方塊。目前先不要儲存您的變更。
5. 在搜尋方塊中，輸入 **Force**，然後在清單中選擇 Force_MFA 旁的核取方塊。
6. 選擇連接政策。

步驟 3：測試您的使用者存取權限

在本教程的這一部分中，您以測試使用者身分登入並確認政策是否正常運作。

1. **MFAUser**使用您在上一節中指派的密碼登入 AWS 帳戶 身分。使用 URL：`https://<alias or account ID number>.signin.aws.amazon.com/console`
2. 選擇 EC2 來開啟 Amazon EC2 主控台，並確認使用者沒有許可執行任何操作。
3. 在右上方的導覽列中，選擇您的 MFAUser 使用者名稱，然後選擇 安全憑證 。



4. 現在新增一個 MFA 裝置。在 Multi-Factor Authentication (MFA) (多重要素驗證 (MFA)) 區段，選擇 Assign MFA device (指派 MFA 裝置)。

   > **Note**
   >
   > 您可能會收到錯誤，告知您未被授權執行 `iam:DeleteVirtualMFADevice`。如果有人先前開始將虛擬 MFA 裝置指派給這個使用者但取消程序，就可能發生此錯誤。若要繼續，您或其他管理員必須刪除該使用者未經指派的現有虛擬 MFA 裝置。如需詳細資訊，請參閱 我沒有授權執行：IAM：DeleteVirtualMFA 設備。

5. 對於本教程，我們使用一個虛擬 (軟體式) MFA 裝置，例如，手機上的 Google Authenticator 應用程式。選擇 Authenticator app (驗證器應用程式)，然後按 Next (下一步)。

   IAM 將產生並顯示虛擬 MFA 裝置的配置資訊，包括 QR 碼圖形。此圖形是私密組態金鑰的表示形式，適用於不支援 QR 碼的裝置上的手動輸入。

6. 開啟您的虛擬 MFA 應用程式。(如需可以作為託管虛擬 MFA 裝置的應用程式清單，請查看虛擬 MFA 應用程式。) 如果虛擬 MFA 應用程式支援多個帳戶 (多個虛擬 MFA 裝置)，請選擇對應的選項以建立新帳戶 (新的虛擬 MFA 裝置)。

7. 判定 MFA 應用程式是否支援 QR 碼，然後執行以下操作之一：

   - 從精靈中，選擇 Show QR code (顯示 QR 碼)。然後使用應用程式來掃描 QR 碼。例如，您可選擇相機圖示或選擇與 Scan code (掃描碼) 類似的選項，然後使用裝置的相機掃描碼。

- 在 Set up device (設定裝置) 精靈中，選擇 Show secret key (顯示私密金鑰)，然後在您的 MFA 應用程式中輸入私密金鑰。

完成操作後，虛擬 MFA 裝置會開始產生一次性密碼。

8. 在 Set up device (設定裝置) 精靈中的 Enter the code from your authenticator app. (輸入來自您的驗證器應用程式的代碼。) 方塊中，輸入虛擬 MFA 裝置上目前顯示的一次性密碼。選擇 Register MFA (註冊 MFA)。

> **Important**
>
> 產生代碼之後立即提交您的請求。如果在產生代碼後等待太久才提交請求，MFA 裝置將成功與使用者建立關聯。但是，MFA 裝置並不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

虛擬 MFA 裝置現在已準備就緒，可與 AWS.

9. 登出主控台，然後再次以 **`MFAUser`** 身分登入。這次 AWS 會提示您從手機輸入 MFA 代碼。收到該代碼後，請將代碼輸入方塊中，然後選擇 Submit (提交)。
10. 選擇 EC2 來再次開啟 Amazon EC2 主控台。請注意，這次您可以看到所有資訊，並且可以執行所需的任何動作。如果您以此使用者身分前往任何其他主控台，您會看到存取遭拒訊息。原因在於此教學中的政策僅會將存取權授與 Amazon EC2。

相關資源

如需其他資訊，請參閱以下主題：

- 在中使用多因素身份驗證（MFA）AWS
- 為中的使用者啟用 MFA 裝置 AWS
- 透過您的 IAM 登入頁面來使用 MFA 裝置

IAM 身分 (使用者、使用者群組和角色)

> **Tip**
>
> 登入時遇到問題 AWS嗎？ 請確定您位於正確的登入頁面。
>
> - 若要以 AWS 帳戶根使用者 (帳戶擁有者) 的身分登入，請使用您在建立 AWS 帳戶.
> - 若要以 IAM 使用者的身分登入，請使用帳戶管理員提供給您的憑證登入 AWS。
> - 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。
>
> 如需登入教學課程，請參閱《AWS 登入 使用者指南》中的如何登入 AWS。

> **Note**
>
> 如需請求支援，請勿使用此頁面上的 Feedback (意見回饋) 連結。 AWS 文件團隊會收到您輸入的意見反應，而不是 Sup AWS port 人員。請改為選擇此頁面頂端的 Contact Us (聯絡我們) 連結。我們會提供各種資源的連結，協助您取得所需的支援。

帳戶的 AWS 帳戶根使用者 或管理使用者可以建立 IAM 身分。IAM 身分提供對 AWS 帳戶的存取權限。IAM 使用者群組是以單位形式管理的 IAM 使用者集合。IAM 身分代表人類使用者或程式設計式工作負載，並可進行身分驗證，然後授權在 AWS中執行動作。每個 IAM 身分可以與一或多個政策相關聯。策略會決定使用者、角色或使用者群組成員可以執行的動作、對哪些 AWS 資源以及在何種情況下可執行的動作。

AWS 帳戶 根使用者

當您第一次建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務 和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。

> **Important**
>
> 強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需檢視需要您以根使用者身分登入的任務完整清單，請參閱 需要根使用者憑證的任務。

IAM 使用者

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。最佳實務建議盡可能依賴暫時憑證，而不是建立擁有長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。在建立存取金鑰前，請檢閱長期存取金鑰的替代方案。如果您有需要存取金鑰的特定使用案例，我們建議您視需要更新存取金鑰。如需詳細資訊，請參閱 對於需要長期憑證的使用案例，請視需要更新存取金鑰。若要將 IAM 使用者新增至您的 AWS 帳戶，請參閱在您的中建立 IAM 使用者 AWS 帳戶。

> **Note**
>
> 我們的安全最佳實務是建議您透過聯合身分來提供資源存取權限，而不是建立 IAM 使用者。若要了解需要 IAM 使用者的特定情形，請參閱建立 IAM 使用者 (而非角色) 的時機。

IAM 使用者群組

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程序變得更為容易。例如，您可以擁有一個名為 IAMPublishers 的群組，並為該群組提供發佈工作負載通常所需的許可類型。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但未與特定的人員建立關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需有關角色使用方法的詳細資訊，請參閱 使用 IAM 角色。

使用暫時性憑證的 IAM 角色適用於下列情況：

- 聯合身分使用者存取 – 若要向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資

訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。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)。若要了解針對跨帳戶存取權的以角色為基礎和以資源為基礎之政策間的差異，請參閱 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。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。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，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 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至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。如需更多資訊，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。

IAM 中的暫時性憑證

最佳實務是，同時對人類使用者和工作負載使用臨時性憑證。暫時性憑證主要用於 IAM 角色，但也有其他用途。您可以請求具有比標準 IAM 使用者更受限的許可權組的暫時性憑證。這樣可以防止意外執行更受限制憑證不允許的任務。暫時性憑證的優點是它們會在一段時間後自動過期。您可以控制憑證有效的持續時間。

何時使用 IAM Identity Center 使用者？

我們建議所有人類使用者都使用 IAM 身分中心來存取 AWS 資源。IAM 身分中心在以 IAM 使用者身分存取 AWS 資源方面大幅改善。IAM Identity Center 提供：

- 一個身分和指派的中央集合
- 存取整個 AWS 組織中的帳戶
- 連接至您的現有身分提供者
- 臨時憑證
- 多重要素驗證 (MFA)
- 適用於終端使用者的自助式 MFA 組態
- MFA 使用的管理強制執行
- 單一登入所有 AWS 帳戶 權利

如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center。

何時建立 IAM 使用者 (而不是角色)

建議您僅將 IAM 使用者用於聯合身分使用者不支援的使用案例。以下是部分使用案例：

- 無法使用 IAM 角色的工作負載 - 您可以從需要存取 AWS的位置執行工作負載。在某些情況下，您無法使用 IAM 角色提供臨時登入資料，例如 WordPress 外掛程式。在這些情況下，請將 IAM 使用者長期存取金鑰用於該工作負載，對 AWS進行身分驗證。
- 第三方用 AWS 戶端 — 如果您使用的工具不支援 IAM 身分中心的存取，例如未託管的第三方用 AWS 戶端或廠商 AWS，請使用 IAM 使用者長期存取金鑰。
- AWS CodeCommit 存取 — 如果您使 CodeCommit 用儲存程式碼，您可以使用具有 SSH 金鑰或服務特定登入資料的 IAM 使用者，對儲存庫 CodeCommit 進行驗證。除了將 IAM Identity Center 中的使用者用於一般身分驗證之外，我們也建議您這樣做。IAM 身分中心的使用者是員工中需要存取您的雲端應用程式 AWS 帳戶 或雲端應用程式的人員。若要讓使用者在不設定 IAM 使用者的情況下 CodeCommit 存取您的儲存庫，您可以設定git-remote-codecommit公用程式。如需 IAM 和的詳細資訊 CodeCommit，請參閱將 IAM 搭配使用 CodeCommit：Git 登入資料、安全殼層金鑰和 AWS 存取金鑰。如需有關設定公用git-remote-codecommit程式的詳細資訊，請參閱《AWS CodeCommit 使用指南》中的使用旋轉認證連線到 AWS CodeCommit 儲存庫。
- Amazon Keyspaces (適用於 Apache Cassandra) 存取 – 在無法使用 IAM Identity Center 中的使用者的情況下，例如為了測試 Cassandra 相容性，您可以將 IAM 使用者搭配服務特定憑證使用，以便使

用 Amazon Keyspaces 進行身分驗證。IAM 身分中心的使用者是員工中需要存取您的雲端應用程式 AWS 帳戶 或雲端應用程式的人員。您也可以使用暫時性憑證連線到 Amazon Keyspaces。如需詳細資訊，請參閱《Amazon Keyspaces (適用於 Apache Cassandra) 開發人員指南》中的透過 IAM 角色和 SigV4 外掛程式使用暫時性憑證連線到 Amazon Keyspaces。

- 緊急存取：無法存取身分提供者且必須在 AWS 帳戶中採取動作的情況下。您可以將建立緊急存取 IAM 使用者列入彈性計劃中的一環。建議您使用多重要素驗證 (MFA) 嚴格控管緊急使用者憑證並加以保護。

何時建立 IAM 角色 (而不是使用者)

在以下情況下建立 IAM 角色：

您正在建立在 Amazon 彈性運算雲端 (Amazon EC2) 執行個體上執行的應用程式，並且該應用程式向其發出請求 AWS。

不要建立 IAM 使用者並將使用者的憑證傳遞給應用程式或將憑證嵌入應用程式中。反之，請建立您連接至 EC2 執行個體的 IAM 角色，將暫時安全認證給予執行個體上執行的應用程式。當應用程式在中使用這些認證時 AWS，它可以執行附加至該角色的原則所允許的所有作業。如需詳細資訊，請參閱 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可。

您正在建立在行動電話上執行的應用程式，並向 AWS發出請求。

不要建立 IAM 使用者並隨應用程式發佈使用者的存取金鑰。反之，請使用 Login with Amazon、Amazon Cognito、Facebook 或 Google 等身分提供者對使用者進行身分驗證，並將使用者對應到 IAM 角色。該應用程式可以使用該角色來取得暫時安全憑證，該憑證具有連接到角色的政策所指定的許可。如需詳細資訊，請參閱下列內容：

- Amazon Cognito 使用者指南
- OIDC 聯盟

您公司中的使用者會在您的公司網路中經過驗證，並且希望能夠使用 AWS 而不必再次登入 — 也就是說，您想要允許使用者聯合到。 AWS

不要建立 IAM 使用者。設定企業識別系統與之間的同盟關係 AWS。您可利用兩種方式進行：

- 如果貴公司的身分系統與 SAML 2.0 相容，您可以在公司的身分系統與 AWS. 如需詳細資訊，請參閱 SAML 2.0 聯合身分。
- 建立並使用自訂 Proxy 伺服器，將使用者身分從企業轉譯為提供臨時 AWS 安全登入資料的 IAM 角色。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

比較 AWS 帳戶根使用者 登入資料和 IAM 使用者憑證

root 使用者是帳戶擁有者，並在建立 AWS 帳戶 時建立。其他類型的使用者 (包括 IAM 使用 AWS IAM Identity Center 者和使用者) 是由根使用者或帳戶的管理員建立的。所有 AWS 使用者都有安全認證。

根使用者憑證

帳戶擁有者的憑證可以完整存取帳戶中的所有資源。您無法使用 IAM 政策來明確拒絕根使用者的資源存取權。您只能使用 AWS Organizations 服務控制策略（SCP）來限制成員帳戶的根用戶的權限。因此，我們建議您在 IAM 身分中心建立管理使用者，以用於日常 AWS 工作。然後，保護根使用者憑證，只用它們來執行需要您以根使用者身分登入的少數帳戶和服務管理任務。如需檢視這些任務的清單，請參閱 需要根使用者憑證的任務。若要了解如何在 IAM Identity Center 中設定日常使用的管理員，請參閱《IAM Identity Center 使用者指南》中的入門。

IAM 憑證

IAM 使用者是您在其中建立的實體 AWS ，代表使用 IAM 使用者與 AWS 資源互動的個人或服務。這些使用者是您 AWS 帳戶 內部具有特定自訂權限的身分識別。例如，您可以建立 IAM 使用者，並授予他們在 IAM Identity Center 中建立目錄的許可。IAM 使用者擁有長期登入資料，他們可以使用這些登入資料 AWS Management Console，或 AWS 使用 AWS CLI 或 AWS API 以程式設計方式存取。有 step-by-step 關 IAM 使用者如何登入的指示 AWS Management Console，請參閱登入使用者指南中的 AWS Management Console 以 IAM 使用者身分AWS 登入。

一般而言，我們建議您避免建立 IAM 使用者，因為他們擁有長期憑證，例如使用者名稱和密碼等。而是要求人類使用者在存取時使用臨時認證 AWS。您可以為您的人類使用者使用身分識別提供者，AWS 帳戶 藉由假設 IAM 角色提供臨時登入資料來提供聯合存取權。如果是集中式存取管理，我們建議您使用 IAM Identity Center 管理您帳戶的存取權和這些帳戶內的許可。您可以使用 IAM Identity Center 管理使用者身分，或從外部身分提供者管理 中使用者身分的存取許可。如需了解更多資訊，請參閱《IAM Identity Center 使用者指南》中的什麼是 IAM Identity Center。

AWS 帳戶根使用者

首次建立 Amazon Web Services (AWS) 帳戶時，您會從單一登入身分開始，該身分可以完整存取帳戶中的所有 AWS 服務和資源。此身分識別稱為 AWS 帳號 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。

> **Important**
>
> 強烈建議您不使用根使用者處理日常任務，並且遵循 AWS 帳戶的根使用者最佳實務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需檢視需要您以根使用者身分登入的任務完整清單，請參閱 需要根使用者憑證的任務。

下列主題將詳細說明與根使用者相關的管理任務。

任務

- 為您的 AWS 帳戶根使用者 （控制台）啟用多因素身份驗證
- 更改密碼 AWS 帳戶根使用者
- 重設遺失或忘記的根使用者密碼
- 建立根使用者的存取金鑰
- 刪除根使用者的存取金鑰
- 需要根使用者憑證的任務
- 使用根使用者來疑難排解問題
- 相關資訊

為您的 AWS 帳戶根使用者 （控制台）啟用多因素身份驗證

多重要素驗證 (MFA) 是一種簡單而有效的機制，可增強您的安全性。第一個因素-您的密碼-是您記住的秘密，也稱為知識因素。其他因素可能是擁有因素（您擁有的東西，例如安全密鑰）或固有因素（您所屬的東西，例如生物識別掃描）。為了提高安全性，我們強烈建議您設定多重要素驗證 (MFA) 以協助保護您 AWS 的資源。

您可以為 AWS 帳戶根使用者 和 IAM 使用者啟用 MFA。當您為根使用者啟用 MFA 時，它只會影響根使用者認證。如需如何為 IAM 使用者啟用 MFA 的詳細資訊，請參閱中的為 IAM 使用者啟用 MFA 裝置。 AWS

在為 root 使用者啟用 MFA 之前，請檢閱並更新您的帳戶設定和聯絡資訊，以確保您可以存取電子郵件和電話號碼。如果您的 MFA 裝置遺失、遭竊或無法運作，您仍然可以使用該電子郵件和電話號碼驗證身分，以根使用者身分登入。如需了解如何使用其他身分驗證方法登入的詳細資訊，請參閱MFA 裝置遺失或停止運作時怎麼辦？。若要停用這項功能，請聯絡 AWS Support。

主題

- 根使用者可用的 MFA 類型
- 啟用 AWS 帳戶根使用者 (控制台) 的金鑰或安全金鑰
- 針對 AWS 帳戶根使用者 啟用虛擬 MFA 裝置 (主控台)
- 啟用 AWS 帳戶根使用者 （ 控制台 ）的硬件 TOTP 令牌

根使用者可用的 MFA 類型

AWS 為您的 root 使用者支援下列 MFA 類型：密碼金鑰和安全金鑰、虛擬驗證器應用程式以及硬體 TOTP 權杖。

密碼金鑰和安全金鑰

AWS Identity and Access Management 支援 MFA 的密碼金鑰和安全金鑰。根據 FIDO 標準，密碼金鑰使用公開金鑰加密技術來提供比密碼更安全的強式、防網路釣魚驗證。 AWS 支援兩種類型的密碼金鑰：裝置繫結的密碼金鑰 (安全金鑰) 和同步的金鑰。

- 安全密鑰：這些是物理設備，例如 YubiKey，用作身份驗證的第二個因素。
- 同步的密鑰：這些使用提供商（例如谷歌，蘋果，Microsoft 帳戶以及第三方服務（如 1Password，Dashlane 和 Bitwarden）的憑據管理器作為第二個因素。

您可以使用內建的生物特徵驗證器 (例如 Apple 上的 Touch ID MacBooks 和電腦上的 Windows Hello 臉部辨識) 來解鎖憑證管理員並登入 AWS。密碼是由您選擇的供應商使用您的指紋、臉孔或裝置 PIN 碼建立的。您可以跨裝置同步密碼金鑰 AWS，以促進登入，進而增強可用性和可復原性。

FIDO Alliance 維護與 FIDO 規範相容的所有經 FIDO 認證的產品的清單。單一金鑰或安全金鑰可支援多個根使用者帳戶和 IAM 使用者。如需啟用密碼金鑰和安全金鑰的詳細資訊，請參閱啟用 AWS 帳戶根使用者 (控制台) 的金鑰或安全金鑰。

虛擬驗證器應用程式

虛擬身份驗證器應用程序在手機或其他設備上運行，並模擬物理設備。虛擬驗證器應用程式實作以時間為基礎的一次性密碼 (TOTP) 算法，並且支援在單台裝置上使用多個權杖。登入期間出現提示時，使用者必須輸入裝置的有效代碼。分配給用戶的每個令牌必須是唯一的。用戶無法從其他用戶的令牌中鍵入代碼進行身份驗證。

我們強烈建議您在等待硬體的購買核准或等待硬體就定位時，使用虛擬 MFA 裝置。如需可作為虛擬 MFA 裝置使用的一些受支援應用程式的清單，請參閱 M ulti-Factor Authentication (M FA)。如需使用設定虛擬 MFA 裝置的指示 AWS，請參閱針對 AWS 帳戶根使用者 啟用虛擬 MFA 裝置 (主控台)。

硬件 TOTP 令牌

硬體裝置會根據基於時間的一次性密碼 (TOTP) 演算法產生一個六位數的數字代碼。使用者必須在登入期間在第二個網頁上輸入裝置中的有效程式碼。每個指派給使用者的 MFA 裝置都必須是唯一的。使用者無法輸入另一個使用者裝置的代碼來進行身分驗證。如需支援硬體 MFA 裝置的相關資訊，請參閱 Multi-Factor Authentication (MFA)。如需使用設定硬體 TOTP 權杖的指示 AWS，請參閱啟用 AWS 帳戶根使用者（控制台）的硬件 TOTP 令牌。

如果您想要使用實體 MFA 裝置，建議您使用 FIDO 安全金鑰作為硬體 TOTP 裝置的替代方案。FIDO 安全金鑰具有無電池需求、網路釣魚阻力的優點，而且它們支援單一裝置上的多個 root 和 IAM 使用者，以增強安全性。

啟用 AWS 帳戶根使用者 (控制台) 的金鑰或安全金鑰

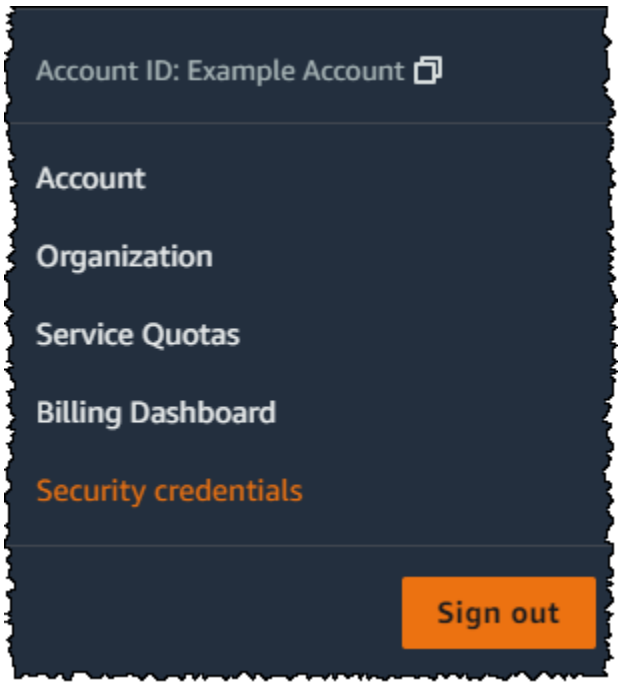
您可以從 AWS Management Console 唯一的，而不是從 AWS CLI 或 AWS API 為 root 使用者設定和啟用金鑰。

啟用 root 使用者的金鑰或安全金鑰 (主控台)

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 IAM 主控台。在下一頁中，輸入您的密碼。

   > **Note**
   >
   > 根使用者無法登入 以 IAM 使用者身分登入 頁面。如果您看到 以 IAM 使用者身分登入 頁面，請選擇頁面底部附近的 使用根使用者電子郵件登入。如需以 root 使用者身分登入的說明，請參閱《使用指南》中的「以 root 使用者身分登入」AWS 登入 。 AWS Management Console

2. 在導覽列右側選擇您的帳戶名稱，然後選擇 安全憑證 。如有需要，選擇 Continue to Security Credentials (繼續至安全憑證)。



3. 在您的 root 使用者 [我的安全認證] 頁面的 [多重要素驗證 (MFA)] 下，選擇 [指派 M FA 裝置]。
4. 在 MFA 裝置名稱頁面上，輸入裝置名稱，選擇 [金鑰] 或 [安全金鑰]，然後選擇 [下一步]。
5. 在 [設定裝置] 上，設定您的金鑰。使用生物辨識資料 (例如臉部或指紋) 建立金鑰、使用裝置 PIN 碼，或將 FIDO 安全金鑰插入電腦的 USB 連接埠並輕觸，以建立金鑰。
6. 依照瀏覽器上的指示，選擇金鑰提供者，或是您要儲存金鑰的地方，以便在裝置上使用。
7. 選擇繼續。

您現在已經註冊了您的金鑰，以便與之搭配 AWS使用。下次您使用 root 使用者認證登入時，您必須使用金鑰進行驗證，才能完成登入程序。

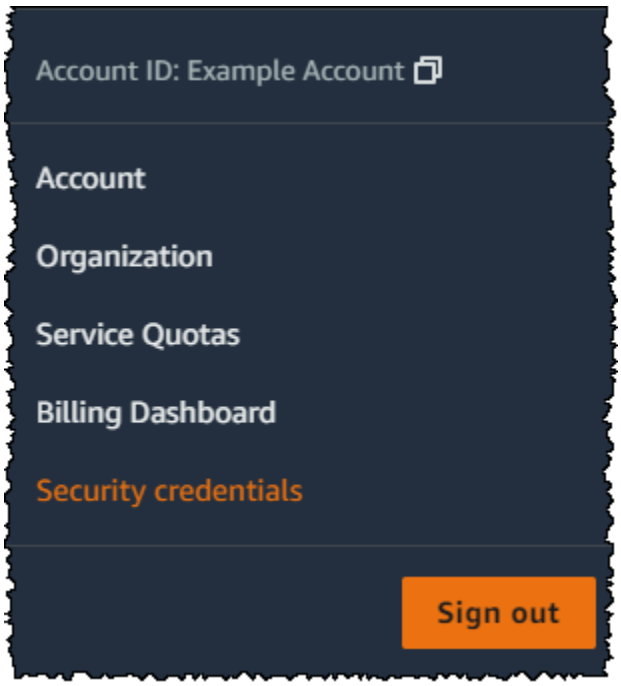
如需疑難排解 FIDO 安全性金鑰問題的說明，請參閱對 FIDO 安全性金鑰進行故障診斷。

針對 AWS 帳戶根使用者 啟用虛擬 MFA 裝置 (主控台)

您可以使用 AWS Management Console 為根使用者設定和啟用虛擬 MFA 裝置。若要啟用的 MFA 裝置 AWS 帳戶，您必須使用根 AWS 使用者認證登入。

設定和啟用虛擬 MFA 裝置以與根使用者一起使用 (主控台)

1. 登入 AWS Management Console。
2. 在導覽列右側，選擇您的帳戶名稱，然後選擇 安全憑證 。如有需要，選擇 Continue to Security Credentials (繼續至安全憑證)。



3. 在 Multi-Factor Authentication (MFA) (多重要素驗證 (MFA)) 區段中，選擇 Assign MFA device (指派 MFA 裝置)。
4. 在精靈中輸入 Device name，然後選擇 驗證器應用程式，再選擇 下一步。

   IAM 將產生並顯示虛擬 MFA 裝置的配置資訊，包括 QR 碼圖形。此圖形是私密組態金鑰的表示形式，適用於不支援 QR 代碼的裝置上的手動輸入。
5. 在裝置上開啟虛擬 MFA 應用程式。

   如果虛擬 MFA 應用程式支援多個虛擬 MFA 裝置或帳戶，請選擇對應的選項以建立新的虛擬 MFA 裝置或帳戶。
6. 設定應用程式的最簡單方法是使用該應用程式掃描 QR 條碼。如果您無法掃描代碼，則可以手動輸入組態資訊。IAM 產生的 QR 碼和秘密設定金鑰與您的帳戶相關聯，無法 AWS 帳戶 與其他帳戶搭配使用。但是，如果您無法存取原始 MFA 裝置，則他們可以重複使用，以便為您的帳戶設定新的 MFA 裝置。

   - 若要使用 QR 碼設定虛擬 MFA 裝置，請從精靈中選擇 Show QR code (顯示 QR 碼)。然後，遵循應用程式說明來掃描代碼。例如，您可能需要選擇相機圖示，或選擇與 Scan account barcode (掃描帳戶代碼) 相似的命令，然後使用裝置的相機掃描 QR 碼。
   - 在 Set up device (設定裝置) 精靈中，選擇 Show secret key (顯示私密金鑰)，然後在您的 MFA 應用程式中輸入私密金鑰。

> **Important**
>
> 對 QR 條碼或私密組態金鑰進行安全備份，或確保為您的帳戶啟用多台 MFA 裝置。您最多可以向您 AWS 帳戶根使用者 和 IAM 使用者註冊八個 MFA 裝置，其中包括目前支援的 MFA 類型的任何組合。虛擬 MFA 裝置可能無法使用，例如，如果您遺失託管在虛擬 MFA 裝置的智慧型手機。如果發生這種情況且您沒有連接到使用者的其他 MFA 裝置，甚至無法透過 復原根使用者 MFA 裝置 登入您的帳戶，您將無法登入您的帳戶，您必須聯絡客戶服務以移除該帳戶的 MFA 保護。

裝置開始產生六位數的號碼。

7. 在精靈中的 MFA code 1 (MFA 代碼 1) 方塊內，輸入虛擬 MFA 裝置上目前顯示的一次性密碼。請等待 30 秒，裝置將產生新的一次性密碼。然後將第二個一次性密碼輸入 MFA code 2 (MFA 代碼 2) 方塊中。選擇 Add MFA (新增 MFA)。

> **Important**
>
> 產生代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

該設備已準備好與一起使用 AWS。如需與 AWS Management Console一起使用 MFA 的詳細資訊，請參閱 透過您的 IAM 登入頁面來使用 MFA 裝置。

啟用 AWS 帳戶根使用者 （控制台）的硬件 TOTP 令牌

您可以從根使用者設定和啟用實體 MFA 裝置， AWS Management Console 而不是從 AWS CLI 或 AWS API。

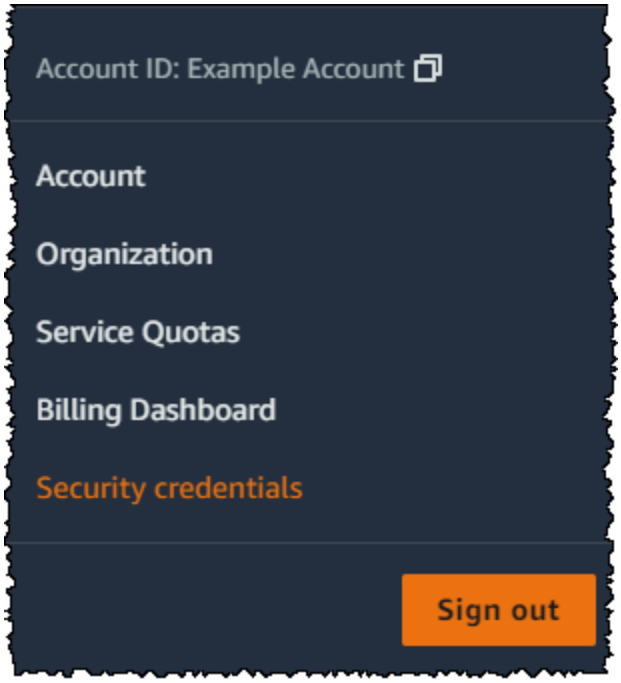
> **Note**
>
> 您可能會看到不同的文字，例如使用 MFA 登入和對您的身分驗證裝置進行疑難排解。不過，其功能是相同的。在任一情況下，若無法使用其他身分驗證要素來驗證您的帳戶電子郵件地址和電話號碼，請聯絡 AWS Support 停用您的 MFA 設定。

針對根使用者啟用 MFA 裝置 (主控台)

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 IAM 主控台。在下一頁中，輸入您的密碼。

   > **Note**
   >
   > 根使用者無法登入 以 IAM 使用者身分登入 頁面。如果您看到 以 IAM 使用者身分登入 頁面，請選擇頁面底部附近的 使用根使用者電子郵件登入。如需以 root 使用者身分登入的說明，請參閱《使用指南》中的「以 root 使用者身分登入」AWS 登入 。 AWS Management Console

2. 在導覽列右側選擇您的帳戶名稱，然後選擇 安全憑證 。如有需要，選擇 Continue to Security Credentials (繼續至安全憑證)。



3. 展開 Multi-factor authentication (MFA) (多重要素驗證 (MFA)) 區段。
4. 選擇 Assign MFA device (指派 MFA 裝置)。
5. 在精靈中，輸入一個裝置名稱，然後選取 Hardware TOTP token (硬體 TOTP 權杖)，再選擇 Next (下一步)。
6. 在 Serial number (序號) 方塊中，輸入在 MFA 裝置背面找到的序號。
7. 在 MFA code 1 (MFA 代碼 1) 方塊中，輸入 MFA 裝置顯示的六位數字。您可能需要按下裝置正面的按鈕以顯示數字。

8. 當裝置在重新整理代碼時，等候 30 秒時間後，在 MFA code 2 (MFA 代碼 2) 方塊中輸入六位數字。您可能需要再次按下裝置正面的按鈕以顯示第二個數字。
9. 選擇 Add MFA (新增 MFA)。現在，MFA 裝置已與 AWS 帳戶建立關聯。

> **Important**
>
> 產生驗證代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置卻變成不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

下次使用根使用者憑證登入時，您必須輸入 MFA 裝置的代碼。

更改密碼 AWS 帳戶根使用者

您可以在安全憑證或帳戶頁面上變更電子郵件地址和密碼。您也可以選擇忘記密碼？ 在 AWS 登錄頁面上重置您的密碼。

若要變更 root 使用者的密碼，您必須以 AWS 帳戶根使用者 而非 IAM 使用者身分登入。若要了解如何重設忘記的根使用者密碼，請參閱 重設遺失或忘記的根使用者密碼。

為了保護您的密碼，請務必遵循這些最佳實務：

- 定期變更您的密碼。
- 請保持您密碼的私密性，因為任何獲知您的密碼的人員都能存取您的帳戶。
- 使用與其他網站不同的密碼。 AWS
- 避免使用容易猜中的密碼。這些密碼包含，例如 `secret`、`password`、`amazon` 或 `123456`。另外還要避免使用字典字詞、您的名字、電子郵件地址或其他可以輕鬆取得的個人資訊。

AWS Management Console

變更根使用者的密碼

> **最低許可**
>
> 若要執行下列步驟，您至少必須擁有下列 IAM 許可：
>
> - 您必須以 AWS 帳戶 root 使用者身分登入，不需要額外的 AWS Identity and Access Management (IAM) 許可。您無法以 IAM 使用者或角色的身分執行這些步驟。

1. 使用您 AWS 帳戶的電子郵件地址和密碼登入 AWS 帳戶根使用者。AWS Management Console
2. 在主控台的右上角，選擇您的帳戶名稱或號碼，然後選擇帳戶。
3. 在帳戶頁面的帳戶設定旁，選擇編輯。出於安全目的，系統會提示您再次進行身分驗證。

> **Note**
>
> 如果您看不到編輯選項，則可能是您未以帳戶根使用者的身分登入。在以 IAM 使用者或角色的身分登入時，您無法修改帳戶設定。

4. 在更新帳戶設定頁面上，選擇密碼下方的編輯。
5. 在更新您的密碼頁面上，填寫目前的密碼、新密碼和確認新密碼欄位。

> **Important**
>
> 確保選擇強式密碼。雖然您可以設定 IAM 使用者的帳戶密碼政策，但該政策不適用於根使用者。

AWS 您的密碼必須符合下列條件：

- 它必須至少有 8 個字元，最多 128 個字元。
- 它至少混用 3 種下列類型字元：大寫、小寫、數字和 ! @ # $ % ^ & * () <> [] {} | _+-= 符號。
- 它不得與您的 AWS 帳戶 姓名或電子郵件地址相同。

> **Note**
>
> AWS 正在推出登錄過程的改進。這些改進之一對您的帳戶強制執行更安全的密碼政策。如果 AWS 已升級您的帳戶，您必須符合前述的密碼政策。如果 AWS 尚未升級您的帳戶，則尚 AWS 未強制執行此政策。不過，我們強烈建議您遵循其準則，以便設定更安全的密碼。

6. 選擇儲存變更。

AWS CLI or AWS SDK

其中一個 AWS SDK 的 AWS CLI 或 API 作業不支援此工作。您只能使用來執行此工作 AWS Management Console。

重設遺失或忘記的根使用者密碼

當您第一次創建時 AWS 帳戶，您提供了一個電子郵件地址和密碼。這些是你的 AWS 帳戶根使用者 憑證 若您忘記根使用者密碼，可從 AWS Management Console重設密碼。

重設您的根使用者密碼：

1. 使用您的 AWS 帳戶 電子郵件地址開始以 root 使用AWS Management Console者身分登入，然後選擇 [下一步]。

> **Note**
>
> 如果您是使用 IAM 使用者憑證登入 AWS Management Console，則必須先登出再重設根使用者密碼。如果您看到帳戶專屬的 IAM 使用者登入頁面，請選擇頁面底部旁的 Sign-in using root account credentials (使用根帳戶憑證來登入)。如有需要，請提供您的帳戶電子郵件地址，然後選擇 Next (下一步)，以存取 Root user sign in (根使用者登入) 頁面。

2. 選擇 Forgot your password? (忘記您的密碼？)。

> **Note**
>
> 如果您是 IAM 使用者，則此選項無法使用。忘記您的密碼？選項僅適用於根使用者帳戶。IAM 使用者必須要求管理員重設忘記的密碼。如需詳細資訊，請參閱忘記 AWS 帳戶

的 IAM 使用者密碼。如果您透過登入 AWS 存取入口網站，請參閱重設您的 IAM 身分中心使用者密碼。

3. 提供與帳戶相關聯的電子郵件地址。然後提供 CAPTCHA 文字並選擇 Continue (繼續)。
4. 查看與您相關聯的電子郵件，以取 AWS 帳戶 得來自 Amazon Web Services 的訊息。電子郵件來自於結尾為 `@verify.signin.aws` 的地址。請遵循電子郵件中的指示進行。若您在帳戶中沒有看到電子郵件，請檢查您的垃圾郵件資料夾。如果您無法再存取該電子郵件，請參閱「AWS 登入使用者指南」中的「我無法存取我 AWS 帳戶的電子郵件」。

建立根使用者的存取金鑰

Warning

我們強烈建議您不要為您的根使用者建立存取金鑰對。由於只有少數工作需要 root 使用者，而且您通常不常執行這些工作，因此建議您登入 AWS Management Console 以執行 root 使用者工作。在建立存取金鑰前，請檢閱長期存取金鑰的替代方案。

雖然我們不建議這樣做，但您可以為 root 使用者建立存取金鑰，以便在 AWS Command Line Interface (AWS CLI) 中執行命令，或使用 root 使用者認證從其中一個 AWS SDK 使用 API 作業。當您建立存取金鑰時，可以將存取金鑰 ID 與私密存取金鑰建立為一組。在建立存取金鑰期間， AWS 讓您有機會檢視和下載存取金鑰的秘密存取金鑰部分。如果您沒有下載或者遺失私密存取金鑰，可以先刪除再建立新的存取金鑰。您可以使用主控台、 AWS CLI或 AWS API 建立根使用者存取金鑰。

新建立存取金鑰的狀態為「作用中」，這表示您可以使用存取金鑰進行 CLI 和 API 呼叫。您可以將最多兩個存取金鑰指派給根使用者。

未使用的存取金鑰應該停用。一旦存取金鑰處於非作用中，您將無法用它來進行 API 呼叫。非作用中金鑰仍會計入您的限制。您可以隨時建立或刪除存取金鑰。不過，刪除存取金鑰之後，即永久消失且無法擷取。

AWS Management Console

若要建立存取金鑰 AWS 帳戶根使用者

最低許可

若要執行下列步驟，您至少必須擁有下列 IAM 許可：

- 您必須以 AWS 帳戶 root 使用者身分登入，不需要額外的 AWS Identity and Access Management (IAM) 許可。您無法以 IAM 使用者或角色的身分執行這些步驟。

1. 使用您 AWS 帳戶的電子郵件地址和密碼登入入門 AWS 帳戶根使用者。 AWS Management Console
2. 在主控台的右上角，選擇您的帳戶名稱或號碼，然後選擇 安全憑證 。
3. 在 Access keys (存取金鑰) 區段中，選擇 Create access key (建立存取金鑰)。如果此選項不可用，表示您所擁有的存取金鑰已達最大數目。您必須先刪除其中一個現有的存取金鑰，才能建立新的金鑰。如需詳細資訊，請參閱 IAM 物件配額。
4. 在根使用者存取金鑰的替代方案頁面上，檢閱安全建議。若要繼續，請選取核取方塊，然後選擇建立存取金鑰。
5. 在擷取存取金鑰頁面上，會顯示您的存取金鑰 ID。
6. 在私密存取金鑰下方，選擇顯示，然後從您的瀏覽器視窗複製存取金鑰 ID 和私密金鑰，將其貼到安全的地方。或者，您可以選擇下載 .csv 檔案，這樣做會下載一個名為 `rootkey.csv` 的檔案，其中包含存取金鑰 ID 和私密金鑰。將檔案儲存到安全的位置。
7. 選擇完成。當您不再需要存取金鑰時，我們建議您將其刪除，或者至少考慮停用該金鑰，以防止其他人濫用。

AWS CLI & SDKs

若要為根使用者建立存取金鑰

> **Note**
>
> 若要以根使用者身分執行下列命令或 API 操作，您必須已有一個作用中的存取金鑰對。如果您沒有任何存取金鑰，可使用 AWS Management Console建立第一個存取金鑰。然後，您可以使用第一個存取金鑰的認證與 AWS CLI 建立第二個存取金鑰，或刪除存取金鑰。

- AWS CLI：AWS create-access-key

  Example

```
$ aws iam create-access-key
{
```

```
    "AccessKey": {
        "UserName": "MyUserName",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "Status": "Active",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "CreateDate": "2021-04-08T19:30:16+00:00"
    }
}
```

- AWS API：CreateAccessKey在 IAM API 參考資料中。

刪除根使用者的存取金鑰

您可以使用 AWS Management Console、 AWS CLI 或 AWS API 刪除根使用者存取金鑰。

AWS Management Console

若要刪除根使用者的存取金鑰

> **最低許可**
>
> 若要執行下列步驟，您至少必須擁有下列 IAM 許可：
>
> - 您必須以 AWS 帳戶 root 使用者身分登入，不需要額外的 AWS Identity and Access Management (IAM) 許可。您無法以 IAM 使用者或角色的身分執行這些步驟。

1. 使用您 AWS 帳戶的電子郵件地址和密碼登入入門 AWS 帳戶根使用者。 AWS Management Console
2. 在主控台的右上角，選擇您的帳戶名稱或號碼，然後選擇 安全憑證 。
3. 在存取金鑰區段中，選擇您想要刪除的存取金鑰，然後在動作下方選擇刪除。

   > **Note**
   >
   > 或者，您可以停用存取金鑰，而不是永久刪除它。這可讓您在未來繼續使用它，而無需變更金鑰 ID 或私密金鑰。當密鑰處於非活動狀態時，任何嘗試在對 AWS API 的請求中使用它都會失敗，並且錯誤訪問被拒絕。

4. 在刪除 <access key ID> 對話方塊中，選擇停用，輸入存取金鑰 ID 以確認您想要刪除它，然後選擇刪除。

AWS CLI & SDKs

若要刪除根使用者的存取金鑰

> **最低許可**
>
> 若要執行下列步驟，您至少必須擁有下列 IAM 許可：
>
> - 您必須以 AWS 帳戶 root 使用者身分登入，不需要額外的 AWS Identity and Access Management (IAM) 許可。您無法以 IAM 使用者或角色的身分執行這些步驟。

- AWS CLI：AWS delete-access-key

  Example

  ```
  $ aws iam delete-access-key \
      --access-key-id AKIAIOSFODNN7EXAMPLE
  ```

  此命令成功後就不會產生輸出。

- AWS API：DeleteAccessKey

需要根使用者憑證的任務

> **Important**
>
> 登入時遇到問題 AWS嗎？ 請確定您位在使用者類型的正確 AWS 登入頁面。如果您是 AWS 帳戶根使用者 (帳戶擁有者)，則可以 AWS 使用建立 AWS 帳戶. 如果您是 IAM 使用者，您的帳戶管理員可以為您提供可用來登入 AWS的憑證。如果您需要請求支持，請不要使用此頁面上的反饋鏈接，因為表單是由 AWS 文檔團隊收到的，而不是 AWS Support。請在 [聯絡我們] 頁面上選擇 [仍無法登入您的 AWS 帳戶]，然後選擇其中一個可用的支援選項。

建議您在中配置管理使用者， AWS IAM Identity Center以執行每日工作和存取 AWS 資源。不過，您可以只以帳戶根使用者身分登入來執行任務下面所列的任務。

帳戶管理工作

- 變更您的帳戶設定。這包括帳戶名稱、電子郵件地址、根使用者密碼和根使用者存取金鑰。其他帳戶設定，例如聯絡資訊、付款貨幣偏好設定 AWS 區域，以及不需要 root 使用者憑證。
- 還原 IAM 使用者許可。如果唯一的 IAM 管理員不小心撤銷自己的許可，您可以根使用者的身分登入，即可編輯政策和還原這些許可。
- 關閉您的 AWS 帳戶.

  如需詳細資訊，請參閱下列主題：

  - 如何將我的擁有權指派 AWS 帳戶 給另一個實體？ 。
  - 如何關閉我的 AWS 帳戶？ 。
  - 關閉一個獨立的 AWS 帳戶。

帳單工作

- 啟動對帳單與成本管理主控台的 IAM 存取。
- 某些帳單任務僅限於 root 用戶。如需詳細資訊，請參閱 AWS Billing 使用指南 AWS 帳戶中的〈管理〉。
- 檢視特定稅務發票。擁有 aws-Portal：ViewBilling權限的 IAM 使用者可以檢視和下載來自 AWS 歐洲的加值稅發票，但不能檢視和下載來自 AWS Inc. 或 Amazon Internet Services Private Limited (AISPL) 的加值稅發票。

AWS GovCloud (US) 任務

- 註冊 AWS GovCloud (US)。
- 請求 AWS GovCloud (US) 帳號根使用者存取金鑰 AWS Support。

Amazon EC2 任務

- 在預留執行個體市場註冊為賣方。

AWS KMS 任務

- 如果 AWS Key Management Service 金鑰變得無法管理，管理員可以透過連絡方式將其復原 AWS Support；但是，請透過確認票證 OTP AWS Support 回應您 root 使用者的主要電話號碼以進行授權。

Amazon Mechanical Turk 任務

- AWS 帳戶 將您的 mTurk 請求者帳戶鏈接。

Amazon 簡單存儲服務任務

- 設定 Amazon S3 儲存貯體，以啟用 MFA (多重因素認證)。
- 編輯或刪除拒絕所有主體的 Amazon S3 儲存貯體政策。

Amazon 簡單隊列服務任務

- 編輯或刪除拒絕所有主體的 Amazon SQS 資源政策。

使用根使用者來疑難排解問題

使用此處的資訊來幫助您疑難排解與 AWS 帳戶根使用者有關的問題。

在以帳戶根使用者身分登入時，我無法執行本該可以執行的任務

當您以帳戶根使用者身分登入時，如果您無法完成任務，則您的帳戶可能是 AWS Organizations中的組織成員。如果是這樣，而且您的組織管理員使用服務控制政策 (SCP) 來限制帳戶的許可，則所有使用者，包括根使用者，都會受到影響。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。

我忘記我 AWS 帳戶的根使用者密碼

如果您是 root 使用者，且遺失或忘記了您的密碼 AWS 帳戶，您可以重設密碼。您必須知道用來建立 AWS 帳戶的電子郵件地址，且您必須能夠存取該電子郵件帳戶。如需詳細資訊，請參閱 重設遺失或忘記的根使用者密碼。

我無法存取我的電子郵件 AWS 帳戶

當您建立時 AWS 帳戶，您會提供電子郵件地址和密碼。這些是 AWS 帳戶根使用者的登入資料。如果您不確定與您相關聯的電子郵件地址 AWS 帳戶，請搜尋來自@signin.aws或傳送@verify.signin.aws至您組織的任何電子郵件地址 (可能已用來開啟 AWS 帳戶.

如果您知道電子郵件地址，但無法存取該電子郵件，請先嘗試使用以下其中一個選項復原對電子郵件的存取：

- 如果您擁有電子郵件地址的網域，您可以還原刪除的電子郵件地址。或者，您可以為您的電子郵件帳戶設定全部截獲，這會「截獲」所有傳送到已不在郵件伺服器上電子郵件地址的訊息，並將這些訊息重新引導到另一個電子郵件地址。
- 如果帳戶上的電子郵件地址屬於您的公司電子郵件系統，我們建議您與 IT 系統管理員聯絡。這也許有助您重新取得電子郵件的存取許可。

如果您仍然無法登入 AWS 帳戶，您可以前往「聯絡我們」找到其他支援選項。

相關資訊

下列文章將提供有關如何使用根使用者的更多資訊。

- 保護我 AWS 帳戶 及其資源的一些最佳做法是什麼？
- 如何建立 EventBridge 事件規則以通知我已使用 root 使用者？
- 監控並通知 AWS 帳戶根使用者 活動
- 監控 IAM 根使用者活動

IAM 使用者

> **Important**
>
> IAM 最佳實務建議您要求人類使用者與身分識別提供者的聯合使用，以 AWS 使用臨時登入資料存取，而不是使用具有長期登入資料的 IAM 使用者。

AWS Identity and Access Management (IAM) 使用者是您在中建立的實體 AWS。IAM 使用者代表使用 IAM 使用者與之互動的人類使用者或工作負載 AWS。中的使用者 AWS 包含名稱和認證。

具有管理員許可的 IAM 使用者與 AWS 帳戶根使用者並不相同。如需有關根使用者的詳細資訊，請參閱 AWS 帳戶根使用者。

如何 AWS 識別 IAM 使用者

當建立 IAM 使用者時，IAM 建立這些方法來識別該使用者：

- IAM 使用者的「易用名稱」，即您在建立 IAM 使用者時指定的名稱，例如 `Richard` 或 `Anaya`。這些是您在 AWS Management Console中看到的名稱。

- IAM 使用者的 Amazon Resource Name (ARN)。當您需要唯一識別所有 IAM 使用者時，請使用 ARN。 AWS例如，在 Amazon S3 儲存貯體的 IAM 政策中，您可以使用 ARN 將 IAM 使用者指定為 `Principal`。IAM 使用者的 ARN 可能如下所示：

  `arn:aws:iam::account-ID-without-hyphens:user/Richard`
- IAM 使用者的唯一識別碼。只有當您使用 API、Windows 專用工具或建立 IAM 使用者時 PowerShell，才會傳回此 ID；您在主控台中看不 AWS CLI 到此識別碼。

如需有關這些識別碼的詳細資訊，請參閱 IAM 識別碼。

IAM 使用者和憑證

您可以根據 IAM 使用者登入資料，以不同的方式存 AWS 取：

- 主控台密碼：IAM 使用者可以輸入的密碼，以登入互動式工作階段 (如 AWS Management Console)。停用 IAM 使用者的密碼 (主控台存取權) 可防止他們 AWS Management Console 使用其登入認證登入。它不會變更其許可，也不會阻止他們使用擔任的角色來存取主控台。
- 存取金鑰：用來向 AWS發出程式化呼叫。但在為 IAM 使用者建立存取金鑰之前，您可以考慮其他更安全的替代方案。如需詳細資訊，請參閱 AWS 一般參考 中的長期存取金鑰的考量事項和替代方案。如果 IAM 使用者具有作用中的存取金鑰，他們會繼續運作並允許透過 Windows 工具 AWS CLI PowerShell、 AWS API 或 AWS Console Mobile Application 用程式進行存取。
- 搭配使用的安全殼層金鑰 CodeCommit：OpenSSH 格式的安全殼層公開金鑰，可用來進行驗證。CodeCommit
- 伺服器憑證：可用來驗證某些 AWS 服務的 SSL/TLS 憑證。建議您使用 AWS Certificate Manager (ACM) 來佈建、管理和部署伺服器憑證。只有當您必須在 ACM 不支援的區域中支援 HTTPS 連接時，才應使用 IAM。如需了解哪些區域支援 ACM，請參閱 AWS 一般參考 中的 AWS Certificate Manager 端點和配額。

您可以選擇最適合您 IAM 使用者的憑證。當您使用 AWS Management Console 來建立 IAM 使用者，您必須選擇至少包含一個主控台密碼或存取金鑰。根據預設，使用 AWS CLI 或 AWS API 建立的全新 IAM 使用者沒有任何類型的登入資料。您必須根據使用案例，建立 IAM 使用者的憑證類型。

您可以使用以下選項來管理密碼、存取金鑰和多重要素驗證 (MFA) 裝置：

- 管理 IAM 使用者的密碼。建立和變更允許存取 AWS Management Console的密碼。設定密碼政策，以強制執行最低密碼複雜性。允許使用者變更自己的密碼。
- 管理 IAM 使用者的存取金鑰。建立和更新存取金鑰，以便以程式設計方式存取您帳戶中的資源。

- 為 IAM 使用者啟用多重要素驗證 (MFA)。根據最佳實務，我們建議您要求帳戶中的所有 IAM 使用者進行多重要素驗證。使用 MFA 時，使用者需要提供兩種身分驗證形式：首先，提供憑證，那是屬於他們的使用者身分 (密碼或存取金鑰) 的一部分。此外，他們提供硬體裝置或智慧型手機或平板電腦上的應用程式所產生的臨時數字代碼。
- 尋找未使用的密碼和存取金鑰。任何擁有您帳戶或帳戶中 IAM 使用者的密碼或存取金鑰的人都可以存取您的 AWS 資源。安全最佳實務是在使用者不再需要密碼和存取金鑰時將其移除。
- 下載您帳戶的憑證報告。您可以產生並下載「憑證報告」，其中會列出帳戶中的所有 IAM 使用者，及其各種憑證的狀態，包括密碼、存取金鑰和 MFA 裝置。對於密碼和存取金鑰、憑證報告會顯示密碼或存取金鑰最近使用的狀況。

IAM 使用者和許可

在預設情況下，新的 IAM 使用者沒有許可採取任何行動。他們沒有執行任何 AWS 操作或訪問任何 AWS 資源的授權。擁有個別 IAM 使用者的好處是您可以為每個使用者個別指派許可。您可以將管理許可指派給幾個使用者，這些使用者可以管理您的 AWS 資源，甚至可以建立和管理其他 IAM 使用者。但是，在大多數情況下，您只想將使用者的權限限制為工作所需的任務 (AWS 動作或作業) 和資源。

試想一名叫做 Diego 的使用者。當您建立 IAM 使用者 `Diego` 時，您為該使用者建立一個密碼並連接許可，讓該使用者可以啟動特定 Amazon EC2 執行個體並從 Amazon RDS 資料庫的表格中讀取 (GET) 資訊。有關如何建立使用者並授予其初始憑證和許可的程序的更多資訊，請參閱 在您的中建立 IAM 使用者 AWS 帳戶。有關如何變更現有使用者的程序的詳細資訊，請參閱 變更 IAM 使用者的許可。有關如何變更使用者的密碼或存取金鑰的程序的詳細資訊，請參閱 管理使用者密碼 AWS 和 管理 IAM 使用者的存取金鑰。

您也可以新增許可界限到您的 IAM 使用者。許可界限是一項進階功能，可讓您使用 AWS 受管政策來限制以身分為基礎的政策可授予 IAM 使用者或角色的最大許可。如需有關政策類型及其使用的詳細資訊，請參閱 IAM 中的政策和許可。

IAM 使用者和帳戶

每位 IAM 使用者只能與一個 AWS 帳戶建立關聯。由於 IAM 使用者是在您的內部定義的 AWS 帳戶，因此他們不需要登記付款方式 AWS。IAM 使用者在您帳戶中執行的任何 AWS 活動都會從您的帳戶中收取費用。

AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

IAM 使用者做為服務帳戶

IAM 使用者是擁有關聯憑證和許可之 IAM 中的資源。IAM 使用者可以代表人員或應用程式，其會使用它的憑證來做出 AWS 請求。這通常稱為「服務帳戶」。如果您選擇在您的應用程式使用 IAM 使用者的長期憑證，請勿在應用程式程式碼中直接內嵌存取金鑰。 AWS SDK 和 AWS Command Line Interface 允許您將訪問密鑰放在已知位置，以便您不必將它們保留在代碼中。如需詳細資訊，請參閱 AWS 一般參考 中的適當管理 IAM 使用者存取金鑰。或者，最佳實務的作法是可以使用臨時安全憑證 (IAM 角色)，而不是長期存取金鑰。

在您的中建立 IAM 使用者 AWS 帳戶

Follow us on Twitter

Important

IAM 最佳實務建議您要求人類使用者與身分識別提供者的聯合使用，以 AWS 使用臨時登入資料存取，而不是使用具有長期登入資料的 IAM 使用者。

Note

如果您是因為正在尋找 Product Advertising API 相關資訊以在您的網站上銷售 Amazon 產品而找到此頁面，請參閱 Product Advertising API 5.0 文件。
如果您從 IAM 主控台到達此頁面，可能您的帳戶並不包含 IAM 使用者，即使您登入。您可以使用角色登入為 AWS 帳戶根使用者，或者使用暫時性登入資料登入。若要進一步了解 IAM 身分，請參閱IAM 身分 (使用者、使用者群組和角色)。

建立使用者並使該使用者執行工作的程序，包含以下步驟：

1. 在 AWS Management Console、、、適用於視窗的 AWS CLI工具中建立使用者 PowerShell，或使用 AWS API 作業。如果您在中建立使用者 AWS Management Console，則會根據您的選擇自動處理步驟 1—4。如果您透過程式設計的方式建立使用者，則必須個別執行每一個這些步驟。
2. 根據使用者需要的存取類型，建立使用者的憑證：
   - 啟用主控台存取 — 選用：如果使用者需要存取 AWS Management Console，請為使用者建立密碼。針對使用者停用主控台存取可以防止他們使用其使用者名稱和密碼登入 AWS Management Console 。它不會變更其許可，也不會阻止他們使用擔任的角色來存取主控台。

> **Tip**
>
> 僅建立使用者需要的憑證。例如，對於只需要透過存取權的使用者 AWS Management Console，請勿建立存取金鑰。

3. 透過將使用者新增到一個或多個群組，提供使用者執行所需任務的許可。您也可以將許可政策直接連接到使用者來授予許可。不過，我們建議您改為將您的使用者放入群組中，透過連接到這些群組的政策來管理許可。您也可以使用許可界限來限制使用者可以擁有的許可，但這並不常見。
4. (選用) 藉由連接標籤將中繼資料新增至使用者。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
5. 提供使用者必要的登入資訊。這包括使用者提供這些憑證所在的帳戶登入頁面的密碼和主控台 URL。如需詳細資訊，請參閱 IAM 使用者如何登入 AWS。
6. (選用) 設定使用者的多重要素驗證 (MFA)。MFA 要求使用者在每次登入 AWS Management Console. one-time-use
7. (選用) 提供使用者許可，以管理他們自己的安全憑證 (根據預設，使用者沒有管理他們自己的憑證的許可)。如需詳細資訊，請參閱 允許 IAM 使用者變更自己的密碼。

如需有關建立使用者所需的許可資訊，請參閱 存取 IAM 資源所需的許可。

主題

- 建立 IAM 使用者 (主控台)
- 建立 IAM 使用者 (AWS CLI)
- 建立身分與存取權管理AWS 使用者

建立 IAM 使用者 (主控台)

您可以使用建 AWS Management Console 立 IAM 使用者。

建立 IAM 使用者 (主控台)

1. 按照《AWS 登入使用者指南》 如何登入 AWS 主題中適合您使用者類型的登入程序操作。
2. 在 主控台首頁 頁面上選取 IAM 服務。
3. 在導覽窗格中選取 使用者，然後選取 新增使用者。
4. 在 指定使用者詳細資訊 頁面 使用者詳細資訊 下方的 使用者名稱 中輸入新使用者的名稱。這是新使用者的 AWS登入名稱。

> **Note**
>
> AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。使用者名稱可以是長達 64 個字母、數字以及這些字元的組合：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、@ 符號、底線 (_) 以及連字號 (-)。名稱在帳戶中必須是唯一的。它們無法透過大小寫進行區分。例如，您不可以建立兩個名為 TESTUSER 和 testuser 的使用者。使用者名稱用在政策中或作為 ARN 的一部分時，名稱區分大小寫。當主控台中的客戶顯示使用者名稱時 (例如在登入程序期間)，使用者名稱不區分大小寫。

5. 選取 [提供使用者存取權 — AWS Management Console 選用這會產生新使用者的 AWS Management Console 登入認證]。

   系統會詢問您是否要向使用者提供主控台存取。建議您在 IAM Identity Center (而非 IAM) 中建立使用者。

   - 若要切換至在 IAM Identity Center 建立使用者，請選擇 在 Identity Center 中指定使用者。

     如果您尚未啟用 IAM Identity Center，選擇此選項後，系統會將您導向主控台中的服務頁面，以便您啟用該項服務。如需此程序的詳細資訊，請參閱AWS IAM Identity Center 使用者指南中的 IAM 身分中心的一般工作入門

     如果您已啟用 IAM Identity Center，選擇此選項後，系統會將您導向 IAM Identity Center 中的 指定使用者詳細資訊 頁面。如需此程序的詳細資訊，請參閱使用指南中的「新增AWS IAM Identity Center使用者」

   - 如果您無法使用 IAM Identity Center，請選擇 我想要建立 IAM 使用者，然後繼續執行此程序。

   a. 在 主控台密碼 中選取下列其中一個選項：

      - 自動產生的密碼：使用者會取得符合 帳戶密碼政策 的隨機產生密碼。您可在進入 擷取密碼 頁面時檢視或下載密碼。
      - 自訂密碼：系統會將您於方塊內輸入的密碼指派給使用者。

   b. (選用) 系統在預設情況下會選取 使用者在下次登入時必須建立新的密碼 (建議)，以確保能夠強制使用者在第一次登入時變更其密碼。

> **Note**
>
> 如果管理員已啟用 允許使用者變更自己的密碼 帳戶密碼政策設定，則此核取方塊不會執行任何動作。否則，它會自動將名為的 `IAMUserChangePassword` AWS 受管政策連接到新使用者。政策會授予他們變更自己密碼的許可。

6. 選取 下一步。
7. 在 設定許可 頁面上指定為這位使用者指派許可的方式。選擇下列三個選項之一：
   - 新增使用者至群組：如果您想要將使用者指派至一或多個已經有許可政策的群組，請選擇此選項。IAM 會在您帳戶中顯示一系列群組及其連接的政策。您可以選取一或多個現有群組，或選擇 建立群組 來建立新的群組。如需詳細資訊，請參閱 變更 IAM 使用者的許可。
   - 複製許可：選擇此選項可將所有群組成員資格、連接的受管政策、內嵌的政策及任何現有的 許可界限 從現有使用者複製到新的使用者。IAM 會在您帳戶中顯示一份使用者清單。選擇許可最符合新使用者需求的選項。
   - 直接附加策略 — 選取此選項即可查看您帳戶中 AWS 受管理和客戶管理策略的清單。選取您想要連接至使用者的政策，或選擇 建立政策 來開啟新的瀏覽器標籤並建立新的政策。如需詳細資訊，請參閱程序 建立 IAM 政策 中的步驟 4。建立政策後，請關閉該標籤並返回您的原始標籤，以便將政策新增至使用者。

   > **Tip**
   >
   > 盡可能將您的政策連接到群組，然後讓使用者成為相應群組的成員。

8. (選用) 設定許可界限。這是進階功能。

   開啟 許可界限 區段，並選擇 使用許可界限來控制許可上限。IAM 會顯示您帳戶中受 AWS 管政策和客戶受管政策的清單。選取用於許可界限的政策，或選擇 建立政策 來開啟新的瀏覽器標籤並建立新的政策。如需詳細資訊，請參閱程序 建立 IAM 政策 中的步驟 4。在您建立政策後，關閉該標籤並返回您的原始標籤，以選取用於許可界限的政策。
9. 選取下一步。
10. (選用) 在 檢閱和建立 頁面的 標籤 下方選擇 新增標籤，透過將標籤做為鍵值對連接，來將中繼資料新增至使用者。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
11. 查看您目前為止所做的所有選擇。準備好繼續時，請選取 建立使用者。
12. 在 擷取密碼 頁面上取得指派給使用者的密碼：

- 選取密碼旁邊的 顯示 來檢視使用者的密碼，以便手動記錄密碼。
- 選取 下載 .csv 將使用者的登入憑證下載為 .csv 檔案，以便儲存到安全的位置。

13. 選取 電子郵件登入指示。您的本機郵件用戶端會開啟可供您自訂和傳送給使用者的草稿。電子郵件範本包含每位使用者的以下詳細資訊：

    - 使用者名稱
    - 帳戶登入頁面的 URL。使用以下範例，取代正確的帳戶 ID 號碼或帳戶別名：

    ```
    https://AWS-account-ID or alias.signin.aws.amazon.com/console
    ```

    > **Important**
    >
    > 使用者的密碼不會包含在產生的電子郵件中。您必須以遵循組織安全準則的方式向使用者提供密碼。

14. 如果使用者也需要存取金鑰來進行程式設計存取，請參閱管理 IAM 使用者的存取金鑰。

建立 IAM 使用者 (AWS CLI)

您可以使用 AWS CLI 建立 IAM 使用者。

建立 IAM 使用者 (AWS CLI)

1. 建立使用者。

    - aws iam create-user

2. (選用) 提供使用者 AWS Management Console存取權。這需要密碼。您還必須提供使用者您帳戶登入頁面的 URL。

    - AWS IAM create-login-profile

3. (選用) 提供使用者程式設計存取權。這需要存取金鑰。

    - AWS IAM create-access-key
    - 視窗工具 PowerShell:新 IAM AccessKey
    - IAM API：CreateAccessKey

> **Important**
>
> 這是您檢視或下載秘密存取金鑰的唯一機會，您必須先將此資訊提供給使用者，才能使用 AWS API。請將使用者的新存取金鑰 ID 和私密存取金鑰存放在安全處。在此步驟之後，您將無法再存取該私密金鑰。

4. 新增使用者到一個或多個群組。您指定的群組應具有授予適當許可給使用者的連接政策。

   - AWS IAM add-user-to-group

5. (選用) 連接政策到定義使用者許可的使用者。注意：建議您透過將使用者新增至群組並將政策連接至群組，來管理使用者許可，而不要將政策直接連接至使用者。

   - AWS IAM attach-user-policy

6. (選用) 藉由連接標籤將自訂屬性新增至使用者。如需詳細資訊，請參閱 管理 IAM 使用者 (AWS CLI 或 AWS API) 的標籤。

7. (選用) 提供使用者許可，來管理他們自己的安全憑證。如需詳細資訊，請參閱 AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

建立身分與存取權管理AWS 使用者

您可以使用 AWS API 建立 IAM 使用者。

若要從 (AWS API) 建立身分與存取權管理使用者

1. 建立使用者。

   - CreateUser

2. (選用) 提供使用者 AWS Management Console存取權。這需要密碼。您還必須提供使用者您帳戶登入頁面的 URL。

   - CreateLoginProfile

3. (選用) 提供使用者程式設計存取權。這需要存取金鑰。

   - CreateAccessKey

> **Important**
>
> 這是您檢視或下載秘密存取金鑰的唯一機會，您必須先將此資訊提供給使用者，才能使用 AWS API。請將使用者的新存取金鑰 ID 和私密存取金鑰存放在安全處。在此步驟之後，您將無法再存取該私密金鑰。

4. 新增使用者到一個或多個群組。您指定的群組應具有授予適當許可給使用者的連接政策。

   - AddUserToGroup

5. (選用) 連接政策到定義使用者許可的使用者。注意：建議您透過將使用者新增至群組並將政策連接至群組，來管理使用者許可，而不要將政策直接連接至使用者。

   - AttachUserPolicy

6. (選用) 藉由連接標籤將自訂屬性新增至使用者。如需詳細資訊，請參閱 管理 IAM 使用者 (AWS CLI 或 AWS API) 的標籤。
7. (選用) 提供使用者許可，來管理他們自己的安全憑證。如需更多詳細資訊，請參閱 AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

控制 IAM 使用者對 AWS Management Console的存取

具有透 AWS 帳戶 過登入您的權限的 IAM 使用者 AWS Management Console 可以存取您的 AWS 資源。下列清單顯示您可以透過授與 IAM 使用者存取您 AWS 帳戶 資源的方式 AWS Management Console。它還顯示 IAM 使用者如何透過 AWS 網站存取其他 AWS 帳戶功能。

> **Note**
>
> 使用 IAM 免費。

該 AWS Management Console

您為需要存取 AWS Management Console的每個 IAM 使用者建立密碼。使用者可透過已啟用 IAM 的 AWS 帳戶 登入頁面存取主控台。如需有關存取登入頁面的資訊，請參閱《AWS 登入 使用者指南》 中的 如何登入 AWS。如需有關建立密碼的資訊，請參閱管理使用者密碼 AWS。

您可以移除其密碼，以防止 IAM 使用者存取。 AWS Management Console 這樣可以防止他們 AWS Management Console 使用其登錄憑據登錄。它不會變更其許可，也不會阻止他們使用擔任

的角色來存取主控台。如果使用者具有使用中的存取金鑰，則他們會繼續運作並允許透過 Windows 工具 AWS CLI PowerShell、 AWS API 或 AWS Console Mobile Application 用程式進行存取。

您的 AWS 資源，例如 Amazon EC2 執行個體、Amazon S3 儲存貯體等

即使 IAM 使用者有密碼，還是需要許可才能存取您的 AWS 資源。在建立 IAM 使用者時，該使用者預設是沒有許可的。若要為您的 IAM 使用者提供所需的許可，請將政策連接到他們。如果您有多個 IAM 使用者使用相同的資源執行相同的任務，則可以將這些 IAM 使用者指派給一個群組。然後，將許可指派給該群組。如需有關建立 IAM 使用者與群組的詳細資訊，請參閱 IAM 身分 (使用者、使用者群組和角色)。如需使用政策來設定許可的詳細資訊，請參閱 AWS 資源存取管理。

AWS 討論區

任何人都可以讀取 AWS 開發論壇上的文章。想要在 AWS 討論區張貼問題或意見的使用者可以使用自己的使用者名稱。使用者第一次張貼文到 AWS 討論區時，系統會提示使用者輸入暱稱和電子郵件地址。只有該使用者可以在 AWS 討論區中使用該暱稱。

您的 AWS 帳戶 帳單和使用資訊

您可以授權使用者存取您的 AWS 帳戶 帳單和使用資訊。如需詳細資訊，請參閱 AWS Billing 使用者指南中的控制帳單資訊的存取。

您的 AWS 帳戶 設定檔資訊

使用者無法存取您的 AWS 帳戶 設定檔資訊。

您的 AWS 帳戶 安全憑證

使用者無法存取您的 AWS 帳戶 安全認證。

> **Note**
>
> 無論介面為何，IAM 政策控制存取權。例如，您可以提供使用者存取 AWS Management Console所需的密碼。該使用者 (或使用者所屬的任何群組) 適用的政策會控制使用者可以在 AWS Management Console中做哪些事。或者，您可以為用戶提供用於對其進行 API 調用的 AWS 訪問密鑰 AWS。這些政策會控制使用者可以透過使用這些存取金鑰進行身分驗證的文件庫或用戶端，呼叫哪些動作。

IAM 使用者如何登入 AWS

若要以 IAM 使用者 AWS Management Console 身分登入，除了使用者名稱和密碼之外，您還必須提供帳戶 ID 或帳戶別名。當您的管理員在主控台中建立 IAM 使用者時，他們應已將您的登入憑證傳送給您，包括您的使用者名稱，以及包含您的帳戶 ID 或帳戶別名的帳戶登入頁面 URL。

```
https://My_AWS_Account_ID.signin.aws.amazon.com/console/
```

> **秘訣**
>
> 要在網頁瀏覽器為您的帳戶登入頁面建立書籤，您應該在書籤項目手動輸入帳戶的登入 URL。請勿使用 web 瀏覽器的書籤功能，因為重新引導會模糊登入 URL。

您也可以在以下通用登入端點登入，並手動輸入您的帳戶 ID 或帳戶別名：

```
https://console.aws.amazon.com/
```

為了方便起見， AWS 登入頁面會使用瀏覽器 Cookie 來記住 IAM 使用者名稱和帳戶資訊。下次使用者前往中的任何頁面時 AWS Management Console，主控台會使用 Cookie 將使用者重新導向至帳戶登入頁面。

您只能存取管理員在附加至 IAM 使用者身分的政策中指定的 AWS 資源。若要在主控台中工作，您必須擁有執行主控台執行之動作的權限，例如列出和建立 AWS 資源。如需詳細資訊，請參閱 AWS 資源存取管理 及 以身分為基礎的 IAM 政策範例。

> **Note**
>
> 如果您的組織已有現有的身分系統，您可能會想建立單一登入 (SSO) 選項。SSO 可讓使用者存取您 AWS Management Console 的帳戶，而不需要他們擁有 IAM 使用者身分。SSO 也不需要使用者登入您組織的網站和 AWS 分開登入。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

登錄登錄詳細信息 CloudTrail

如果您啟 CloudTrail 用將登入事件記錄到記錄檔，您必須瞭解如何 CloudTrail選擇記錄事件的位置。

- 如果您的使用者直接登入到主控台，則會根據所選服務主控台是否支援區域，將它們重新引導到全域登入或區域登入端點。例如，主要主控台首頁支援區域，所以您若登入以下 URL：

  ```
  https://alias.signin.aws.amazon.com/console
  ```

  系統會將您重新導向至區域登入端點`https://us-east-2.signin.aws.amazon.com`，例如，在使用者區域的 CloudTrail 記錄檔中產生地區記錄項目：

  另一方面，Amazon S3 主控台不支援區域，所以您若登入以下 URL

  ```
  https://alias.signin.aws.amazon.com/console/s3
  ```

  AWS 將您重定向到全局登錄端點`https://signin.aws.amazon.com`，從而導致全局 CloudTrail 日誌條目。
- 您可以手動請求特定區域登入端點，只要使用如下所示的 URL 語法，登入啟用區域的主要主控台首頁：

  ```
  https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
  ```

  AWS 將您重新導向至`ap-southeast-1`地區登入端點，並導致地區 CloudTrail 記錄事件。

如需 CloudTrail 和 IAM 的詳細資訊，請參閱使用 CloudTrail.

如果使用者需要以程式設計存取的方式使用您的帳戶，則您可以為每位使用者建立存取金鑰對 (存取金鑰 ID 和私密存取金鑰)。但在為使用者建立存取金鑰之前，您可以考慮其他更安全的替代方案。如需詳細資訊，請參閱 AWS 一般參考 中的長期存取金鑰的考量事項和替代方案。

透過您的 IAM 登入頁面來使用 MFA 裝置

設定使用者的多重要素驗證 (MFA) 裝置必須使用自己的 MFA 裝置登入 AWS Management Console。使用者輸入其登入認證後，請 AWS 檢查使用者的帳戶，以查看該使用者是否需要 MFA。以下各主題提供使用者在需要 MFA 時完成登入方式的資訊。

主題

- 使用已啟用的多台 MFA 裝置登入
- 使用 FIDO 安全性金鑰登入
- 使用虛擬 MFA 裝置登入

- 使用硬體 TOTP 權杖登入

使用已啟用的多台 MFA 裝置登入

如果使用者以 AWS 帳戶 root 使用者或 IAM 使用者身分登入，且該帳戶啟用了多個 MFA 裝置，則只需使用一個 MFA 裝置即可登入。 AWS Management Console 使用者使用其密碼進行身分驗證以後，他們要選取希望使用哪種 MFA 裝置類型來完成身分驗證。然後，使用者會收到使用其所選裝置類型進行身分驗證的提示。

使用 FIDO 安全性金鑰登入

若使用者需要 MFA，則會出現第二個登入頁面。使用者需要點選 FIDO 安全性金鑰。

> **Note**
>
> Google Chrome 瀏覽器使用者不應在要求 Verify your identity with amazon.com (使用 amazon.com 驗證您身分) 的彈出畫面中選擇任何可用選項。您只需要點選安全性金鑰。

與其他 MFA 裝置不同，FIDO 安全性金鑰不會失去同步。管理員可以在其遺失或損壞時停用 FIDO 安全性金鑰。如需詳細資訊，請參閱 停用 MFA 裝置 (主控台)。

如需支援 WebAuthn 與 FIDO 相容裝置的瀏覽器的相關資訊，請參閱。 AWS 使用金鑰和安全金鑰的支援組態

使用虛擬 MFA 裝置登入

若使用者需要 MFA，則會出現第二個登入頁面。在 MFA code (MFA 代碼) 方塊中，使用者必須輸入 MFA 應用程式提供的數字代碼。

如果 MFA 代碼正確，則使用者可以存取 AWS Management Console。如果程式碼不正確，使用者可以重試其他代碼。

虛擬 MFA 裝置可能會失去同步。如果使用者在嘗試數次之 AWS Management Console 後無法登入，系統會提示使用者同步處理虛擬 MFA 裝置。使用者可以根據螢幕上的提示同步虛擬 MFA 裝置。如需如何代表中的使用者同步處理裝置的詳細資訊 AWS 帳戶，請參閱重新同步虛擬及硬體 MFA 裝置。

使用硬體 TOTP 權杖登入

若使用者需要 MFA，則會出現第二個登入頁面。在 MFA code (MFA 代碼) 方塊中，使用者必須輸入硬體 TOTP 權杖提供的數字代碼。

如果 MFA 代碼正確，則使用者可以存取 AWS Management Console。如果程式碼不正確，使用者可以重試其他代碼。

硬體 TOTP 權杖可能不同步。如果使用者在嘗試數次 AWS Management Console 之後無法登入，系統會提示使用者同步處理 MFA Token 裝置。使用者可以根據螢幕上的提示同步 MFA 權杖裝置。如需如何代表中的使用者同步處理裝置的詳細資訊 AWS 帳戶，請參閱重新同步虛擬及硬體 MFA 裝置。

管理 IAM 使用者

> **Note**
>
> 最佳作法是，建議您要求人類使用者與身分識別提供者使用同盟，才能 AWS 使用臨時登入資料進行存取。如果遵循最佳實務，則您不用管理 IAM 使用者和群組。相反地，您的使用者和群組會在外部進行管理， AWS 並且能夠以同盟身分存取 AWS 資源。聯合身分識別是來自企業使用者目錄、Web 身分識別提供者、 AWS Directory Service、Identity Center 目錄的使用者，或使用透過身分識別來源提供的認證存取 AWS 服務的任何使用者。聯合身分使用由其身分提供者定義的群組。如果您使用的是 AWS IAM Identity Center，請參閱使用指南中的在 IAM 身分中心中管理身分，以取得有關在 IAM 身分中心建立使用AWS IAM Identity Center 者和群組的資訊。

Amazon Web Services 提供在 AWS 帳戶中管理 IAM 使用者的多項工具。您可以列出帳戶或使用者群組中的 IAM 使用者，也可以列出使用者所屬的所有使用者群組。您可以重新命名或變更 IAM 使用者的路徑。如果您要改用聯合身分而非 IAM 使用者，您可以刪除 AWS 帳戶的 IAM 使用者，或停用該使用者。

如需有關新增、變更或移除 IAM 使用者之受管政策的詳細資訊，請參閱 變更 IAM 使用者的許可。如需管理 IAM 使用者之內嵌政策的資訊，請參閱新增和移除 IAM 身分許可、編輯 IAM 政策 和 刪除 IAM 政策。最佳實務的做法是，使用受管政策而不是內嵌政策。AWS 受管政策為很多常見使用案例授予許可。請記住， AWS 受管理的政策可能不會針對您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。因此，我們建議您定義使用案例的客戶管理政策，以便進一步減少許可。如需詳細資訊，請參閱 AWS 受管理政策。如需針對特定工作職能所設計之 AWS 受管理原則的詳細資訊，請參閱AWS 受管理的工作職能政策。

若要了解驗證 IAM 政策的資訊，請參閱 驗證 IAM 政策。

> **Tip**
>
> IAM Access Analyzer 可以分析您 IAM 角色使用的服務和動作，然後產生您可以使用的精細政策。測試產生的每個政策後，您可以將政策部署到生產環境。這可確保您僅授予所需的許可給工作負載。如需政策產生的詳細資訊，請參閱 IAM Access Analyzer 政策產生。

如需管理 IAM 使用者密碼的資訊，請參閱 管理 IAM 使用者密碼。

主題

- 檢視使用者存取
- 列出 IAM 使用者
- 重新命名 IAM 使用者
- 刪除 IAM 使用者
- 停用 IAM 使用者

檢視使用者存取

刪除使用者之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

列出 IAM 使用者

您可以列出您 AWS 帳戶 或特定 IAM 使用者群組中的 IAM 使用者，並列出使用者所在的所有使用者群組。有關列出使用者所需的許可資訊，請參閱 存取 IAM 資源所需的許可。

列出帳戶中所有使用者

- AWS Management Console：在導覽窗格中，選擇 Users (使用者)。主控台會顯示您中的使用者 AWS 帳戶。
- AWS CLI：aws iam list-users
- AWS API：ListUsers

若要列出特定使用者群組中的使用者

- AWS Management Console：在導覽窗格中，選擇 User groups (使用者群組)，選擇使用者群組名稱，然後選擇 Users (使用者) 索引標籤。
- AWS CLI：aws iam get-group
- AWS API：GetGroup

列出使用者所屬的所有使用者群組

- AWS Management Console：在導覽窗格中，選擇 Users (使用者)，選擇使用者名稱，然後選擇 Groups (群組) 標籤。
- AWS CLI：AWS list-groups-for-user
- AWS API：ListGroupsForUser

重新命名 IAM 使用者

若要變更使用者的名稱或路徑 AWS CLI，您必須使用 Windows PowerShell 工具或 AWS API。主控台中沒有用於重新命名使用者的選項。有關重新命名使用者所需的許可資訊，請參閱 存取 IAM 資源所需的許可。

當您更改使用者名稱或路徑時，會發生以下情況：

- 連接到使用者的所有政策繼續採用新使用名稱。
- 採用新使用者名稱的使用者保留在原來的使用者群組。
- 該使用者的唯一 ID 保持不變。如需有關唯一 ID 的詳細資訊，請參閱 唯一識別碼。
- 任何將該使用者視為「主體」(向該使用者授予存取權) 的資源或角色政策都會自動更新，以使用新名稱或路徑。例如，Amazon SQS 中的任何佇列類型政策或 Amazon S3 中的任何資源類型政策都會自動更新，以使用新名稱和路徑。

IAM 不會自動更新將該使用者視為資源以使用新使用者名稱或路徑的政策；您只能手動更新。例如，該使用者 `Richard` 連接了一個政策，而該政策讓使用者可管理其自己的安全憑證。如果管理員將 `Richard` 重新命名為 `Rich`，則管理員還需要更新該政策以將資源從：

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Richard
```

變更為此：

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Rich
```

如果管理員更改路徑，也會發生這種情況；管理員需要更新政策以反映該使用者使用新路徑。

重新命名使用者

- AWS CLI：aws iam update-user
- AWS API：UpdateUser

刪除 IAM 使用者

如果該使用者退出您的公司，您可以從該 AWS 帳戶 使用者刪除 IAM 使用者。如果使用者暫時離開，您可以停用使用者的存取權，而不是依照 停用 IAM 使用者 中所述從帳戶中刪除他們。

主題

- 刪除 IAM 使用者 (主控台)
- 刪除 IAM 使用者 (AWS CLI)

刪除 IAM 使用者 (主控台)

當您使用刪除 IAM AWS Management Console 使用者時，IAM 會自動為您刪除下列資訊：

- 使用者
- 任何使用者群組成員關係，即從該使用者所屬的任何 IAM 使用者群組中移除該使用者
- 與使用者有關的任何密碼
- 屬於使用者的存取金鑰
- 內嵌於使用者的所有政策 (透過使用者群組許可適用於使用者的政策不受影響)

> **Note**
>
> 當您刪除使用者時，IAM 會移除任何連接至使用者的受管政策，但不會刪除受管政策。

- 任何相關的 MFA 裝置

刪除 IAM 使用者 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)，然後選取您要刪除之使用者名稱旁的核取方塊。
3. 在頁面頂端，選擇 Delete (刪除)。
4. 確認對話方塊中，在文字輸入欄位中輸入使用者名稱以確認刪除使用者。選擇 Delete (刪除)。

刪除 IAM 使用者 (AWS CLI)

不同於 AWS Management Console，當您使用刪除使用者時 AWS CLI，您必須手動刪除附加至使用者的項目。此程序說明步驟。

從帳戶中刪除使用者 (AWS CLI)

1. 如果使用者有密碼，刪除該使用者的密碼。

   `aws iam delete-login-profile`
2. 如果使用者有存取金鑰，則將其刪除。

   `aws iam list-access-keys` (列出使用者的存取金鑰) 和 `aws iam delete-access-key`
3. 刪除使用者的簽署憑證。注意，當您刪除安全憑證時，將會永遠消失、無法還原。

   `aws iam list-signing-certificates` (列出使用者的簽署的憑證) 和 `aws iam delete-signing-certificate`
4. 如果使用者有 SSH 公有金鑰，則將其刪除。

   `aws iam list-ssh-public-keys` (列出使用者的 SSH 公有金鑰) 和 `aws iam delete-ssh-public-key`
5. 刪除使用者的 Git 憑證。

   `aws iam list-service-specific-credentials` (列出使用者的 Git 憑證) 和 `aws iam delete-service-specific-credential`
6. 如果使用者有多重要素驗證 (MFA) 裝置，請將其停用。

   `aws iam list-mfa-devices` (列出使用者的 MFA 裝置)、`aws iam deactivate-mfa-device` (停用裝置) 和 `aws iam delete-virtual-mfa-device` (永久刪除虛擬 MFA 裝置)
7. 刪除使用者的內嵌政策。

aws iam list-user-policies (列出使用者的內嵌政策) 和 aws iam delete-user-policy (刪除政策)

8. 分開連接到該使用者的任何受管政策。

   aws iam list-attached-user-policies (列出連接到使用者的受管政策) 和 aws iam detach-user-policy (分開政策)

9. 將該使用者從任何使用者群組中移除。

   aws iam list-groups-for-user (列出使用者所屬的使用者群組) 和 aws iam remove-user-from-group

10. 刪除使用者。

    aws iam delete-user

停用 IAM 使用者

當 IAM 使用者暫時離開公司時，您可能需要停用他們。您可以將其 IAM 使用者登入資料保留在適當的位置，並仍然封鎖其 AWS 存取權

若要停用使用者，請建立並連接政策以拒絕使用者存取 AWS。您可以稍後恢復使用者的存取權限。

以下是您可以連接到使用者以拒絕其存取的兩個拒絕政策範例。

下列政策不包含時間限制。您必須移除政策才能恢復使用者的存取權。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*"
      }
   ]
}
```

下列政策包括在 2024 年 12 月 24 日晚上 11:59 (UTC) 啟動政策的條件，並於 2025 年 2 月 28 日晚上 11:59 (UTC) 結束該政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
          "DateGreaterThan": {"aws:CurrentTime": "2024-12-24T23:59:59Z"},
          "DateLessThan": {"aws:CurrentTime": "2025-02-28T23:59:59Z"}
          }
       }
   ]
}
```

變更 IAM 使用者的許可

您可以變更 IAM 使用者的群組成員資格、複製現有使用者的許可、直接將政策附加到使用者，或設定許可界限，以變更其中 IAM 使用者的許可。 AWS 帳戶 許可界限控制使用者可以擁有的許可上限。權限界限是一 AWS 項進階功能。

如需有關修改使用者許可所需的許可資訊，請參閱 存取 IAM 資源所需的許可。

主題

- 檢視使用者存取
- 根據使用者的存取活動產生政策
- 新增許可到使用者 (主控台)
- 變更使用者的許可 (主控台)
- 從使用者移除許可政策 (主控台)
- 從使用者移除許可界限 (主控台)
- 添加和刪除用戶的權限（ AWS CLI 或 AWS API ）

檢視使用者存取

變更使用者的許可之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

根據使用者的存取活動產生政策

您有時可能會對 IAM 實體 (使用者或角色) 授予超出其要求的許可。為了協助您精簡所授予的許可，您可以根據實體的存取活動產生 IAM 政策。IAM Access Analyzer 會檢閱您的 AWS CloudTrail 記錄檔，並產生政策範本，其中包含實體在指定日期範圍內使用的許可。您可以使用範本建立具有精細許可的受管政策，然後將其連接至 IAM 實體。如此一來，您只會針對特定使用案例授與使用者或角色與 AWS 資源互動所需的權限。如需進一步了解，請參閱 根據存取活動產生政策。

新增許可到使用者 (主控台)

IAM 提供三種方法將許可政策新增到使用者：

- 新增使用者至群組 – 讓使用者成為群組的成員。群組的政策連接到使用者。
- 從現有的使用者複製許可 – 複製所有群組成員資格、連接的受管政策、內嵌政策，以及來源使用者的任何現有許可界限。
- 直接連接政策到使用者 – 將受管政策直接連接到使用者。為方便管理許可，請將您的政策連接到群組，然後讓使用者成為相應群組的成員。

> **Important**
>
> 如果使用者具有許可界限，則您無法新增比許可界限允許的還要多的許可至使用者。

透過將使用者新增到群組來新增許可

新增使用者至群組會立即影響使用者。

若要透過將使用者新增到群組來新增許可

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者 。
3. 在主控台的 群組 欄中，檢閱使用者的目前群組成員資格。如有必要，請透過完成以下步驟將欄位新增到使用者表格：

   1. 在最右側的表格上方，選擇設定符號 ()。

2. 在 管理欄 對話方塊中，選取 群組 欄。或者，您也可以清除不想在使用者表格中顯示的任何欄標題的核取方塊。
3. 選擇 關閉 返回使用者清單。

群組 欄會告訴您使用者所屬的群組。此欄位包括多達兩個群組的群組名稱。如果使用者是三個或以上群組的成員，前兩個群組會顯示 (以字母順序排列)，以及包含其他群組成員資格的數量。例如，如果使用者屬於群組 A、群組 B、群組 C 和群組 D，則欄位包含值 Group A, Group B + 2 more (群組 A、群組 B+2 更多)。若要查看使用者所屬的群組總數，您可以將 群組計數 欄新增至使用者資料表。

4. 選擇您想要為其修改許可的使用者名稱。
5. 選擇 許可 標籤，然後選擇 新增許可。選擇 將使用者新增至群組。
6. 勾選您要使用者加入的各個群組的核取方塊。如果成為群組的成員，清單會顯示每個群組的名稱和使用者所接收的政策。
7. (選用) 除了從現有群組選取之外，您還可以選擇 建立群組 來定義一個新的群組：

   a. 在新標籤中的 使用者群組名稱 中輸入新群組的名稱。

   > **Note**
   >
   > AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。群組名稱可以是長達 128 個字母、數字以及這些字元的組合：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、@ 符號以及連字號 (-)。名稱在帳戶中必須是唯一的。它們無法透過大小寫進行區分。例如，您無法建立兩個名為 TESTGROUP 和 testgroup 的群組。

   b. 選擇一個或多個您要連接到群組的受管政策的核取方塊。您也可以選擇 建立政策 來建立新的受管政策。如果您這麼做時，在新政策完成時會返回此瀏覽器標籤或視窗；選擇 重新整理，然後選擇新的政策，以將它連接至您的群組。如需詳細資訊，請參閱 建立 IAM 政策。
   c. 選擇 建立使用者群組。
   d. 返回原始標籤，重新整理您的群組清單。然後，選取新群組的核取方塊。
8. 選擇 下一步，以查看要新增至使用者的群組成員資格清單。然後選擇 新增許可。

透過從其他使用者複製來新增許可

複製許可會立即影響使用者。

若要透過從其他使用者複製許可來新增許可至使用者

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者，並選擇您要修改其許可的使用者名稱，然後選擇 許可 標籤。
3. 選擇 新增許可，然後選擇 複製現有使用者的許可。清單會顯示可用的使用者及其群組成員資格和連接的政策。如果完整的群組或政策清單無法容納在一行中，您可以選擇 and ***n*** more (以及更多) 的連結。這麼做會開啟新的瀏覽器索引標籤，並看到完整的政策清單 (許可索引標籤) 和群組 (群組索引標籤)。
4. 選擇您要複製許可的使用者旁的選項按鈕。
5. 選擇 下一步，以查看要對使用者進行的變更清單。然後選擇 新增許可。

透過直接連接政策到使用者來新增許可

連接政策會立即影響使用者。

若要透過直接連接受管政策來新增許可到使用者

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者，並選擇您要修改其許可的使用者名稱，然後選擇 許可 標籤。
3. 選擇 新增許可，然後選擇 直接連接政策。
4. 選擇一個或多個您要連接到使用者的受管政策的核取方塊。您也可以選擇 建立政策 來建立新的受管政策。如此，您會在新政策建立完成時返回此瀏覽器標籤或視窗。選擇 重新整理，然後選取要連接至使用者之新政策的核取方塊。如需詳細資訊，請參閱 建立 IAM 政策。
5. 選擇 下一步，以查看要連接至使用者的政策清單。然後選擇 新增許可。

設定使用者的許可界限

設定許可界限會立即影響使用者。

設定使用者的許可界限

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者 。

3. 選擇您想要為其變更許可界限的使用者名稱。
4. 選擇 許可 標籤。如有必要，請開啟 許可界限 區段，然後選擇 設定許可界限。
5. 選擇要用於許可界限的政策。
6. 選擇 設定界限。

變更使用者的許可 (主控台)

IAM 可讓您以下列方式變更與使用者建立關聯的許可：

- 編輯許可政策 – 編輯使用者的內嵌政策、使用者群組的內嵌政策，或編輯直接連接到使用者或從群組連接的受管政策。如果使用者具有許可界限，則您無法提供比當成使用者的許可界限來人用的政策所允許的還要更多的許可。
- 變更許可界限 – 變更用作使用者的許可界限的政策。這會擴展或限制使用者可以擁有的許可上限。

編輯連接到使用者的許可政策

變更許可會立即影響使用者。

編輯使用者連接的受管政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇您想要為其變更許可政策的使用者名稱。
4. 選擇 Permissions (許可) 標籤。如有必要，請開啟 許可政策 區段。
5. 選擇您要編輯以檢視詳細資訊的政策名稱。選擇 政策使用 標籤，以檢視可能會受到政策編輯影響的其他實體。
6. 選擇 許可 標籤，然後檢閱政策授予的許可。然後選擇 編輯政策。
7. 編輯該政策並解決任何政策驗證建議。如需詳細資訊，請參閱 編輯 IAM 政策。
8. 選擇 檢閱政策 ，並檢閱政策摘要，然後選擇 儲存變更 。

變更使用者的許可界限

變更許可界限會立即影響使用者。

變更用於設定使用者許可界限的政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 使用者 。
3. 選擇您想要為其變更許可界限的使用者名稱。
4. 選擇 Permissions (許可) 標籤。如有必要，開啟 許可界限 區段，然後選擇 變更界限。
5. 選擇要用於許可界限的政策。
6. 選擇 設定界限。

從使用者移除許可政策 (主控台)

移除政策會立即影響使用者。

若要移除 IAM 使用者的許可

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇您想要移除其許可界限的使用者名稱。
4. 選擇 Permissions (許可) 標籤。
5. 如果您想要透過移除現有政策來移除許可，請檢視 類型 了解使用者如何取得該政策，再選擇 移除 來移除該政策：

   - 如果因群組成員資格而套用政策，則選擇 移除 會從群組移除使用者。請記住，您可能會有多個政策連接到單一群組。如果您移除群組中的使用者，使用者會失去透過群組成員資格收到的「所有」政策的存取權。
   - 如果政策是直接連接到使用者的受管政策，則選擇 移除 會將政策與使用者分開。這不會影響政策本身或政策可能連接的任何其他目標實體。
   - 如果政策是內嵌政策，則選擇 X 會從 IAM 移除政策。直接連接到使用者的內嵌政策只存在於該使用者。

從使用者移除許可界限 (主控台)

移除許可界限會立即影響使用者。

從使用者移除許可界限

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇您想要移除其許可界限的使用者名稱。
4. 選擇 Permissions (許可) 標籤。如有必要，請開啟 許可界限 區段，然後選擇 移除界限。
5. 選擇 移除範圍 以確認您想要移除許可界限。

添加和刪除用戶的權限（AWS CLI 或 AWS API）

若要以程式設計方式新增或移除許可，您必須新增或移除群組成員資格、連接或分開受管政策，或者新增或刪除內嵌政策。如需詳細資訊，請參閱下列主題：

- 在 IAM 使用者群組中新增和移除使用者
- 新增和移除 IAM 身分許可

管理使用者密碼 AWS

您可以您帳戶中的 IAM 使用者管理密碼。IAM 使用者需要密碼才能存取 AWS Management Console. 使用者不需要密碼即可使用 Windows PowerShell 工具 AWS CLI、 AWS SDK 或 API 以程式設計方式存取 AWS 資源。對於這些環境，您可以選擇指派 IAM 使用者存取金鑰。不過，建議您先考慮其他更安全的方案來替代存取金鑰。如需詳細資訊，請參閱 AWS 安全認證。

目錄

設定 IAM 使用者的帳戶密碼政策

您可以在您的上設定自訂密碼政策， AWS 帳戶 以指定 IAM 使用者密碼的複雜性要求和強制輪替期間。如果您未設定自訂密碼政策，IAM 使用者密碼必須符合預設 AWS 密碼政策。如需詳細資訊，請參閱 自訂密碼政策選項。

主題

設定密碼政策的規則

IAM 密碼政策不適用於 AWS 帳戶根使用者 密碼或 IAM 使用者存取金鑰。如果密碼過期，IAM 使用者將無法登入， AWS Management Console 但可以繼續使用其存取金鑰。

當您建立或變更密碼政策時，大多數密碼政策設定將在使用者下次變更密碼時強制執行。但是，某些設定會立即強制執行。例如：

- 當最短長度和字元類型要求變更時，將在使用者下次變更密碼時強制執行這些設定。即使現有的密碼不遵守更新的密碼政策，也不會強制使用者變更現有的密碼。
- 當您設定密碼過期期間時，會立即強制執行過期期間。例如，假設您將密碼過期期間設定為 90 天。在這種情況下，現有密碼超過 90 天的所有 IAM 使用者的密碼會到期。這些使用者必須在下次登入時變更其密碼。

在指定次數的登入嘗試失敗之後，您無法建立「鎖定政策」來鎖定使用者帳戶。為了增強安全性，我們建議您將強式密碼政策與多重要素驗證 (MFA) 結合。如需有關 MFA 的詳細資訊，請參閱 在中使用多因素身份驗證（ MFA ） AWS。

設定密碼政策所需的許可

您必須設定許可，以允許 IAM 實體 (使用者或角色) 檢視或編輯其帳戶密碼政策。您可以在 IAM 政策中包含下列密碼政策動作：

- `iam:GetAccountPasswordPolicy` – 允許實體檢視其帳戶的密碼政策
- `iam:DeleteAccountPasswordPolicy` – 允許實體刪除其帳戶的自訂密碼政策，並還原為預設密碼政策

- `iam:UpdateAccountPasswordPolicy` – 允許實體建立或變更其帳戶的自訂密碼政策

下列政策允許完整存取權以檢視和編輯帳戶密碼政策。若要了解如何使用此範例 JSON 政策文件來建立 IAM 政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessPasswordPolicy",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:DeleteAccountPasswordPolicy",
                "iam:UpdateAccountPasswordPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

如需 IAM 使用者變更其自己密碼所需許可的資訊，請參閱 允許 IAM 使用者變更自己的密碼。

預設密碼政策

如果管理員未設定自訂密碼政策，IAM 使用者密碼必須符合預設 AWS 密碼政策。

預設密碼政策會強制執行下列條件：

- 密碼長度最短為 8 個字元，最長為 128 個字元。
- 至少混用 3 種下列類型字元：大寫、小寫、數字和非英數字元 (! @ # $ % ^ & * ( ) _ + - = [ ] { } | ')
- 與您的 AWS 帳戶 姓名或電子郵件地址不相同
- 保證密碼不會過期

自訂密碼政策選項

當您設定帳戶的自訂密碼政策時，您可以指定下列條件：

- 密碼最小長度 – 您可以指定至少 6 個字元，最多可指定 128 個字元。

- 密碼強度 – 您可以選取下列任一核取方塊來定義 IAM 使用者密碼的強度：
  - 至少需要一個拉丁字母 (A–Z) 的大寫字母
  - 至少需要一個拉丁字母 (a–z) 的小寫字母
  - 至少需要有一個數字
  - 至少需要一個非英數字元的字元 ! @ # $ % ^ & * ( ) _ + - = [ ] { } | '
- Turn on password expiration (啟用密碼過期) – 您可以選取並指定至少 1 天，且最長為 1,095 天的 IAM 使用者密碼有效期，設定後生效。例如，如果您指定 90 天的到期日，它會立即影響所有使用者。對於密碼超過 90 天的使用者，當他們在變更後登入主控台時，必須設定新密碼。擁有密碼 75-89 天前的使用者會收到有關其密碼到期的 AWS Management Console 警告。IAM 使用者可隨時變更密碼 (如果他們有許可)。當他們設定新的密碼時，該密碼的到期期間將重新開始。IAM 使用者一次只能有一個有效的密碼。
- 密碼過期需要管理員重設 — 選取此選項可防止 IAM 使用者在密碼 AWS Management Console 到期後使用更新自己的密碼。在選取此選項之前，請確認您的 AWS 帳戶 有多個具有管理許可的使用者，以重設 IAM 使用者密碼。具有 `iam:UpdateLoginProfile` 許可的管理員可以重設 IAM 使用者密碼。具有 `iam:ChangePassword` 許可和作用中存取金鑰的 IAM 使用者可以程式設計方式重設自己的 IAM 使用者主控台密碼。如果您清除此核取方塊，密碼到期的 IAM 使用者仍必須先設定新密碼，才能存取 AWS Management Console。
- Allow users to change their own password (允許使用者變更自己的密碼) – 您可以允許帳戶中的所有 IAM 使用者變更自己的密碼。這可讓使用者僅針對其使用者存取 `iam:ChangePassword` 動作，並存取 `iam:GetAccountPasswordPolicy` 動作。此選項不會將許可政策連接到每個使用者。反之，IAM 會在帳戶層級套用許可到所有使用者。或者，您只允許一些使用者管理自己的密碼。若要執行這項操作，請清除此核取方塊。如需有關使用政策以限制誰可以管理密碼的詳細資訊，請參閱 允許 IAM 使用者變更自己的密碼。
- 防止密碼重複使用 – 您可以防止 IAM 使用者重複使用舊密碼的指定數字。您可以指定長度下限為 1，以及長度上限為 24 的舊密碼，無法重複使用。

設定密碼政策 (主控台)

您可以使用 AWS Management Console 建立、變更或刪除自訂密碼原則。

建立自訂密碼政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇帳戶設定。

3. 在 Password policy (密碼政策) 區段中，選擇 Edit (編輯)。
4. 選擇 Custom (自訂) 以使用自訂密碼政策。
5. 選取要套用至密碼政策的選項，然後選擇 Save changes (儲存變更)。
6. 選擇 Set custom (設定自訂)，確認您要設定自訂密碼政策。

變更自訂密碼政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇帳戶設定。
3. 在 Password policy (密碼政策) 區段中，選擇 Edit (編輯)。
4. 選取要套用至密碼政策的選項，然後選擇 Save changes (儲存變更)。
5. 選擇 Set custom (設定自訂)，確認您要設定自訂密碼政策。

要刪除自訂密碼政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇帳戶設定。
3. 在 Password policy (密碼政策) 區段中，選擇 Edit (編輯)。
4. 選擇 IAM default (IAM 預設值) 以刪除自訂密碼政策，然後選擇 Save changes (儲存變更)。
5. 選擇 Set default (設定預設值)，確認您要設定 IAM 預設密碼政策。

設定密碼政策 (AWS CLI)

您可以使用 AWS Command Line Interface 來設定密碼原則。

若要管理自訂帳號密碼策略 AWS CLI

執行下列命令：

- 若要建立或變更自訂密碼政策：`aws iam update-account-password-policy`
- 若要檢視密碼政策：`aws iam get-account-password-policy`
- 若要刪除自訂密碼政策：`aws iam delete-account-password-policy`

設定密碼原則 (AWS API)

您可以使用 AWS API 操作來設置密碼策略。

從 AWS API 管理自訂帳戶密碼政策

呼叫以下操作：

- 若要建立或變更自訂密碼政策：UpdateAccountPasswordPolicy
- 若要檢視密碼政策：GetAccountPasswordPolicy
- 若要刪除自訂密碼政策：DeleteAccountPasswordPolicy

管理 IAM 使用者密碼

使用來處理 AWS 資源 AWS Management Console 的 IAM 使用者必須擁有密碼才能登入。您可以建立、更改或刪除您的 AWS 帳戶中 IAM 使用者的密碼。

將密碼指派給使用者之後，使用者可以 AWS Management Console 使用您帳戶的登入 URL 登入，如下所示：

```
https://12-digit-AWS-account-ID or alias.signin.aws.amazon.com/console
```

如需 IAM 使用者如何登入的詳細資訊 AWS Management Console，請參閱使用AWS 登入 者指南 AWS中的如何登入。

即使使用者有自己的密碼，還是需要許可才能存取您的 AWS 資源。使用者預設沒有任何許可。為授予使用者所需的許可，您可向使用者或使用者所屬的群組分配政策。如需有關建立使用者與群組的詳細資訊，請參閱 IAM 身分 (使用者、使用者群組和角色)。如需使用政策來設定許可的詳細資訊，請參閱 變更 IAM 使用者的許可。

您可以授予使用者變更自己密碼的許可。如需詳細資訊，請參閱 允許 IAM 使用者變更自己的密碼。如需有關使用者如何存取您帳戶登入頁面的資訊，請參閱 《AWS 登入 使用者指南》 中的 如何登入 AWS。

主題

- 建立、變更或刪除 IAM 使用者密碼 (主控台)
- 建立、變更或刪除 IAM 使用者密碼 (AWS CLI)
- 建立、變更或刪除 IAM 使用者密碼 (AWS API)

建立、變更或刪除 IAM 使用者密碼 (主控台)

您可以使用 AWS Management Console 來管理 IAM 使用者的密碼。

當使用者離開您的組織或不再需要 AWS 存取權時，請務必尋找他們正在使用的認證，並確保他們不再運作。在理想情況下，如果不再需要可刪除憑證。有需要時，您隨時可以在日後重新建立它們。至少也應變更憑證，讓前任持有者無法再繼續存取。

為 IAM 使用者新增密碼 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇您想要為之建立密碼的使用者名稱。
4. 選擇 安全憑證 標籤，然後在 主控台登入 下方選擇 啟用主控台存取。
5. 在 [啟用主控台存取權] 中，對於 [主控台密碼]，選擇要讓 IAM 產生密碼還是建立自訂密碼：
   - 若要讓 IAM 產生密碼，請選擇 Autogenerated password (自動產生的密碼)。
   - 若要建立自訂密碼，請選擇 Custom password (自訂密碼) 並輸入密碼。

   > **Note**
   > 您建立的密碼必須符合帳戶的密碼政策。

6. 若要要求使用者登入時建立新的密碼，請選擇 使用者必須在下次登入時建立新的密碼。然後選擇「啟用控制台訪問」

   > **Important**
   > 若您選取 使用者必須在下次登入時建立新的密碼 選項，請確認使用者擁有更改密碼的許可。如需詳細資訊，請參閱 允許 IAM 使用者變更自己的密碼。

7. 若要檢視密碼以便與使用者共用，請在 [主控台密碼] 對話方塊中選擇 [顯示]。

   > **Important**
   > 基於安全原因，在完成此步驟後您無法存取該密碼，但您可以隨時建立新密碼。

為 IAM 使用者變更密碼 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇您想要為之變更密碼的使用者名稱。
4. 選擇 安全憑證 標籤，然後在 主控台登入 下方選擇 管理主控台存取。
5. 在 [管理主控台存取] 中，選擇 [重設密碼 (如果尚未選取) 如果停用主控台存取權，則不需要密碼。
6. 對於主控台存取，請選擇要讓 IAM 產生密碼還是建立自訂密碼：
   - 若要讓 IAM 產生密碼，請選擇 Autogenerated password (自動產生的密碼)。
   - 若要建立自訂密碼，請選擇 Custom password (自訂密碼) 並輸入密碼。

   > **Note**
   >
   > 如果目前已設定密碼，您建立的密碼必須符合帳戶的密碼政策。

7. 若要要求使用者登入時建立新的密碼，請選擇 使用者必須在下次登入時建立新的密碼。

   > **Important**
   >
   > 若您選取 使用者必須在下次登入時建立新的密碼 選項，請確認使用者擁有更改密碼的許可。如需詳細資訊，請參閱 允許 IAM 使用者變更自己的密碼。

8. 若要撤銷使用者的作用中主控台工作階段，請選擇「撤銷作用中主控台 接著選擇 Apply (套用)。

   當您撤銷使用者的作用中主控台工作階段時，IAM 會將新的內嵌政策附加至使用者，拒絕所有動作的所有權限。它包含一個條件，只有在您撤銷權限的時間點之前建立工作階段，以及 future 大約 30 秒時，才會套用限制。如果使用者在您撤銷權限後建立新的工作階段，則拒絕原則不會套用至該使用者。如果使用者使用此方法撤銷自己的作用中主控台工作階段，則會立即從. AWS Management Console

   > **Important**
   >
   > 若要成功撤銷使用者的作用中主控台工作階段，您必須擁有該使用者的`PutUserPolicy`權限。這可讓您將`AWSRevokeOlderSessions`內嵌原則附加至使用者。

9. 若要檢視密碼以便與使用者共用，請在 [主控台密碼] 對話方塊中選擇 [顯示]。

> **Important**
>
> 基於安全原因，在完成此步驟後您無法存取該密碼，但您可以隨時建立新密碼。

刪除 (停用) IAM 使用者密碼 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇您想要為之刪除密碼的使用者名稱。
4. 選擇 安全憑證 標籤，然後在 主控台登入 下方選擇 管理主控台存取。
5. 在 [管理主控台存取] 中，選擇 [停用主控台存取 (如果尚未選取 如果停用主控台存取權，則不需要密碼。
6. 若要撤銷使用者的作用中主控台工作階段，請選擇「撤銷作用中主控台 然後選擇禁用訪問。

> **Important**
>
> 若要成功撤銷使用者的作用中主控台工作階段，您必須擁有該使用者的`PutUserPolicy`權限。這可讓您將`AWSRevokeOlderSessions`內嵌原則附加至使用者。

當您撤銷使用者的作用中主控台工作階段時，IAM 會在 IAM 使用者中內嵌新的內嵌政策，拒絕所有動作的所有許可。它包含一個條件，只有在您撤銷權限的時間點之前建立工作階段，以及 future 大約 30 秒時，才會套用限制。如果使用者在您撤銷權限後建立新的工作階段，則拒絕原則不會套用至該使用者。如果使用者使用此方法撤銷自己的作用中主控台工作階段，則會立即從. AWS Management Console

> **Important**
>
> 您可以移除其密碼，以防止 IAM 使用者存取。 AWS Management Console 這樣可以防止他們 AWS Management Console 使用其登錄憑據登錄。它不會變更其許可，也不會阻止他們使用擔任的角色來存取主控台。如果使用者具有使用中的存取金鑰，則他們會繼續運作並允許透過

Windows 工具 AWS CLI PowerShell、 AWS API 或 AWS Console Mobile Application 用程式進行存取。

建立、變更或刪除 IAM 使用者密碼 (AWS CLI)

您可以使用 AWS CLI API 管理 IAM 使用者的密碼。

建立密碼 (AWS CLI)

1. （可選）要確定用戶是否有密碼，請運行以下命令：aws iam get-login-profile
2. 要創建密碼，請運行以下命令：aws iam create-login-profile

若要變更使用者的密碼 (AWS CLI)

1. （可選）要確定用戶是否有密碼，請運行以下命令：aws iam get-login-profile
2. 要更改密碼，請運行以下命令：aws iam update-login-profile

若要刪除 (停用) 使用者的密碼 (AWS CLI)

1. （可選）要確定用戶是否有密碼，請運行以下命令：aws iam get-login-profile
2. (選用) 若要判斷密碼前次使用的時間，請執行此命令：aws iam get-user
3. 要刪除密碼，請運行以下命令：aws iam delete-login-profile

> **Important**
>
> 當您刪除使用者密碼時，使用者將無法繼續登入 AWS Management Console。如果使用者具有使用中的存取金鑰，他們會繼續運作並允許透過 Windows PowerShell 工具或 AWS API 函數呼叫進行存取。 AWS CLI當您使用 AWS CLI PowerShell、Windows 工具或 AWS API 刪除使用者時 AWS 帳戶，您必須先使用此作業刪除密碼。如需詳細資訊，請參閱 刪除 IAM 使用者 (AWS CLI)。

在指定時間之前撤銷使用者的作用中主控台工作階段 (AWS CLI)

1. 若要嵌入在指定時間之前撤銷 IAM 使用者作用中主控台工作階段的內嵌政策，請使用下列內嵌政策並執行此命令：aws iam put-user-policy

此內嵌政策會拒絕所有權限，並包含AWS#TokenIssue##條件索引鍵。它會在內嵌政策Condition元素中的指定時間之前撤銷使用者的作用中主控台工作階段。將aws:TokenIssueTime條件索引鍵值取代為您自己的值。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {
        "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
      }
    }
  }
}
```

2. （可選）若要列出 IAM 使用者中內嵌的內嵌政策名稱，請執行以下命令：aws iam list-user-policies
3. （可選）若要檢視 IAM 使用者中內嵌的具名內嵌政策，請執行以下命令：aws iam get-user-policy

建立、變更或刪除 IAM 使用者密碼 (AWS API)

您可以使用 AWS API 管理 IAM 使用者的密碼。

若要建立密碼 (AWS API)

1. (選擇性) 若要判斷使用者是否有密碼，請呼叫此作業：GetLoginProfile
2. 要創建密碼，請調用以下操作：CreateLoginProfile

若要變更使用者的密碼 (AWS API)

1. (選擇性) 若要判斷使用者是否有密碼，請呼叫此作業：GetLoginProfile
2. 要更改密碼，請調用以下操作：UpdateLoginProfile

若要刪除 (停用) 使用者的密碼 (AWS API)

1. (選擇性) 若要判斷使用者是否有密碼，請執行下列命令：GetLoginProfile
2. (選擇性) 若要判斷上次使用密碼的時間，請執行下列命令：GetUser
3. 若要刪除密碼，請執行以下指令：DeleteLoginProfile

> **Important**
>
> 當您刪除使用者密碼時，使用者將無法繼續登入 AWS Management Console。如果使用者具有使用中的存取金鑰，他們會繼續運作並允許透過 Windows PowerShell 工具或 AWS API 函數呼叫進行存取。 AWS CLI當您使用 AWS CLI PowerShell、Windows 工具或 AWS API 刪除使用者時 AWS 帳戶，您必須先使用此作業刪除密碼。如需詳細資訊，請參閱 刪除 IAM 使用者 (AWS CLI)。

在指定時間之前撤銷使用者的作用中主控台工作階段 (AWS API)

1. 若要嵌入在指定時間之前撤銷 IAM 使用者作用中主控台工作階段的內嵌政策，請使用下列內嵌政策並執行此命令：PutUserPolicy

   此內嵌政策會拒絕所有權限，並包含`AWS#TokenIssue##`條件索引鍵。它會在內嵌政策`Condition`元素中的指定時間之前撤銷使用者的作用中主控台工作階段。將`aws:TokenIssueTime`條件索引鍵值取代為您自己的值。

   ```
   {
     "Version": "2012-10-17",
     "Statement": {
       "Effect": "Deny",
       "Action": "*",
       "Resource": "*",
       "Condition": {
         "DateLessThan": {
           "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
         }
       }
     }
   }
   ```

2. (選擇性) 若要列出 IAM 使用者中內嵌的內嵌政策名稱，請執行以下命令：ListUserPolicies

3. (選擇性) 若要檢視 IAM 使用者中內嵌的具名內嵌政策，請執行以下命令：GetUserPolicy

允許 IAM 使用者變更自己的密碼

> **Note**
>
> 具有聯合身分的使用者將使用其身分提供者定義的程序來變更他們的密碼。最佳作法是要求人類使用者與身分識別提供者使用同盟，才能 AWS 使用臨時登入資料進行存取。

您可以授予 IAM 使用者許可，讓他們能夠登入 AWS Management Console變更自己的密碼。您可以使用兩種方式的其中一種來執行此動作：

- 允許帳戶中的所有 IAM 使用者變更自己的密碼。
- 僅允許選定的 IAM 使用者變更自己的密碼。在這種情況下您停用選項，讓所有使用者變更自己的密碼，並使用 IAM 政策僅為一些使用者授予許可。這種方法允許這些使用者變更自己的密碼以及選擇性的其他憑證，如他們自己的存取金鑰。

> **Important**
>
> 建議您設定自訂密碼政策，以要求 IAM 使用者建立強式密碼。

允許所有 IAM 使用者變更自己的密碼

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，按一下 Account Settings (帳戶設定)。
3. 在 Password policy (密碼政策) 區段中，選擇 Edit (編輯)。
4. 選擇 Custom (自訂) 以使用自訂密碼政策。
5. 選取 Allow users to change their own password (允許使用者變更自己的密碼)，然後選擇 Save changes (儲存變更)。這允許帳戶中的所有使用者僅針對其使用者存取 `iam:ChangePassword` 動作，並可存取 `iam:GetAccountPasswordPolicy` 動作。
6. 為使用者提供下列指示，以便變更其密碼：IAM 使用者如何變更自己的密碼。

如需有關可用來變更帳戶密碼原則 (包括讓所有使用者變更自己的密碼) 的 [Windows PowerShell 工具] 和 API 命令的相關資訊，請參閱設定密碼政策 (AWS CLI)。 AWS CLI

允許選定的 IAM 使用者變更自己的密碼

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，按一下 Account Settings (帳戶設定)。
3. 在 Password policy (密碼政策) 區段中，請確認未選取 Allow users to change their own password (允許使用者變更自己的密碼)。如果已選取此核取方塊，所有使用者可以更改自己的密碼。(請參閱之前的程序)。
4. 如果沒有使用者，可建立使用者，讓他們可以變更自己的密碼。如需詳細資訊，請參閱 在您的中建立 IAM 使用者 AWS 帳戶。
5. (選用) 為被允許變更密碼的使用者建立 IAM 群組，然後從之前的步驟將使用者加入到群組。如需詳細資訊，請參閱 管理 IAM 使用者群組。
6. 指派以下政策到群組。如需詳細資訊，請參閱 管理 IAM 政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:GetAccountPasswordPolicy",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ChangePassword",
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

   此原則會授與ChangePassword動作的存取權，讓使用者只能從主控台、Windows PowerShell 工具或 API 變更自己的密碼。 AWS CLI它也會授與GetAccountPasswordPolicy動作的存取權，以便讓使用者檢視目前的密碼原則；需要此權限，使用者才能在 [變更密碼] 頁面上檢視帳號密碼策略。使用者必須能夠讀取目前的密碼政策，以確保變更的密碼符合政策的要求。
7. 為使用者提供下列指示，以便變更其密碼：IAM 使用者如何變更自己的密碼。

如需詳細資訊

如需有關管理憑證的詳細資訊，請參閱下列主題：

- 允許 IAM 使用者變更自己的密碼
- 管理使用者密碼 AWS
- 設定 IAM 使用者的帳戶密碼政策
- 管理 IAM 政策
- IAM 使用者如何變更自己的密碼

IAM 使用者如何變更自己的密碼

如果您已獲得變更自己的 IAM 使用者密碼的權限，您可以使用中的特殊頁面 AWS Management Console 來執行此操作。您也可以使用 AWS CLI 或 AWS API。

主題

- 必要許可
- IAM 使用者如何變更他們自己的密碼 (主控台)
- IAM 使用者如何變更自己的密碼 (AWS CLI 或 AWS API)

必要許可

若要變更您 IAM 使用者的密碼，您必須擁有以下政策的許可：AWS：允許 IAM 使用者在安全登入資料頁面上變更自己的主控台密碼。

IAM 使用者如何變更他們自己的密碼 (主控台)

下列程序說明 IAM 使用者如何使用 AWS Management Console 變更自己的密碼。

變更您 IAM 使用者的密碼 (主控台)

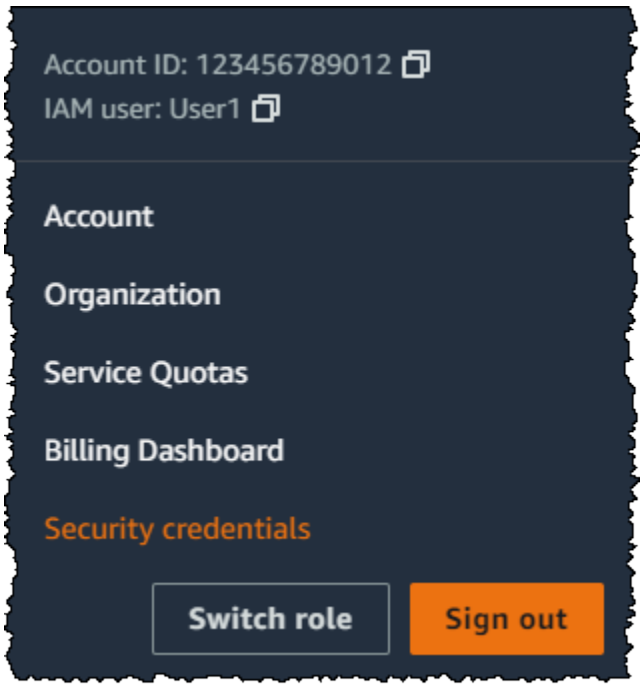
1. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 IAM 主控台。

> **Note**
>
> 為方便起見， AWS 登入頁面會使用瀏覽器 Cookie 來記住您的 IAM 使用者名稱和帳戶資訊。如果您先前以不同的使用者身分登入，請選擇在頁面底部附近的 Sign in to a different

account (登入不同的帳戶)，返回主要登入頁面。您可以在該處輸入帳 AWS 戶 ID 或帳戶別名，以重新導向至帳戶的 IAM 使用者登入頁面。

若要取得您的 AWS 帳戶 ID，請聯絡您的系統管理員。

2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 安全憑證 。



3. 在 AWS IAM 憑證索引標籤上，選擇更新密碼。
4. 針對 Current password (目前的密碼)，請輸入目前的密碼。輸入 New password (新密碼) 及 Confirm new password (確認新密碼) 的新密碼。接著選擇更新密碼。

> **Note**
>
> 新密碼必須符合帳戶密碼政策的要求。如需詳細資訊，請參閱 設定 IAM 使用者的帳戶密碼政策。

IAM 使用者如何變更自己的密碼 (AWS CLI 或 AWS API)

下列程序說明 IAM 使用者如何使用 AWS CLI 或 AWS API 變更自己的密碼。

若要變更您自己的 IAM 密碼，請使用下列：

- AWS CLI: `aws iam change-password`

- AWS API：ChangePassword

管理 IAM 使用者的存取金鑰

Follow us on Twitter

> **Important**
>
> 最佳實務的作法是使用臨時性安全憑證 (如 IAM 角色)，而不是建立長期憑證 (如存取金鑰)。在建立存取金鑰前，請檢閱長期存取金鑰的替代方案。

存取金鑰是 IAM 使用者或 AWS 帳戶根使用者的長期憑證。您可以使用存取金鑰來簽署 AWS CLI 或 AWS API 的程式設計要求 (直接或使用 AWS SDK)。如需詳細資訊，請參閱 簽署 AWS API 要求。

存取金鑰包含兩個部分：存取金鑰 ID (例如 AKIAIOSFODNN7EXAMPLE) 和私密存取金鑰 (例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY)。您必須一起使用存取金鑰 ID 和私密存取金鑰來驗證您的請求。

建立存取金鑰對時，將存取金鑰 ID 和私密存取金鑰儲存在安全位置。私密存取金鑰只會在您建立它的時候顯示一次。如果您遺失了私密存取金鑰，則必須刪除該存取金鑰並新建一個。如需詳細資訊，請參閱 重設遺失或忘記的密碼或存取金鑰 AWS。

每位使用者最多可以擁有兩個存取金鑰。

> **Important**
>
> 安全地管理存取金鑰。請勿將您的存取金鑰提供給未經授權的當事方，即便是協助尋找您的帳戶識別符也不妥。執行此作業，可能會讓他人能夠永久存取您的帳戶。

下列主題將詳細說明與存取金鑰相關的管理任務。

主題

- 管理存取金鑰所需的許可
- 管理存取金鑰 (主控台)
- 管理存取金鑰 (AWS CLI)
- 管理存取金鑰 (AWS API)

- 更新存取金鑰
- 保護存取金鑰
- 稽核存取金鑰

管理存取金鑰所需的許可

> **Note**
>
> `iam:TagUser` 是新增和編輯存取金鑰描述的選用許可。如需更多資訊，請參閱標記 IAM 使用者

為 IAM 使用者建立存取金鑰，您必須擁有以下政策的許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:GetUser",
                "iam:ListAccessKeys",
                "iam:TagUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

為您自己的 IAM 使用者更新存取金鑰，您必須擁有以下政策的許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:GetAccessKeyLastUsed",
                "iam:GetUser",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:TagUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

管理存取金鑰 (主控台)

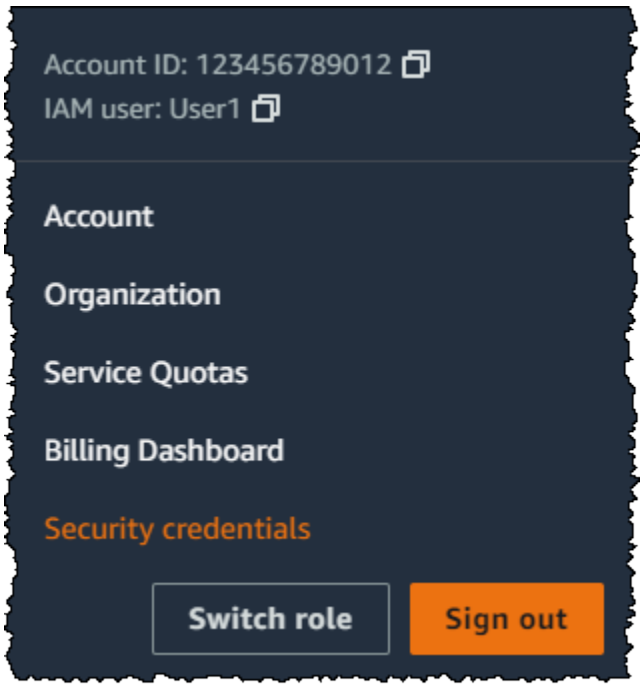
您可以使用 AWS Management Console 來管理 IAM 使用者的存取金鑰。

建立、修改或刪除您自己的存取金鑰 (主控台)

1. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 IAM 主控台。

   > **Note**
   >
   > 為方便起見， AWS 登入頁面會使用瀏覽器 Cookie 來記住您的 IAM 使用者名稱和帳戶資訊。如果您先前以不同的使用者身分登入，請選擇在頁面底部附近的 Sign in to a different account (登入不同的帳戶)，返回主要登入頁面。您可以在該處輸入帳 AWS 戶 ID 或帳戶別名，以重新導向至帳戶的 IAM 使用者登入頁面。

   若要取得您的 AWS 帳戶 ID，請聯絡您的系統管理員。
2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 安全憑證 。



執行以下任意一項：

建立存取金鑰

1. 在 Access keys (存取金鑰) 區段中，選擇 Create access key (建立存取金鑰)。如果您已經有兩個存取金鑰，則此按鈕將被停用，您必須先刪除一個存取金鑰才能建立新的存取金鑰。
2. 在 Access key best practices & alternatives (存取金鑰最佳實務與替代方案) 頁面上，選擇您的使用案例以了解可協助您避免建立長期存取金鑰的其他選項。若您判定您的使用案例仍需要存取金鑰，請選取 Other (其他)，然後再選擇 Next (下一步)。
3. (選用) 為存取金鑰設定描述標籤值。這將新增標籤鍵值對到您的 IAM 使用者。這可協助您在未來辨別與更新存取金鑰。標籤索引鍵被設定為存取金鑰 id。標籤值被設定為您指定的存取金鑰描述。完成時，選擇 Create access key (建立存取金鑰)。
4. 在 Retrieve access keys (擷取存取金鑰) 頁面上，選擇 Show (顯示) 以顯示您的使用者的私密存取金鑰的值，或選擇 Download .csv file (下載 .csv 檔案)。這是您儲存您的私密存取金鑰的唯一機會。在將您的私密存取金鑰儲存到安全位置以後，選取 Done (完成)。

要停用存取金鑰

- 在 Access keys (存取金鑰) 區段中，找到您想要停用的金鑰，然後選取 Actions (動作)，再選擇 Deactivate (停用)。出現確認提示時，請選取 Deactivate (停用)。停用的存取金鑰仍會計入兩個存取金鑰的限制。

要啟用存取金鑰

- 在 Access keys (存取金鑰) 區段中，找到要啟用的金鑰，然後選取 Actions (動作)，再選擇 Activate (啟用)。

要刪除不再需要的存取金鑰

- 在 Access keys (存取金鑰) 區段中，找到您想要刪除的金鑰，然後選取 Actions (動作)，再選擇 Delete (刪除)。依照對話中的指示先 Deactivate (停用)，然後再確認刪除。建議您在永久刪除存取金鑰前驗證不再使用該存取金鑰。

建立、修改或刪除另一 IAM 使用者的存取金鑰 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇要為其管理存取金鑰的使用者名稱，然後選擇 Security credentials (安全憑證) 索引標籤。
4. 在 Access keys (存取金鑰) 區段，執行下列項目：
   - 若要建立存取金鑰，選擇 Create access key (建立存取金鑰)。如果按鈕被停用，則您必須先刪除其中一個現有的金鑰，才能建立新的金鑰。在 Access key best practices & alternatives (存取金鑰最佳實務與替代方案) 頁面上，檢閱最佳實務和替代方案。選擇您的使用案例以了解可協助您避免建立長期存取金鑰的其他選項。若您判定您的使用案例仍需要存取金鑰，請選取 Other (其他)，然後再選擇 Next (下一步)。在 Retrieve access keys (擷取存取金鑰) 頁面上，選擇 Show (顯示) 以顯示您的使用者的私密存取金鑰的值。如要將存取金鑰 ID 和私密存取金鑰儲存到 `.csv` 檔案並在您電腦中的安全位置保管，請選擇 Download .csv file (下載 .csv 檔案) 按鈕。當您為您的使用者建立存取金鑰時，在預設情況下，該金鑰對是作用中的，且您的使用者可以立即使用該金鑰對。
   - 若要停用作用中的存取金鑰，請選取 Actions (動作)，然後再選擇 Deactivate (停用)。
   - 若要啟用非作用中的存取金鑰，請選取 Actions (動作)，然後再選擇 Activate (啟用)。

- 若要刪除您的存取金鑰，請選取 Actions (動作)，然後再選擇 Delete (刪除)。依照對話中的指示先 Deactivate (停用)，然後再確認刪除。 AWS 建議在執行此動作前，您要先停用該金鑰，然後測試不再使用該金鑰。使用時 AWS Management Console，您必須先停用金鑰才能刪除金鑰。

列出 IAM 使用者的存取金鑰 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 選擇目標使用者的名稱，然後選擇 Security credentials (安全憑證) 索引標籤。在 Access keys (存取金鑰) 區段中，您將看到使用者的存取金鑰和所顯示的每個金鑰的狀態。

> **Note**
>
> 只有使用者的存取金鑰 ID 是可見的。只有在建立金鑰時才能擷取私密存取金鑰。

列出多個 IAM 使用者的存取金鑰 ID (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 如有必要，請透過完成以下步驟將 Access key ID (存取金鑰 ID) 欄新增到使用者表格：
   a. 在最右側的表格上方，選擇設定圖示 ()。
   b. 在 Manage columns (管理欄) 中，選取 Access key ID (存取金鑰 ID)。
   c. 選擇 Close (關閉) 返回使用者清單。
4. Access key ID (存取金鑰 ID) 欄會顯示每個存取金鑰 ID，其後跟隨其狀態；例如，234782070278420732307623740 23 (Active) (作用中) 或 220937402396702370248434203 27 (Inactive) (非作用中)。

   您可以使用此資訊來查看和複製具有一個或兩個存取金鑰的使用者的存取金鑰。對於沒有存取金鑰的使用者，該欄會顯示 None (無)。

> **Note**
>
> 只有使用者的存取金鑰 ID 和狀態是可見的。只有在建立金鑰時才能擷取私密存取金鑰。

尋找哪位 IAM 使用者擁有特定的存取金鑰 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 在搜尋方塊中，輸入或貼上要尋找的使用者的存取金鑰 ID。
4. 如有必要，請透過完成以下步驟將 Access key ID (存取金鑰 ID) 欄新增到使用者表格：
   a. 在最右側的表格上方，選擇設定圖示 ( )。
   b. 在 Manage columns (管理欄) 中，選取 Access key ID (存取金鑰 ID)。
   c. 選擇 Close (關閉) 返回使用者清單，並確認已篩選的使用者擁有指定的存取金鑰。

管理存取金鑰 (AWS CLI)

若要從管理 IAM 使用者存取金鑰 AWS CLI，請執行下列命令。

- 建立存取金鑰：`aws iam create-access-key`
- 要停用或啟用存取金鑰：`aws iam update-access-key`
- 列出使用者的存取金鑰：`aws iam list-access-keys`
- 若要決定最近使用存取金鑰的時間：`aws iam get-access-key-last-used`
- 若要刪除存取金鑰：`aws iam delete-access-key`

管理存取金鑰 (AWS API)

若要從 AWS API 管理 IAM 使用者的存取金鑰，請呼叫下列操作。

- 建立存取金鑰：`CreateAccessKey`
- 要停用或啟用存取金鑰：`UpdateAccessKey`

- 列出使用者的存取金鑰：ListAccessKeys
- 若要決定最近使用存取金鑰的時間：GetAccessKeyLastUsed
- 若要刪除存取金鑰：DeleteAccessKey

更新存取金鑰

作為安全最佳實務，我們建議在需要時更新 IAM 使用者存取金鑰，如當員工離職時。如果 IAM 使用者已獲得所需許可，他們可以更新自己的存取金鑰。

如需有關授予 IAM 使用者許可，讓他們可更新自己的存取金鑰的詳細資訊，請參閱 AWS：允許 IAM 使用者在安全登入資料頁面上管理自己的密碼、存取金鑰和 SSH 公開金鑰。您還可以套用密碼政策到您的帳戶，以要求所有 IAM 使用者定期更新其密碼並規定必須多久更新一次。如需詳細資訊，請參閱設定 IAM 使用者的帳戶密碼政策。

主題

- 更新 IAM 使用者存取金鑰 (主控台)
- 更新存取金鑰 (AWS CLI)
- 更新存取金鑰 (AWS API)

更新 IAM 使用者存取金鑰 (主控台)

您可以從 AWS Management Console中更新存取金鑰。

在不會中斷您的應用程式下更新 IAM 使用者的存取金鑰 (主控台)

1. 當第一個存取金鑰仍然有效時，建立第二個存取金鑰。
    a. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
    b. 在導覽窗格中，選擇 Users (使用者)。
    c. 選擇目標使用者的名稱，然後選擇 Security credentials (安全憑證) 索引標籤。
    d. 在 Access keys (存取金鑰) 區段中，選擇 Create access key (建立存取金鑰)。在 Access key best practices & alternatives (存取金鑰最佳實務與替代方案) 頁面上，選取 Other (其他)，然後再選擇 Next (下一步)。
    e. (選用) 為存取金鑰設定描述標籤值，以新增標籤鍵值對到此 IAM 使用者。這可協助您在未來辨別與更新存取金鑰。標籤索引鍵被設定為存取金鑰 id。標籤值被設定為您指定的存取金鑰描述。完成時，選擇 Create access key (建立存取金鑰)。

f. 在 Retrieve access keys (擷取存取金鑰) 頁面上，選擇 Show (顯示) 以顯示您的使用者的私密存取金鑰的值，或選擇 Download .csv file (下載 .csv 檔案)。這是您儲存您的私密存取金鑰的唯一機會。在將您的私密存取金鑰儲存到安全位置以後，選取 Done (完成)。

   當您為您的使用者建立存取金鑰時，在預設情況下，該金鑰對是作用中的，且您的使用者可以立即使用該金鑰對。此時，使用者有兩個作用中的存取金鑰。

2. 更新所有應用程式和工具以使用新的存取金鑰。
3. 透過查看 Last used (上次使用) 資訊中最舊的存取金鑰，判斷第一個存取金鑰是否仍在使用中。其中一個方法是等待幾天，然後在繼續之前檢查舊的存取金鑰以供使用。
4. 即使 Last used (上次使用) 資訊指示從未使用舊金鑰，我們建議您不要立即刪除第一個存取金鑰。反之，選擇 Actions (動作)，然後選擇 Deactivate (停用) 來停用第一個存取金鑰。
5. 僅使用新的存取金鑰來確認您的應用程式正在工作。任何仍然使用原始存取金鑰的應用程式和工具，此時將停止運作，因為它們無法再存取 AWS 資源。如果您找到此類應用程式或工具，則可以重新啟用第一個存取金鑰。然後，返回 Step 3 並更新此應用程式以使用新的金鑰。
6. 等待一段時間後確保所有應用程式和工具都已更新，您可以刪除第一個存取金鑰：

   a. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
   b. 在導覽窗格中，選擇 Users (使用者)。
   c. 選擇目標使用者的名稱，然後選擇 Security credentials (安全憑證) 索引標籤。
   d. 在 Access keys (存取金鑰) 區段中，找到您想要刪除的存取金鑰，然後選取 Actions (動作)，再選擇 Delete (刪除)。依照對話中的指示先 Deactivate (停用)，然後再確認刪除。

要確定哪些存取金鑰需要更新或刪除 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 如有必要，請透過完成以下步驟將 Access key age (存取金鑰使用期限) 欄新增到使用者表格：

   a. 在最右側的表格上方，選擇設定圖示 ()。
   b. 在 Manage columns (管理欄) 中，選取 Access key age (存取金鑰使用期限)。
   c. 選擇 Close (關閉) 返回使用者清單。

4. Access key age (存取金鑰使用期限) 列顯示自建立最早的作用中存取金鑰以來的天數。您可以使用此資訊來尋找可能需要更新或刪除存取金鑰的使用者。對於沒有存取金鑰的使用者，該欄會顯示 None (無)。

更新存取金鑰 (AWS CLI)

您可以從 AWS Command Line Interface中更新存取金鑰。

在不會中斷您的應用程式下更新存取金鑰 (AWS CLI)

1. 當第一個存取金鑰仍然有效時，建立第二個存取金鑰，該索引鍵在預設情況下處於作用中。執行以下命令：

   - `aws iam create-access-key`

   此時，使用者有兩個作用中的存取金鑰。
2. 更新所有應用程式和工具以使用新的存取金鑰。
3. 使用此命令判斷第一個存取金鑰是否仍在使用中：

   - `aws iam get-access-key-last-used`

   其中一個方法是等待幾天，然後在繼續之前檢查舊的存取金鑰以供使用。
4. 即使步驟 Step 3 表示不使用舊金鑰，我們也建議您不要立即刪除第一個存取金鑰。反之，使用此命令將第一個存取金鑰的狀態變更為 `Inactive`：

   - `aws iam update-access-key`
5. 僅使用新的存取金鑰來確認您的應用程式正在工作。任何仍然使用原始存取金鑰的應用程式和工具，此時將停止運作，因為它們無法再存取 AWS 資源。如果找到此類應用程式或工具，則可以將其狀態切換回 `Active` 以重新啟用第一個存取金鑰。然後，返回步驟 Step 2 並更新此應用程式以使用新的金鑰。
6. 等待一段時間後確保所有應用程式和工具都已更新，可以使用此命令刪除第一個存取金鑰：

   - `aws iam delete-access-key`

更新存取金鑰 (AWS API)

您可以使用 AWS API 更新存取金鑰。

在不中斷應用程式的情況下更新存取金鑰 (AWS API)

1. 當第一個存取金鑰仍然有效時，建立第二個存取金鑰，該索引鍵在預設情況下處於作用中。呼叫以下操作：

   - `CreateAccessKey`

   此時，使用者有兩個作用中的存取金鑰。
2. 更新所有應用程式和工具以使用新的存取金鑰。
3. 透過呼叫此操作判斷第一個存取金鑰是否仍在使用中：

   - `GetAccessKeyLastUsed`

   其中一個方法是等待幾天，然後在繼續之前檢查舊的存取金鑰以供使用。
4. 即使步驟 Step 3 表示不使用舊金鑰，我們也建議您不要立即刪除第一個存取金鑰。反之，呼叫此操作將第一個存取金鑰的狀態變更為 `Inactive`：

   - `UpdateAccessKey`
5. 僅使用新的存取金鑰來確認您的應用程式正在工作。任何仍然使用原始存取金鑰的應用程式和工具，此時將停止運作，因為它們無法再存取 AWS 資源。如果找到此類應用程式或工具，則可以將其狀態切換回 `Active` 以重新啟用第一個存取金鑰。然後，返回步驟 Step 2 並更新此應用程式以使用新的金鑰。
6. 等待一段時間後確保所有應用程式和工具都已更新，您可以刪除呼叫此操作的第一個存取金鑰：

   - `DeleteAccessKey`

保護存取金鑰

擁有您存取金鑰的任何人都擁有與您相同的 AWS 資源存取層級。因此， AWS 竭盡全力保護您的訪問密鑰，並且，與我們的共享責任模式保持一致，您也應該如此。

展開下列各節，以取得協助您保護存取金鑰的指引。

> **Note**
>
> 您組織的安全要求和政策，可能會與本主題中所述的狀況不同。這裡提供的建議旨在做為一般指導方針。

移除 (或不產生) AWS 帳戶根使用者 存取金鑰

保護帳戶其中一個最佳方法是，不要有 AWS 帳戶根使用者的存取金鑰。除非您必須要有根使用者存取金鑰 (少數情況下)，否則最好不要產生存取金鑰。請改 AWS IAM Identity Center 為在中建立每日管理工作的管理使用者。如需如何在 IAM 身分中心建立管理使用者的詳細資訊，請參閱 IAM 身分中心使用者指南中的入門。

如果您已經有帳戶的根使用者存取金鑰，建議您執行以下步驟：找出應用程式中目前使用金鑰 (如果有) 的地方，以 IAM 使用者存取金鑰取代根使用者存取金鑰。然後停用並移除根使用者存取金鑰。如需有關如何更新存取金鑰的詳細資訊，請參閱 更新存取金鑰。

使用臨時安全憑證 (IAM 角色)，而不是長期存取金鑰

在許多情況下，您不必像 IAM 使用者一樣需要永遠不會過期的長期存取金鑰。反之，您可以建立 IAM 角色並產生暫時安全憑證。臨時安全登入資料包含存取金鑰 ID 和私密存取金鑰，但其中也包含指出登入資料何時到期的安全符記。

例如與 IAM 使用者和根使用者相關聯的長期存取金鑰，會一直保持有效，直到您手動撤銷為止。不過，透過 IAM 角色和其他功能取得的臨時安全登入資料 AWS Security Token Service 會在短時間後過期。使用臨時安全登入資料，可協助降低風險，以防登入資料意外洩露。

請在以下案例中使用 IAM 角色和臨時安全憑證：

- 您有一個應用程式或 AWS CLI 指令碼在 Amazon EC2 執行個體上執行。請勿直接在應用程式中使用存取金鑰。請勿將存取金鑰傳遞至應用程式、將其嵌入應用程式中，或讓應用程式從任何來源讀取存取金鑰。反之，請定義具有應用程式適當權許可的 IAM 角色，然後使用 EC2 角色啟動 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。這樣做會將 IAM 角色與 Amazon EC2 執行個體相關聯。這種做法也可讓應用程式取得臨時安全憑證，以便用來對 AWS進行程式化呼叫。 AWS SDK 和 AWS Command Line Interface (AWS CLI) 可以自動從角色取得臨時認證。
- 您需要授予跨帳戶存取權。使用 IAM 角色培養帳戶之間的信任，然後對一個帳戶中的使用者授予有限的許可，以存取信任的帳戶。如需詳細資訊，請參閱 IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶。
- 您有一個行動應用程式。請勿使用應用程式內嵌存取金鑰，甚至是內嵌於加密儲存中。反之，請使用 Amazon Cognito 來管理應用程式中的使用者身分。此服務可讓您使用 Login with Amazon、Facebook、Google 或任何與身分提供者相容的 OpenID Connect (OIDC)，驗證使用者的身分。然後，您可以使用 Amazon Cognito 憑證供應商，管理應用程式用來向 AWS提出請求的憑證。

- 您想要聯合到， AWS 且您的組織支援 SAML 2.0。如果您處理的組織是具有支援 SAML 2.0 的身分識別提供者的組織，請將提供者設定為使用 SAML。您可以使用 SAML 交換驗證資訊， AWS 並取回一組臨時安全登入資料。如需詳細資訊，請參閱 SAML 2.0 聯合身分。
- 您想要聯合到， AWS 且您的組織擁有內部部署識別身分存放區。如果使用者可以在組織內部進行驗證，您可以撰寫一個應用程式，讓他們發出臨時安全登入資料以存取 AWS 資源。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

> **Note**
>
> 您是否將 Amazon EC2 執行個體與需要以程式設計方式存取 AWS 資源的應用程式搭配使用？如果是，請使用適用於 EC2 的 IAM 角色。

適當管理 IAM 使用者存取金鑰

如果您必須建立以程式設計方式存取的存取金鑰 AWS，請為 IAM 使用者建立存取金鑰，並僅授與使用者所需的權限。

遵循這些預防措施，以協助保護 IAM 使用者存取金鑰：

- 請勿將存取金鑰直接嵌入程式碼。AWS SDK 和 AWS 命令列工具可讓您將存取金鑰放入已知位置，如此一來，就不必將其留在程式碼中。

  將存取金鑰放入以下其中一個位置：

  - 認 AWS 證檔案。 AWS SDK 並 AWS CLI 自動使用您存儲在憑據文件中的 AWS 憑據。

    如需使用 AWS 認證檔案的相關資訊，請參閱 SDK 的說明文件。範例包括AWS SDK for Java 開發人員指南中的設定認 AWS 證和區域，以及AWS Command Line Interface 使用者指南中的組態和認證檔案。

    若要儲存 AWS SDK for .NET 和的認證 AWS Tools for Windows PowerShell，建議您使用 SDK 存放區。如需詳細資訊，請參閱《AWS SDK for .NET 開發人員指南》中的使用 SDK Store。
  - 環境變數. 在多租用戶系統中，請選擇使用者環境變數，而不是系統環境變數。

    如需有關使用環境變數來存放憑證的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的環境變數。

- 對不同的應用程式使用不同的存取金鑰。如果它們已經公開，這麼做可以隔離許可，並撤銷個別應用程式的存取金鑰。針對不同的應用程式擁有獨立的存取金鑰，也會在 AWS CloudTrail 日誌檔中產生不同的項目。此組態可讓您更容易判斷哪個應用程式執行特定動作。
- 視需要更新存取金鑰。若有存取金鑰可能遭洩露的風險，請更新該存取金鑰並刪除先前的存取金鑰。如需詳細資訊，請參閱更新存取金鑰
- 移除未使用的存取金鑰。如果使用者離開組織，請移除相對應的 IAM 使用者，使該使用者無法再存取您的資源。要了解上次使用訪問密鑰的時間，請使用 GetAccessKeyLastUsedAPI (AWS CLI 命令：`aws iam get-access-key-last-used`) 。
- 為最機密的 API 操作使用臨時憑證並設定多重驗證。利用 IAM 政策，可以指定允許使用者呼叫的 API 操作。在某些情況下，您可能希望在允許使用者執行特別敏感的動作之前，要求使用者經過 AWS MFA 驗證的額外安全性。例如，您可能擁有允許使用者執行 Amazon EC2 `RunInstances`、`DescribeInstances` 與 `StopInstances` 動作的政策。但是您可能想要限制破壞性的動作，例如，`TerminateInstances`並確保使用者只有在使用 AWS MFA 裝置進行驗證時才能執行該動作。如需詳細資訊，請參閱 設定受 MFA 保護的 API 存取。

使用存取鍵存取行動應 AWS 用程式

您可以使用 AWS 移動應用程序訪問一組有限的 AWS 服務和功能。行動應用程式可協助您在外出時支援事件回應。如需詳細資訊及下載應用程式，請參閱 AWS 主控台行動應用程式。

您可以使用主控台密碼或存取金鑰登入行動應用程式。根據最佳實務，請勿使用根使用者存取金鑰。相反地，我們強烈建議您除了在行動裝置上使用密碼或生物識別鎖定外，還要建立專門用於使用行動應用程式管理 AWS 資源的 IAM 使用者。如果您遺失行動裝置，您可以移除 IAM 使用者的存取權。

使用存取金鑰登入 (行動應用程式)

1. 在行動裝置上開啟應用程式。
2. 如果這是您第一次將身分新增至裝置，請依序選擇 Add an identity (新增身分) 及選擇 Access keys (存取金鑰)。

   如果您已經使用其他身分登入，請依序選擇功能表圖示及 Switch identity (切換身分)。然後依序選擇 Sign in as a different identity (以不同的身分登入) 及 Access keys (存取金鑰)。
3. 在 Access keys (存取金鑰) 頁面輸入您的資訊。
   - 存取金鑰 ID – 輸入您的存取金鑰 ID。
   - 私密存取金鑰 – 輸入您的私密存取金鑰。

- 身分名稱 – 輸入將顯示在行動應用程式中的身分識別名稱。這不需要與您的 IAM 使用者名稱相符。
- 身分識別 PIN – 建立您將在未來登入期間使用的個人識別碼 (PIN)。

> **Note**
>
> 如果您啟用了 AWS 流動應用程式的生物辨識功能，系統會提示您使用指紋或臉部辨識來進行驗證，而非 PIN 碼。如果生物特徵辨識失敗，系統可能會提示您輸入 PIN 碼。

4. 選擇 Verify and add keys (驗證並新增金鑰)。

   您現在可以使用行動應用程式存取精選的一組資源。

相關資訊

下列主題提供設定 AWS SDK 和使用存取金鑰 AWS CLI 的指引：

- 在AWS SDK for Java 開發人員指南中設置 AWS 憑據和區域
- 《AWS SDK for .NET 開發人員指南》中的使用 SDK Store
- 《AWS SDK for PHP 開發人員指南》中的對 SDK 提供憑證
- 博托 3 (適用於 Python 的AWS SDK) 文件中的設定
- 《AWS Tools for Windows PowerShell 使用者指南》中的使用 AWS 憑證
- 《AWS Command Line Interface 使用者指南》中的組態和憑證檔案
- 《AWS SDK for .NET 開發人員指南》中的使用 IAM 角色授予存取權
- AWS SDK for Java 2.x 中的為 Amazon EC2 設定 IAM 角色

稽核存取金鑰

您可以檢閱程式碼中的 AWS 存取金鑰，以判斷金鑰是否來自您擁有的帳戶。您可以使用`aws sts get-access-key-info` AWS CLI 命令或 `GetAccessKeyInfo` AWS API 操作傳遞訪問密鑰 ID。

AWS CLI 和 AWS API 作業會傳回存取金鑰所屬的識別碼。 AWS 帳戶 開頭為 `AKIA` 的存取金鑰是 IAM 使用者或 AWS 帳戶根使用者的長期登入資料。以開頭的存取金鑰 ID `ASIA` 是使用 AWS STS 作業建立的暫時認證。如果回應中的帳戶是您的，您可以根使用者的身分登入並檢閱根使用者存取金鑰。然後，您可以提取憑證報告來了解擁有金鑰的是哪位 IAM 使用者。若要瞭解誰要求`ASIA`存取金鑰的臨時登入資料，請檢視 CloudTrail 記錄檔中的 AWS STS 事件。

基於安全考量，您可以檢閱 AWS CloudTrail 記錄檔以瞭解中執行動作的人員 AWS。您可以使用角色信任政策中的 `sts:SourceIdentity` 條件金鑰，請求使用者在擔任角色時指定身分。例如，您可以請求 IAM 使用者將自己的使用者名稱指定為其來源身分。這可以協助您判斷哪位使用者在 AWS中執行了特定動作。如需詳細資訊，請參閱 `sts:SourceIdentity`。

此操作不會指出存取金鑰的狀態。金鑰可能是作用中、非作用中或已刪除。作用中金鑰可能沒有操作的執行許可。提供已刪除的存取金鑰可能會傳回金鑰不存在的錯誤。

重設遺失或忘記的密碼或存取金鑰 AWS

> **Important**
>
> 登入時遇到問題 AWS嗎？ 請確定您位在使用者類型的正確 AWS 登入頁面。如果您是 AWS 帳戶根使用者 (帳戶擁有者)，則可以 AWS 使用建立 AWS 帳戶. 如果您是 IAM 使用者，您的帳戶管理員可以為您提供可用來登入 AWS的憑證。如果您需要請求支持，請不要使用此頁面上的反饋鏈接，因為表單是由 AWS 文檔團隊收到的，而不是 AWS Support。請改為在 Contact Us (聯絡我們) 頁面上選擇 Still unable to log into your AWS account (仍然無法登入您的帳戶)，然後選擇其中一個可用支援選項。

在主要登入頁面上，您必須輸入您的電子郵件地址以根使用者身分登入，或輸入您的帳戶 ID 以 IAM 使用者身分登入。您只能在符合使用者類型的登入頁面中提供密碼。如需了解詳細資訊，請參閱登入 AWS Management Console。

如果您位於正確的登入頁面，但遺失或忘記密碼或存取金鑰，您無法從 IAM 中擷取密碼或金鑰。請使用下列方法來重設密碼與索引鍵：

- AWS 帳戶根使用者 密碼 — 如果您忘記 root 使用者密碼，您可以從中重設密碼 AWS Management Console。如需詳細資訊，請參閱本主題後面部分的 the section called “重設遺失或忘記的根使用者密碼”。
- AWS 帳戶 存取金鑰 — 如果您忘記帳戶存取金鑰，您可以建立新的存取金鑰，而不會停用現有的存取金鑰。如果未在使用現有索引鍵，您可以將它們刪除。如需詳細資訊，請參閱 建立根使用者的存取金鑰 和 刪除根使用者的存取金鑰。
- IAM 使用者密碼 – 若您是 IAM 使用者且您忘記密碼，必須請求管理員重設密碼。若要了解管理員管理您的密碼的方式，請參閱 管理 IAM 使用者密碼。
- IAM 使用者存取金鑰 – 若您是 IAM 使用者且您忘記存取金鑰，將需要有新的存取金鑰。若您有自行建立存取金鑰的許可，可在 管理存取金鑰 (主控台) 中找到建立新金鑰的說明。若您沒有所需的許

可，必須請求管理員建立新的存取金鑰。如果您仍在使用舊索引鍵，請求您的管理員不要刪除舊索引鍵。若要了解管理員管理您的存取金鑰的方式，請參閱 管理 IAM 使用者的存取金鑰。

在中使用多因素身份驗證（MFA） AWS

Follow us on Twitter

為了提高安全性，我們建議您設定多重要素驗證 (MFA) 以協助保護您 AWS 的資源。您可以為 AWS 帳戶根使用者 和 IAM 使用者啟用 MFA。當您為根使用者啟用 MFA 時，它僅影響根使用者憑證。帳戶中的 IAM 使用者都有自己憑證的不同身分，並且每個身分都有自己的 MFA 組態。

您可以以目前受支援 MFA 類型的任意組合為您的 AWS 帳戶根使用者 和 IAM 使用者註冊最多八台 MFA 裝置。如需支援的 MFA 類型的詳細資訊，請參閱 適用於 IAM 使用者的可用 MFA 類型。對於多個 MFA 裝置，只需要一個 MFA 裝置即可透過該使用者 AWS CLI 身分登入 AWS Management Console 或建立工作階段。

> **Note**
>
> 我們建議您要求您的人類使用者在存取時使用臨時登入資料 AWS。你有沒有考慮過使用 AWS IAM Identity Center？ 您可以使用 IAM 身分中心集中管理多個存取權限， AWS 帳戶 並從單一位置為使用者提供所有指派帳戶的 MFA 保護的單一登入存取權。使用 IAM Identity Center，您可以在 IAM Identity Center 中建立和管理使用者身分，或輕鬆連線至您現有的 SAML 2.0 相容身分提供者。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

適用於 IAM 使用者的可用 MFA 類型

MFA 增加了額外的安全性，因為它要求用戶在訪問 AWS 網站或服務時從 AWS 支持的 MFA 機制提供唯一的身份驗證，以及他們的常規登錄憑據。 AWS 支援下列 MFA 類型：密碼金鑰和安全金鑰、虛擬驗證器應用程式，以及硬體 TOTP 權杖。

密碼金鑰和安全金鑰

AWS Identity and Access Management 支援 MFA 的密碼金鑰和安全金鑰。根據 FIDO 標準，密碼金鑰使用公開金鑰加密技術來提供比密碼更安全的強式、防網路釣魚驗證。 AWS 支援兩種類型的密碼金鑰：裝置繫結的密碼金鑰 (安全金鑰) 和同步的金鑰。

- 安全密鑰：這些是物理設備，例如 YubiKey，用作身份驗證的第二個因素。

- 同步的密鑰：這些使用提供商（例如谷歌，蘋果，Microsoft 帳戶以及第三方服務（如 1Password，Dashlane 和 Bitwarden）的憑據管理器作為第二個因素。

您可以使用內建的生物特徵驗證器 (例如 Apple 上的 Touch ID MacBooks 和電腦上的 Windows Hello 臉部辨識) 來解鎖憑證管理員並登入 AWS。密碼是由您選擇的供應商使用您的指紋、臉孔或裝置 PIN 碼建立的。您可以跨裝置同步密碼金鑰 AWS，以促進登入，進而增強可用性和可復原性。

FIDO Alliance 維護與 FIDO 規範相容的所有經 FIDO 認證的產品的清單。單一金鑰或安全金鑰可支援多個根使用者帳戶和 IAM 使用者。如需為 IAM 使用者啟用密碼金鑰和安全金鑰的詳細資訊，請參閱啟用金鑰或安全金鑰 (主控台)。

虛擬驗證器應用程式

虛擬身份驗證器應用程序在手機或其他設備上運行，並模擬物理設備。虛擬驗證器應用程式實作以時間為基礎的一次性密碼 (TOTP) 算法，並且支援在單台裝置上使用多個權杖。登入期間出現提示時，使用者必須輸入裝置的有效代碼。分配給用戶的每個令牌必須是唯一的。用戶無法從其他用戶的令牌中鍵入代碼進行身份驗證。

我們強烈建議您在等待硬體的購買核准或等待硬體就定位時，使用虛擬 MFA 裝置。如需可作為虛擬 MFA 裝置使用的一些受支援應用程式的清單，請參閱 M ulti-Factor Authentication (M FA)。如需為 IAM 使用者設定虛擬 MFA 裝置的指示，請參閱啟用虛擬多重要素驗證 (MFA) 裝置 (主控台)。

硬件 TOTP 令牌

硬體裝置會根據基於時間的一次性密碼 (TOTP) 演算法產生一個六位數的數字代碼。使用者必須在登入期間在第二個網頁上輸入裝置中的有效程式碼。每個指派給使用者的 MFA 裝置都必須是唯一的。使用者無法輸入另一個使用者裝置的代碼來進行身分驗證。如需支援硬體 MFA 裝置的相關資訊，請參閱 M ulti-Factor Authentication (M FA)。如需為 IAM 使用者設定硬體 TOTP 權杖的指示，請參閱啟用硬體 TOTP 權杖 (主控台)。

如果您想要使用實體 MFA 裝置，建議您使用安全金鑰作為硬體 TOTP 裝置的替代方案。安全金鑰具有無電池需求、網路釣魚阻力的優點，而且可在單一裝置上支援多個 root 和 IAM 使用者，以增強安全性。

> **Note**
>
> SMS 簡訊式 MFA - AWS 結束了對啟用 SMS 多重要素驗證 (MFA) 的支援。我們建議擁有 IAM 使用者使用 SMS 簡訊型 MFA 的客戶切換至下列其中一種替代方法：金鑰或安全金鑰、虛擬 (軟體型) MFA 裝置或硬體 MFA 裝置。您可以使用已分配的 SMS MFA 裝置來識別帳戶中的使

用者。為此，請前往 IAM 主控台，從導覽窗格中選擇 Users (使用者)，然後在表的 MFA 列中尋找具有 SMS 的使用者。

主題

為中的使用者啟用 MFA 裝置 AWS

設定 MFA 的步驟取決於您使用的 MFA 裝置類型。

主題

啟用 MFA 裝置的一般步驟

以下概觀程序說明如何設定和使用 MFA，並提供相關資訊的連結。

> **注意**
>
> 您還可以觀看此英語視頻，如何設置 M AWS ulti-Factor Authentication (MFA) 和 AWS 預算警報，以獲取更多信息。

1. 取得 MFA 裝置，如下列其中一項。每個 AWS 帳戶根使用者 或 IAM 使用者最多可啟用八個 MFA 裝置，其中包含下列任何類型的組合。
   - 虛擬 MFA 裝置，一種符合 RFC 6238 (以標準為基礎的 TOTP (以時間為基礎的單次密碼) 演算法) 的軟體應用程式。您可以在手機或其他裝置上安裝此應用程式。如需可以用來做為虛擬 MFA 裝置的支援應用程式清單，請參閱多重要素驗證。
   - 具有AWS 支援組態的金鑰或安全金鑰。FIDO Alliance 維護與 FIDO 規範相容的所有經 FIDO 認證的產品的清單。
   - 來自第三方供應商的硬體型 MFA 裝置，例如 Token 裝置。這些令牌專門用於 AWS 帳戶. 如需詳細資訊，請參閱 啟用硬體 TOTP 權杖 (主控台)。您只能使用具有安全共享其唯一令牌種子的令牌 AWS。令牌種子是在令牌生產時生成的秘密密鑰。從其他來源購買的令牌將無法與 IAM 一起使用。為確保兼容性，您必須從以下鏈接之一購買硬件 MFA 設備：OTP 令牌或 OTP 顯示卡。
2. 啟用 MFA 裝置。
   - 虛擬或硬體 TOTP 權杖 — 您可以使用 AWS CLI 命令或 AWS API 作業為 IAM 使用者啟用虛擬 MFA 裝置。您無法使用 AWS CLI、 AWS API、Windows PowerShell 工 AWS 帳戶根使用者 具或任何其他命令列工具來啟用 MFA 裝置。不過，您可以使用 AWS Management Console 為根使用者啟用 MFA 裝置。
   - 密碼金鑰和安全金鑰 — 具有密碼金鑰或安全金鑰的根使用者和 IAM 使用者可以從 AWS Management Console 唯一啟用，而非從 AWS CLI 或 AWS API 啟用。

   如需啟用每種類型 MFA 裝置的資訊，請參閱下列頁面：
   - 虛擬 MFA 裝置：啟用虛擬多重要素驗證 (MFA) 裝置 (主控台)
   - 密碼金鑰和安全金鑰：啟用金鑰或安全金鑰 (主控台)
   - 硬體 TOTP 權杖：啟用硬體 TOTP 權杖 (主控台)
3. 啟用多太 MFA 裝置 (建議使用)
   - 我們建議您為. 中的 AWS 帳戶根使用者 AWS 帳戶和 IAM 使用者啟用多個 MFA 裝置。這允許您提高 AWS 帳戶 中的安全標準，並對高度權限使用者 (例如 AWS 帳戶根使用者) 的存取權管理進行簡化。
   - 您最多可以向您 AWS 帳戶根使用者 和 IAM 使用者註冊八個 MFA 裝置，其中包括目前支援的 MFA 類型的任何組合。對於多個 MFA 裝置，您只需要一個 MFA 裝置即可透過該使用者 AWS CLI 身分登入 AWS Management Console 或建立工作階段。IAM 使用者必須使用現有的 MFA 裝置進行身分驗證，才能啟用或停用其他 MFA 裝置。
   - 如果 MFA 裝置遺失、遭竊或無法存取，您可以使用剩餘的其中一個 MFA 裝置存取， AWS 帳戶而無需執行 AWS 帳戶 復原程序。如果 MFA 裝置遺失或遭竊，應取消 MFA 裝置與 IAM 主體的可能關聯。

- 使用多個 MFA 可讓您位於不同地理位置的員工或遠端工作，使用硬體式 MFA 進行存取， AWS 而無需協調員工之間單一硬體裝置的實體交換。
- 針對 IAM 主體使用額外的 MFA 裝置，可讓您在日常使用期間使用一或多個 MFA 裝置，同時將實體 MFA 裝置維護在安全的實體位置 (例如文件庫)，或是用於備份和備援的安全位置。

4. 登入或存取 AWS 資源時，使用 MFA 裝置。
   - 密碼金鑰和安全金鑰 — 若要存取 AWS 網站，請輸入您的認證，然後根據您擁有的金鑰類型，點選 FIDO 安全金鑰、輸入裝置 PIN 碼，或在出現提示時提供指紋或臉孔。
   - 虛擬 MFA 裝置和硬體 TOTP 權杖 — 若要存取 AWS 網站，除了使用者名稱和密碼之外，您還需要裝置上的 MFA 代碼。

     若要存取 MFA 保護的 API 操作，您需要以下資訊：
     - MFA 代碼
     - MFA 裝置的識別符 (實體裝置的裝置序號或者在 AWS中定義的虛擬裝置的 ARN)
     - 平常使用的存取金鑰 ID 及私密存取金鑰

> 備註
>
> - 您無法將 FIDO 安全性金鑰的 MFA 資訊傳遞至 AWS STS API 作業，以要求臨時登入資料。
> - 您無法使用 AWS CLI 命令或 AWS API 作業來啟用 FIDO 安全金鑰。
> - 您不能為多個根或 IAM MFA 裝置使用相同的名稱。

如需更多詳細資訊，請參閱 透過您的 IAM 登入頁面來使用 MFA 裝置。

啟用金鑰或安全金鑰 (主控台)

密碼金鑰是一種多重要素驗證 (MFA) 裝置，可用來保護資源。 AWS AWS 支持同步的密鑰和設備綁定密鑰（也稱為安全密鑰）。

同步的密碼金鑰可讓 IAM 使用者在其許多裝置 (甚至是新裝置) 上存取其 FIDO 登入憑證，而不必在每個帳戶上重新註冊每個裝置。同步的密鑰包括谷歌，蘋果和 Microsoft 等第一方憑據管理器以及第三方憑據管理器，例如 1Password，達什巷和 Bitwarden 作為第二個因素。您還可以使用設備上的生物識別技術（例如，TouchID，FaceID，Windows Hello）來解鎖您選擇的憑據管理器以使用密鑰。

或者，裝置繫結的密碼會繫結至 FIDO 安全金鑰，您可以插入電腦上的 USB 連接埠，然後在提示時點選以安全地完成登入程序。如果您已經將 FIDO 安全性金鑰與其他服務搭配使用，而且該金鑰具

有AWS 受支援的組態 (例如 Yubico 的 YubiKey 5 系列)，您也可以搭配使用。 AWS否則，如果要在中用 WebAuthn 於 MFA，則需要購買 FIDO 安全金鑰。 AWS此外，FIDO 安全金鑰可支援同一裝置上的多個 IAM 或 root 使用者，進而增強其帳戶安全性的公用程式。如需兩種裝置類型的規格及購買資訊，請參閱多重要素驗證。

您最多可以向您 AWS 帳戶根使用者 和 IAM 使用者註冊八個 MFA 裝置，其中包括目前支援的 MFA 類型的任何組合。對於多個 MFA 裝置，您只需要一個 MFA 裝置即可透過該使用者 AWS CLI 身分登入 AWS Management Console 或建立工作階段。我們建議您註冊多個 MFA 裝置。例如，您可以註冊內建驗證器，也可以註冊存放在實體安全位置的安全金鑰。如果無法使用內建驗證器，則可以使用已註冊的安全金鑰。對於驗證器應用程式，我們也建議您在這些應用程式中啟用雲端備份或同步功能，以協助避免在具有驗證器應用程式的裝置遺失或損壞時，失去對帳戶的存取權限。

> **Note**
>
> 存取 AWS時，我們建議您要求人類使用者使用暫時性憑證。您的使用者可以 AWS 與身分識別提供者聯合，在該提供者中使用其公司認證和 MFA 組態進行驗證。若要管理對應用程式的存取 AWS 和商業應用程式，建議您使用 IAM 身分中心。如需詳細資訊，請參閱 IAM 身分中心使用者指南。

主題

- 必要許可
- 為您自己的 IAM 使用者 (主控台) 啟用金鑰或安全金鑰
- 為其他 IAM 使用者 (主控台) 啟用金鑰或安全金鑰
- 取代金鑰或安全金鑰
- 使用金鑰和安全金鑰的支援組態

必要許可

若要為自己的 IAM 使用者管理 FIDO 金鑰，同時保護敏感的 MFA 相關動作，您必須擁有下列政策的許可：

> **Note**
>
> ARN 值是靜態值，不是哪項協定被用來註冊驗證器的指示器。我們已經棄用了 U2F，因此所有新實現都使用 WebAuthn.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

為您自己的 IAM 使用者 (主控台) 啟用金鑰或安全金鑰

您可以 AWS Management Console 僅從或 AWS API 為自己的 IAM 使用者啟用金鑰 AWS CLI 或安全金鑰。您必須擁有裝置的實體存取權，才能啟用安全金鑰。

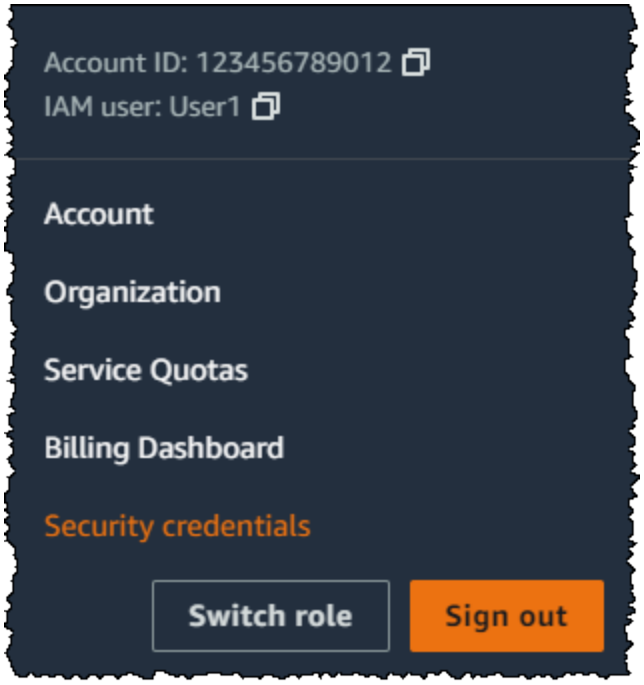
為您自己的 IAM 使用者 (主控台) 啟用金鑰或安全金鑰

1. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 IAM 主控台。

> **Note**
>
> 為方便起見，AWS 登入頁面會使用瀏覽器 Cookie 來記住您的 IAM 使用者名稱和帳戶資訊。如果您先前以不同的使用者身分登入，請選擇在頁面底部附近的 Sign in to a different account (登入不同的帳戶)，返回主要登入頁面。您可以在該處輸入帳 AWS 戶 ID 或帳戶別名，以重新導向至帳戶的 IAM 使用者登入頁面。

若要取得您的 AWS 帳戶 ID，請聯絡您的系統管理員。

2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 安全憑證 。



3. 在選取的 IAM 使用者頁面上，選擇安全登入資料索引標籤。
4. 在 Multi-Factor Authentication (MFA) (多重要素驗證 (MFA)) 區段下方，選擇 Assign MFA device (指派 MFA 裝置)。
5. 在 MFA 裝置名稱頁面上，輸入裝置名稱，選擇 [金鑰] 或 [安全金鑰]，然後選擇 [下一步]。
6. 在 [設定裝置] 上，設定您的金鑰。使用生物辨識資料 (例如臉部或指紋) 建立金鑰、使用裝置 PIN 碼，或將 FIDO 安全金鑰插入電腦的 USB 連接埠並輕觸，以建立金鑰。
7. 按照瀏覽器上的說明進行操作，然後選擇「繼續」。

您現在已註冊您的金鑰或安全金鑰，以便搭配 AWS使用。若要取得有關搭配使用 MFA 的資訊 AWS Management Console，請參閱透過您的 IAM 登入頁面來使用 MFA 裝置。

為其他 IAM 使用者 (主控台) 啟用金鑰或安全金鑰

您可以 AWS Management Console 僅從或 AWS API 為其他 IAM 使用者啟用金鑰 AWS CLI 或安全性。

為其他 IAM 使用者 (主控台) 啟用金鑰或安全性

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇使用者 。
3. 在 [使用者] 底下，選擇您要啟用 MFA 的使用者名稱。
4. 在選取的 IAM 使用者頁面上，選擇安全登入資料索引標籤。
5. 在 Multi-Factor Authentication (MFA) (多重要素驗證 (MFA)) 區段下方，選擇 Assign MFA device (指派 MFA 裝置)。
6. 在 MFA 裝置名稱頁面上，輸入裝置名稱，選擇 [金鑰] 或 [安全金鑰]，然後選擇 [下一步]。
7. 在 [設定裝置] 上，設定您的金鑰。使用生物辨識資料 (例如臉部或指紋) 建立金鑰、使用裝置 PIN 碼，或將 FIDO 安全金鑰插入電腦的 USB 連接埠並輕觸，以建立金鑰。
8. 按照瀏覽器上的說明進行操作，然後選擇「繼續」。

您現在已為其他 IAM 使用者註冊金鑰或安全金鑰，以便與其他 IAM 使用者搭配 AWS使用。若要取得有關搭配使用 MFA 的資訊 AWS Management Console，請參閱透過您的 IAM 登入頁面來使用 MFA 裝置。

取代金鑰或安全金鑰

與您 AWS 帳戶根使用者 和 IAM 使用者一次最多可以指派給使用者的任何目前支援 MFA 類型組合的八個 MFA 裝置。如果使用者遺失 FIDO 驗證器或因為任何原因需要更換，您必須先停用舊的 FIDO 驗證器。然後，再為使用者新增新的 MFA 裝置。

- 如需停用目前與 IAM 使用者相關聯的裝置，請參閱 停用 MFA 裝置。
- 若要為 IAM 使用者新增新的 FIDO 安全性金鑰，請參閱 為您自己的 IAM 使用者 (主控台) 啟用金鑰或安全金鑰。

如果您無法存取新的金鑰或安全金鑰，您可以啟用新的虛擬 MFA 裝置或硬體 TOTP 權杖。請參閱以下其中一項以取得說明：

- 啟用虛擬多重要素驗證 (MFA) 裝置 (主控台)
- 啟用硬體 TOTP 權杖 (主控台)

使用金鑰和安全金鑰的支援組態

您可以使用 FIDO2 裝置繫結的密碼金鑰 (也稱為安全金鑰) 做為 IAM 使用目前支援的組態的多重要素驗證 (MFA) 方法。其中包括 IAM 支援的 FIDO2 裝置，以及支援 FIDO2 的瀏覽器。在註冊 FIDO2 裝置之前，請確認您使用的是最新的瀏覽器和作業系統 (OS) 版本。功能在不同瀏覽器、驗證器和作業系統用戶端上的行為可能有所不同。如果您在一個瀏覽器上註冊裝置失敗，您可以嘗試使用其他瀏覽器註冊。

FIDO2 是 FIDO U2F 的開放認證標準，也是 FIDO U2F 的延伸，可根據公有金鑰加密技術提供相同的高度安全性。FIDO2 由 W3C 網頁驗證規範 (WebAuthn API) 和 FIDO 聯盟用戶端對驗證器通訊協定 (CTAP) 組成，這是一種應用程式層通訊協定。CTAP 可透過外部驗證器在用戶端或平台之間進行通訊，例如瀏覽器或作業系統。當您在中啟用 FIDO 認證驗證器時 AWS，安全金鑰會建立新的金鑰組，僅供搭配使用。 AWS首先，您需要輸入您的憑證。出現提示時，您點選安全金鑰，此金鑰會回應由發出的驗證挑戰 AWS。若要進一步了解 FIDO2 標準，請參閱 FIDO2 專案。

AWS支援的 FIDO2 裝置

IAM 支援透過 USB、藍牙或 NFC 連接至您裝置的 FIDO2 安全裝置。IAM 還支持平台身份驗證器，例如觸摸 ID，面 ID 或視窗你好。

> **Note**
>
> AWS 需要存取電腦上的實體 USB 連接埠，以驗證您的 FIDO2 裝置。安全金鑰不適用於虛擬機器、遠端連線或瀏覽器的無痕模式。

FIDO Alliance 會維護與 FIDO 規範相容之所有 FIDO2 產品的清單。

支援 FIDO2 的瀏覽器

在網頁瀏覽器中執行的 FIDO2 安全裝置的可用性取決於瀏覽器和作業系統的組合。下列瀏覽器目前支援使用安全金鑰：

	macOS 10.15+	Windows 10	Linux	iOS 14.5+	Android 7+
Chrome	是	是	是	是	否
Safari	是	否	否	是	否
Edge	是	是	否	是	否
Firefox	是	是	否	是	否

> **Note**
>
> 大多數目前支援 FIDO2 的 Firefox 版本預設不會啟用支援。有關在火狐瀏覽器中啟用 FIDO2 支持的說明，請參閱對 FIDO 安全性金鑰進行故障診斷。

如需有關支援 FiDO2 認證裝置之瀏覽器的詳細資訊 YubiKey，請參閱 FIDO2 和 U2F 的作業系統和網頁瀏覽器支援。

瀏覽器外掛程式

AWS 僅支援本機支援 FIDO2 的瀏覽器。 AWS 不支持使用插件添加 FIDO2 瀏覽器支持。某些瀏覽器外掛程式與 FIDO2 標準不相容，因此可能會導致 FIDO2 安全金鑰無法預期的結果。

如需停用瀏覽器外掛程式和其他故障診斷提示，請參閱 我無法啟用 FIDO 安全性金鑰。

裝置認證

我們只會在註冊安全金鑰期間擷取和指派裝置相關認證，例如 FIPS 驗證和 FIDO 認證等級。從 FIDO Alliance Metadata Service (MDS) 中擷取裝置認證。如果安全金鑰的認證狀態或等級變更，它不會自動反映在裝置標籤中。若要更新裝置的認證資訊，請再次註冊裝置以擷取更新的認證資訊。

AWS 在裝置註冊期間提供下列認證類型作為條件索引鍵，可從 FIDO MDS 中取得：
FIPS-140-2、FIPS-140-3 和 FIDO 認證等級。您可以根據偏好的認證類型和等級，在其 IAM 政策中指定特定驗證器的註冊。如需詳細資訊，請參閱以下政策。

裝置認證的範例政策

下列使用案例顯示可讓您使用 FIPS 認證註冊 MFA 裝置的範例政策。

主題

使用案例 1：僅允許註冊具有 FIPS-140-2 L2 認證的裝置

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        }
    ]
}
```

使用案例 2：允許註冊具有 FIPS-140-2 L2 和 FIDO L1 認證的裝置

```
{
    "Version": "2012-10-17",
    "Statement": [{
```

```
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2",
                    "iam:FIDO-certification": "L1"
                }
            }
        }
    ]
}
```

使用案例 3：允許註冊具有 FIPS-140-2 L2 或 FIPS-140-3 L2 認證的裝置

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L2"
                }
            }
        }
    ]
}
```

使用案例 4：允許註冊具有 FIPS-140-2 L2 憑證的裝置，並支援其他 MFA 類型，例如虛擬身分驗證器和硬體 TOTP

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey": "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey": "Activate",
```

```
                    "iam:FIPS-140-2-certification": "L2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "Null": {
                    "iam:RegisterSecurityKey": "true"
                }
            }
        }
    ]
}
```

AWS CLI 和 AWS API

AWS 僅支援在. AWS Management Console和 API 中不支援針對 MFA 使用密碼金鑰AWS CLI和安全金鑰，也不支援存取 MFA 保護的 AWS API 作業。

其他資源

- 如需中使用金鑰和安全金鑰的詳細資訊 AWS，請參閱啟用金鑰或安全金鑰 (主控台)。
- 如需中密碼金鑰和安全金鑰疑難排解的說明 AWS，請參閱對 FIDO 安全性金鑰進行故障診斷。
- 如需有關 FIDO2 支援的一般行業資訊，請參閱 FIDO2 專案。

啟用虛擬多重要素驗證 (MFA) 裝置 (主控台)

您可以使用手機或其他裝置，做為虛擬多重要素驗證 (MFA) 裝置。若要執行此操作，安裝與 RFC 6238 (以標準為基礎的 TOTP (以時間為基礎的單次密碼) 演算法) 相容的行動應用程式。這些應用程式產生六位數的身分驗證代碼。由於它們能在不安全的行動裝置上執行，虛擬 MFA 可能不會提供與 FIDO 安全金鑰相同的安全層級。我們強烈建議您在等待硬體的購買核准或等待硬體就定位時，使用虛擬 MFA 裝置。

大多數虛擬 MFA 應用程序支持創建多個虛擬設備，允許您為多個 AWS 帳戶 或用戶使用同一個應用程序。您最多可以向您 AWS 帳戶根使用者 和 IAM 使用者註冊八個 MFA 裝置，其中包括目前支援的 MFA 類型的任何組合。對於多個 MFA 裝置，您只需要一個 MFA 裝置即可透過該使用者 AWS CLI 身分登入 AWS Management Console 或建立工作階段。我們建議您註冊多個 MFA 裝置。對於驗證器應

用程式，我們也建議您在這些應用程式中啟用雲端備份或同步功能，以協助避免在具有驗證器應用程式的裝置遺失或損壞時，失去對帳戶的存取權限。

如需您可以使用的虛擬 MFA 應用程式清單，請參閱多重要素驗證。 AWS 需要可產生六位數 OTP 的虛擬 MFA 應用程式。

主題

- 必要許可
- 針對 IAM 使用者啟用虛擬 MFA 裝置 (主控台)
- 取代虛擬 MFA 裝置

必要許可

若要為 IAM 使用者AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的 MFA 裝置管理虛擬 MFA 裝置時，您必須擁有以下政策的許可：。

針對 IAM 使用者啟用虛擬 MFA 裝置 (主控台)

您可以在中使用 IAM 為帳戶中的 IAM 使用者啟用和管理虛擬 MFA 裝置。 AWS Management Console 您可以將標籤連接至 IAM 資源 (包括虛擬 MFA 裝置)，以識別、整理和控制其存取權。只有在使用 AWS CLI 或 AWS API 時，才能標記虛擬 MFA 裝置。若要使用 AWS CLI 或 AWS API 啟用和管理 MFA 裝置，請參閱啟用及管理虛擬 MFA 裝置 (AWS CLI 或 AWS API)。如需標記 IAM 資源的詳細資訊，請參閱 標記 IAM 資源。

> **Note**
>
> 您必須擁有實體存取託管使用者的虛擬 MFA 裝置的硬體，才能設定 MFA。例如，您可以為使用者設定 MFA，該使用者將使用在智慧型手機上執行的虛擬 MFA 裝置。在這種情況下，您必須有可用的智慧型手機，才能完成精靈。因此，您可能想要讓使用者設定和管理自己的虛擬 MFA 裝置。在這種情況下，您必須授予使用者執行必要 IAM 動作的許可。如需詳細資訊以及授予這些許可的 IAM 政策範例，請參閱 IAM 教學課程：允許使用者管理其憑證和 MFA 設定和範例政策 AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的 MFA 裝置。

為 IAM 使用者啟用虛擬 MFA 裝置 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在導覽窗格中，選擇使用者 。
3. 在使用者清單中選擇 IAM 使用者名稱。
4. 選擇 安全憑證 標籤。在 Multi-Factor Authentication (MFA) (多重要素驗證 (MFA)) 區段下方，選擇 Assign MFA device (指派 MFA 裝置)。
5. 在精靈中輸入 Device name，然後選擇 驗證器應用程式，再選擇 下一步。

   IAM 將產生並顯示虛擬 MFA 裝置的配置資訊，包括 QR 碼圖形。此圖形代表「私密組態金鑰」，可用來在不支援 QR 碼的裝置上手動輸入。
6. 開啟您的虛擬 MFA 應用程式。如需可以用於託管虛擬 MFA 裝置的應用程式清單，請參閱多重要素驗證。

   如果虛擬 MFA 應用程式支援多個虛擬 MFA 裝置或帳戶，請選擇對應的選項以建立新的虛擬 MFA 裝置或帳戶。
7. 判定 MFA 應用程式是否支援 QR 碼，然後執行以下操作之一：

   - 從精靈中，選擇 Show QR code (顯示 QR 碼)，然後使用應用程式掃描 QR 碼。例如，您可選擇相機圖示或選擇與 Scan code (掃描碼) 類似的選項，然後使用裝置的相機掃描碼。
   - 在精靈中，選擇 Show secret key (顯示私密金鑰)，然後在您的 MFA 應用程式中輸入私密金鑰。

   完成操作後，虛擬 MFA 裝置會開始產生一次性密碼。
8. 在 設定裝置 頁面中的 MFA 代碼 1 方塊內輸入虛擬 MFA 裝置上目前顯示的一次性密碼。請等待 30 秒，裝置將產生新的一次性密碼。然後將第二個一次性密碼輸入 MFA code 2 (MFA 代碼 2) 方塊中。選擇 Add MFA (新增 MFA)。

   > **Important**
   >
   > 產生代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

虛擬 MFA 裝置現在已準備就緒，可與 AWS. 若要取得有關搭配使用 MFA 的資訊 AWS Management Console，請參閱透過您的 IAM 登入頁面來使用 MFA 裝置。

取代虛擬 MFA 裝置

您最多可以向您 AWS 帳戶根使用者 和 IAM 使用者註冊八個 MFA 裝置，其中包括目前支援的 MFA 類型的任何組合。如果使用者遺失裝置或基於任何原因需要汰換，您必須先停用舊裝置。然後，再為使用者加入新的裝置。

- 如需停用目前與另一個 IAM 使用者相關聯的裝置，請參閱 停用 MFA 裝置。
- 若要新增另一個 IAM 使用者的取代用虛擬 MFA 裝置，請遵循上方 針對 IAM 使用者啟用虛擬 MFA 裝置 (主控台) 程序中的步驟。
- 若要新增的取代虛擬 MFA 裝置 AWS 帳戶根使用者，請遵循程序針對 AWS 帳戶根使用者 啟用虛擬 MFA 裝置 (主控台)中的步驟。

啟用硬體 TOTP 權杖 (主控台)

硬體 TOTP 權杖會在以時間為基礎的一次性密碼 (TOTP) 演算法的基礎上產生六位數字程式碼。使用者必須在登入程序期間出現提示時輸入裝置中的有效代碼。指派給使用者的每個 MFA 裝置必須是唯一的；使用者不能從另一個使用者的裝置中輸入代碼進行身分驗證。MFA 裝置無法跨帳戶或使用者共用。

硬體 TOTP 權杖與 FIDO 安全金鑰都是您購買的實體裝置。硬體 MFA 裝置會在您登入時產生 TOTP 代碼以進行驗證。 AWS他們依賴電池， AWS 隨著時間的推移，可能需要更換和重新同步。FIDO 安全金鑰使用公開金鑰加密技術，不需要電池，並提供順暢的驗證程序。我們建議使用 FIDO 安全密鑰來抵抗網絡釣魚，這為 TOTP 設備提供了更安全的替代方案。此外，FIDO 安全金鑰可支援同一裝置上的多個 IAM 或 root 使用者，進而增強其帳戶安全性的公用程式。如需兩種裝置類型的規格及購買資訊，請參閱多重要素驗證。

您可以從 AWS Management Console、命令列或 IAM API 為 IAM 使用者啟用硬體 TOTP 權杖。若要為您的啟用 MFA 裝置 AWS 帳戶根使用者，請參閱啟用 AWS 帳戶根使用者 （ 控制台 ）的硬件 TOTP 令牌。

您最多可以向您 AWS 帳戶根使用者 和 IAM 使用者註冊八個 MFA 裝置，其中包括目前支援的 MFA 類型的任何組合。對於多個 MFA 裝置，您只需要一個 MFA 裝置即可透過該使用者 AWS CLI 身分登入 AWS Management Console 或建立工作階段。

> **Important**
>
> 建議您為您的使用者啟用多台 MFA 裝置，以便在一台 MFA 裝置遺失或無法存取時繼續存取您的帳戶。

> **Note**
>
> 如果您想要從命令列啟用 MFA 裝置，請使用 `aws iam enable-mfa-device`。若要使用 IAM API 啟用 MFA 裝置，請使用 `EnableMFADevice` 操作。

主題

- 必要許可
- 啟用您 IAM 使用者的硬體 TOTP 權杖 (主控台)
- 啟用另一個 IAM 使用者的硬體 TOTP 權杖 (主控台)
- 取代實體 MFA 裝置

必要許可

若要管理您 IAM 使用者的硬體 TOTP 權杖，同時保護敏感的 MFA 相關動作，您必須擁有下列政策的許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
```

```
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

啟用您 IAM 使用者的硬體 TOTP 權杖 (主控台)

您可以從 AWS Management Console啟用您自己的硬體 TOTP 權杖。

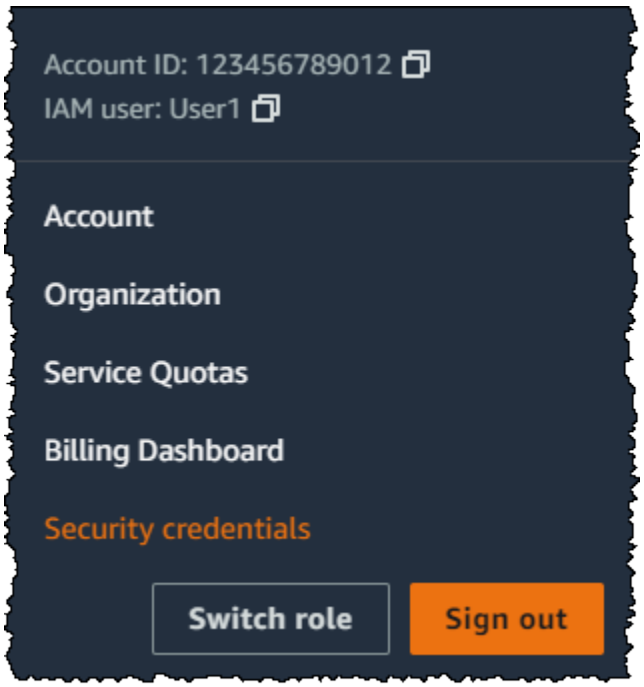
> **Note**
>
> 在您可以啟用硬體 TOTP 權杖之前，您必須擁有裝置的實體存取權。

要啟用您 IAM 使用者的硬體 TOTP 權杖 (主控台)

1. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 IAM 主控台。

   > **Note**
   >
   > 為方便起見， AWS 登入頁面會使用瀏覽器 Cookie 來記住您的 IAM 使用者名稱和帳戶資訊。如果您先前以不同的使用者身分登入，請選擇在頁面底部附近的 Sign in to a different account (登入不同的帳戶)，返回主要登入頁面。您可以在該處輸入帳 AWS 戶 ID 或帳戶別名，以重新導向至帳戶的 IAM 使用者登入頁面。

   若要取得您的 AWS 帳戶 ID，請聯絡您的系統管理員。
2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 安全憑證 。



3. 在 AWS IAM credentials (IAM 憑證) 索引標籤上，在 Multi-factor authentication (MFA) (多重要素驗證 (MFA)) 區段，選擇 Assign MFA device (指派 MFA 裝置)。
4. 在精靈中，輸入一個裝置名稱，然後選取 Hardware TOTP token (硬體 TOTP 權杖)，再選擇 Next (下一步)。
5. 輸入裝置序號。序號通常位於裝置的背面。
6. 在 MFA code 1 (MFA 代碼 1) 方塊中，輸入 MFA 裝置顯示的六位數字。您可能需要按下裝置正面的按鈕以顯示數字。

7. 當裝置在重新整理代碼時，等候 30 秒時間後，在 MFA code 2 (MFA 代碼 2) 方塊中輸入六位數字。您可能需要再次按下裝置正面的按鈕以顯示第二個數字。
8. 選擇 Add MFA (新增 MFA)。

> **Important**
>
> 產生驗證代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置卻變成不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

該設備已準備好與一起使用 AWS。如需與 AWS Management Console一起使用 MFA 的詳細資訊，請參閱 透過您的 IAM 登入頁面來使用 MFA 裝置。

啟用另一個 IAM 使用者的硬體 TOTP 權杖 (主控台)

您可以從 AWS Management Console啟用另一個 IAM 使用者的硬體 TOTP 權杖。

要啟用另一個 IAM 使用者的硬體 TOTP 權杖 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇使用者 。
3. 選擇要為其啟用 MFA 的使用者名稱。
4. 選擇 安全憑證 標籤。在 Multi-Factor Authentication (MFA) (多重要素驗證 (MFA)) 區段下方，選擇 Assign MFA device (指派 MFA 裝置)。
5. 在精靈中，輸入一個裝置名稱，然後選取 Hardware TOTP token (硬體 TOTP 權杖)，再選擇 Next (下一步)。
6. 輸入裝置序號。序號通常位於裝置的背面。
7. 在 MFA code 1 (MFA 代碼 1) 方塊中，輸入 MFA 裝置顯示的六位數字。您可能需要按下裝置正面的按鈕以顯示數字。

8. 當裝置在重新整理代碼時，等候 30 秒時間後，在 MFA code 2 (MFA 代碼 2) 方塊中輸入六位數字。您可能需要再次按下裝置正面的按鈕以顯示第二個數字。
9. 選擇 Add MFA (新增 MFA)。

> **Important**
>
> 產生驗證代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置卻變成不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。這種情況下，您可以重新同步裝置。

該設備已準備好與一起使用 AWS。如需與 AWS Management Console一起使用 MFA 的詳細資訊，請參閱 透過您的 IAM 登入頁面來使用 MFA 裝置。

取代實體 MFA 裝置

與您 AWS 帳戶根使用者 和 IAM 使用者一次最多可以指派給使用者的任何目前支援 MFA 類型組合的八個 MFA 裝置。如果使用者遺失裝置或基於任何原因需要汰換，您必須先停用舊裝置。然後，再為使用者加入新的裝置。

- 如需停用目前與使用者相關聯的裝置，請參閱 停用 MFA 裝置。
- 若要新增 IAM 使用者適用的替代硬體 TOTP 權杖，請遵循本主題稍早介紹的 啟用另一個 IAM 使用者的硬體 TOTP 權杖 (主控台) 程序。
- 若要新增的取代硬體 TOTP 權杖 AWS 帳戶根使用者，請遵循本主題啟用 AWS 帳戶根使用者 （控制台）的硬件 TOTP 令牌前面程序中的步驟。

啟用及管理虛擬 MFA 裝置 (AWS CLI 或 AWS API)

您可以使用 AWS CLI 命令或 AWS API 操作為 IAM 使用者啟用虛擬 MFA 裝置。您無法使用 AWS CLI、 AWS API、Windows PowerShell 工 AWS 帳戶根使用者 具或任何其他命令列工具來啟用 MFA 裝置。不過，您可以使用 AWS Management Console 為根使用者啟用 MFA 裝置。

當您從啟用 MFA 裝置時 AWS Management Console，主控台會為您執行多個步驟。如果您改為使用 Windows PowerShell 適用的 AWS CLI工具或 AWS API 建立虛擬裝置，則必須以正確的順序手動執行步驟。例如，如果要建立虛擬 MFA 裝置，則必須建立 IAM 物件，將程式碼擷取為字串或 QR 碼圖形，然後同步該裝置並將其與 IAM 使用者建立關聯。請參閱 New-IAMVirtualMFADevice 中的範例章節以了解更多詳細資訊。對於實體裝置，您可以跳過建立步驟，直接同步該裝置並將其與使用者建立關聯。

您可以將標籤連接至 IAM 資源 (包括虛擬 MFA 裝置)，以識別、整理和控制其存取權。只有在使用 AWS CLI 或 AWS API 時，才能標記虛擬 MFA 裝置。

使用 SDK 或 CLI 的 IAM 使用者可以透過呼叫 `EnableMFADevice` 啟用額外的 MFA 裝置，或者透過呼叫 `DeactivateMFADevice` 停用現有的 MFA 裝置。若要成功執行此動作，這些使用者必須先使用現有的 MFA 裝置呼叫 `GetSessionToken` 並提交 MFA 代碼。此呼叫會傳回暫時的安全憑證，然後可使用此憑證簽署需要 MFA 身分驗證的 API 作業。如需要求和回應的範例，請參閱 `GetSessionToken` - 不受信任環境中使用者的暫時憑證。

在 IAM 中建立虛擬裝置實體來代表虛擬 MFA 裝置

這些命令提供在以下許多命令中代替序號的裝置 ARN。

- AWS CLI: `aws iam create-virtual-mfa-device`
- AWS API：`CreateVirtualMFADevice`

啟用 MFA 裝置以搭配使用 AWS

這些指令會將裝置與使用者同步， AWS 並將其與使用者建立關聯。如果裝置是虛擬裝置，則將虛擬裝置的 ARN 做為序號使用。

> **Important**
>
> 產生驗證代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置卻變成不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。如果發生這種情況，可以使用下面介紹的命令重新同步裝置。

- AWS CLI: `aws iam enable-mfa-device`
- AWS API：`EnableMFADevice`

停用裝置

這些命令將取消裝置與使用者間的關聯並停用裝置。如果裝置是虛擬裝置，則將虛擬裝置的 ARN 做為序號使用。您也必須單獨刪除虛擬裝置實體。

- AWS CLI: `aws iam deactivate-mfa-device`
- AWS API：`DeactivateMFADevice`

列出虛擬 MFA 裝置實體

使用這些命令來列出虛擬 MFA 裝置的實體。

- AWS CLI: `aws iam list-virtual-mfa-devices`
- AWS API：`ListVirtualMFADevices`

標記虛擬 MFA 裝置

使用這些指令來標記虛擬 MFA 裝置。

- AWS CLI: `aws iam tag-mfa-device`
- AWS API：`TagMFADevice`

列出虛擬 MFA 裝置的標籤

使用這些命令列出連接至虛擬 MFA 裝置的標籤。

- AWS CLI: `aws iam list-mfa-device-tags`
- AWS API：`ListMFADeviceTags`

取消標記虛擬 MFA 裝置

使用這些命令移除連接至虛擬 MFA 裝置的標籤。

- AWS CLI: `aws iam untag-mfa-device`
- AWS API：`UntagMFADevice`

重新同步 MFA 裝置

如果設備正在生成不被接受的代碼，請使用這些命令 AWS。如果裝置是虛擬裝置，則將虛擬裝置的 ARN 做為序號使用。

- AWS CLI: `aws iam resync-mfa-device`
- AWS API：`ResyncMFADevice`

刪除 IAM 中的虛擬 MFA 裝置實體

在裝置與使用者取消關聯後，您可以刪除裝置實體。

- AWS CLI: `aws iam delete-virtual-mfa-device`
- AWS API：`DeleteVirtualMFADevice`

復原遺失或無法運作的虛擬 MFA 裝置

有時，主控虛擬 MFA 應用程式的使用者的裝置會發生遺失、遭到替換，或無法運作。當這種情況發生時，使用者無法自行復原。使用者必須聯絡管理員以停用裝置。如需更多詳細資訊，請參閱 MFA 裝置遺失或停止運作時怎麼辦？。

檢查 MFA 狀態

使用 IAM 主控台來檢查 AWS 帳戶根使用者 或 IAM 使用者是否已啟用有效的 MFA 裝置。

若要查看根使用者的 MFA 狀態

1. 使用根使用者登入資料登入，然後在開啟 IAM 主控台https://console.aws.amazon.com/iam/。AWS Management Console
2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 安全憑證 。
3. 檢查 多重要素驗證 (MFA) 下方，查看 MFA 是否已啟用或停用。如果 MFA 尚未啟動，則會顯示提醒符號
   ()

如果您想要啟用 MFA 帳戶，請參閱下列其中一項：

- 針對 AWS 帳戶根使用者 啟用虛擬 MFA 裝置 (主控台)
- 啟用 AWS 帳戶根使用者 (控制台) 的金鑰或安全金鑰
- 啟用 AWS 帳戶根使用者 （ 控制台 ）的硬件 TOTP 令牌

查看 IAM 使用者的 MFA 狀態

1. 在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 如有必要，請透過完成以下步驟將 MFA 欄新增到使用者表格：
   a. 在最右側的表格上方，選擇設定圖示
      ()。
   b. 在 Manage Columns (管理欄) 中，選取 MFA。
   c. (選用) 清除您您不想在使用者表格中顯示的任何欄標題的核取方塊。
   d. 選擇 Close (關閉) 返回使用者清單。
4. MFA 欄告訴您有關已啟用的 MFA 裝置。如果沒有給使用者作用中的 MFA 裝置，則主控台會顯示 None (無)。如果使用者已啟用 MFA 裝置，則 MFA 欄顯示啟用裝置的類型值為 Virtual (虛擬)、Security Key (安全金鑰)、Hardware (硬體) 或 SMS。

> **Note**
>
> AWS 結束了對啟用 SMS 多因素身份驗證（MFA）的支持。建議擁有使用 SMS 簡訊式 MFA 的 IAM 使用者的客戶改為使用下列任一種替代方式：虛擬 (軟體式) MFA 裝置、FIDO 安全性金鑰或硬體 MFA 裝置。您可以使用已分配的 SMS MFA 裝置來識別帳戶中的使用者。為此，請前往 IAM 主控台，從導覽窗格中選擇 Users (使用者)，然後在表的 MFA 列中尋找具有 SMS 的使用者。

5. 若要查看有關使用者的 MFA 裝置的其他資訊，請選擇要檢查其 MFA 狀態的使用者名稱。然後選擇 Security credentials (安全憑證) 索引標籤。
6. 如果沒有給使用者作用中的 MFA 裝置，則主控台會顯示 無 MFA 裝置。在多重要素驗證 (MFA) 區段中指派 MFA 裝置以改善 AWS 環境的安全性。如果使用者有已啟用的 MFA 裝置，則 Multi-factor authentication (MFA) (多重要素驗證 (MFA)) 區段會顯示有關這些裝置的詳細資訊：
   - 裝置名稱
   - 裝置類型
   - 裝置的識別碼，例如實體裝置的序號或虛擬裝置的 AWS ARN
   - 何時建立裝置

若要移除或重新同步裝置，請選擇裝置旁的選項按鈕，然後選取 Remove (移除) 或 Resync (重新同步)。

如需有關啟用 MFA 的詳細資訊，請參閱以下內容：

- 啟用虛擬多重要素驗證 (MFA) 裝置 (主控台)
- 啟用金鑰或安全金鑰 (主控台)
- 啟用硬體 TOTP 權杖 (主控台)

重新同步虛擬及硬體 MFA 裝置

您可以使用重新同步處理虛擬和硬體多重 AWS 要素驗證 (MFA) 裝置。如果您的裝置在您嘗試使用時未同步，登入嘗試失敗，且 IAM 提示您重新同步裝置。

> **Note**
>
> FIDO 安全性金鑰不會失去同步。若 FIDO 安全性金鑰遺失或損壞，您可以停用它。如需停用任何 MFA 裝置類型的說明，請參閱 為另一個 IAM 使用者停用 MFA 裝置 (主控台)。

AWS 身為管理員，您可以在 IAM 使用者的虛擬和硬體 MFA 裝置不同步時重新同步處理這些裝置。

如果您的 AWS 帳戶根使用者 MFA 裝置無法運作，您可以使用 IAM 主控台重新同步處理裝置，無論是否完成登入程序。如果您無法成功重新同步處理裝置，您可能需要為裝置取消關聯，再重新關聯此裝置。如需如何執行此作業的資訊，請參閱 停用 MFA 裝置 和 為中的使用者啟用 MFA 裝置 AWS。

主題

- 必要許可
- 重新同步虛擬及硬體 MFA 裝置 (IAM 主控台)
- 重新同步虛擬及硬體 MFA 裝置 (AWS CLI)
- 重新同步虛擬和硬體 MFA 裝置 (API)AWS

必要許可

若要為自己的 IAM 使用者重新同步虛擬或硬體 MFA 裝置，您必須擁有以下政策的許可。此政策不允許您建立或停用裝置。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToViewAndManageTheirOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:ListMFADevices",
```

```
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "BlockAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

重新同步虛擬及硬體 MFA 裝置 (IAM 主控台)

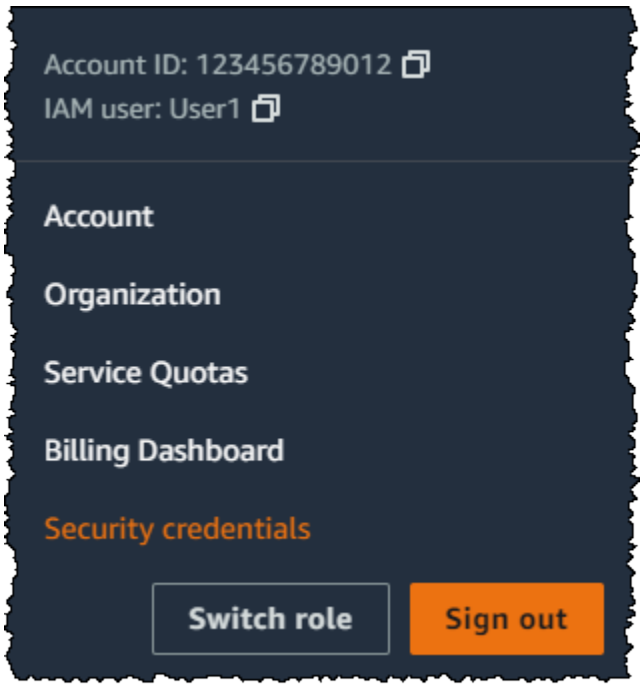
您可以使用 IAM 主控台來重新同步虛擬及硬體 MFA 裝置。

重新同步您 IAM 使用者的虛擬或硬體 MFA 裝置 (主控台)

1. 使用您的 AWS 帳戶 ID 或帳戶別名、IAM 使用者名稱和密碼登入 IAM 主控台。

   > **Note**
   >
   > 為方便起見，AWS 登入頁面會使用瀏覽器 Cookie 來記住您的 IAM 使用者名稱和帳戶資訊。如果您先前以不同的使用者身分登入，請選擇在頁面底部附近的 Sign in to a different account (登入不同的帳戶)，返回主要登入頁面。您可以在該處輸入帳 AWS 戶 ID 或帳戶別名，以重新導向至帳戶的 IAM 使用者登入頁面。

   若要取得您的 AWS 帳戶 ID，請聯絡您的系統管理員。
2. 在右上方的導覽列中，選擇您的使用者名稱，然後選擇 安全憑證 。



3. 在 AWS IAM 憑證 標籤的 多重要素驗證 (MFA) 區段選擇 MFA 裝置旁的選項按鈕，然後選擇 重新同步。
4. 將裝置接下來連續產生的兩個代碼輸入 MFA code 1 (MFA 代碼 1) 和 MFA code 2 (MFA 代碼 2)。然後選擇 Resync (重新同步)。

> **Important**
>
> 產生代碼之後立即提交您的請求。如果您產生代碼，然後等太久而無法提交請求、請求會出現運作，但裝置保持未同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。

重新同步另一個 IAM 使用者的虛擬或硬體 MFA 裝置 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)，然後選擇需要重新同步 MFA 裝置的使用者名稱。
3. 選擇 Security credentials (安全憑證) 索引標籤。在 多重要素驗證 (MFA) 區段中選擇 MFA 裝置旁的選項按鈕，然後選擇 重新同步。
4. 將裝置接下來連續產生的兩個代碼輸入 MFA code 1 (MFA 代碼 1) 和 MFA code 2 (MFA 代碼 2)。然後選擇 Resync (重新同步)。

> **Important**
>
> 產生代碼之後立即提交您的請求。如果您產生代碼，然後等太久而無法提交請求、請求會出現運作，但裝置保持未同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。

登入前重新同步根使用者 MFA (主控台)

1. 在 Amazon Web Services Sign In With Authentication Device (使用身分驗證裝置登入 Amazon Web Services) 頁面上，選擇 Having problems with your authentication device? (您的身分驗證裝置登有問題？)。Click here (請點選此處)。

   > **Note**
   >
   > 您可能會看到不同的文字，例如使用 MFA 登入和對您的身分驗證裝置進行疑難排解。不過，其功能是相同的。

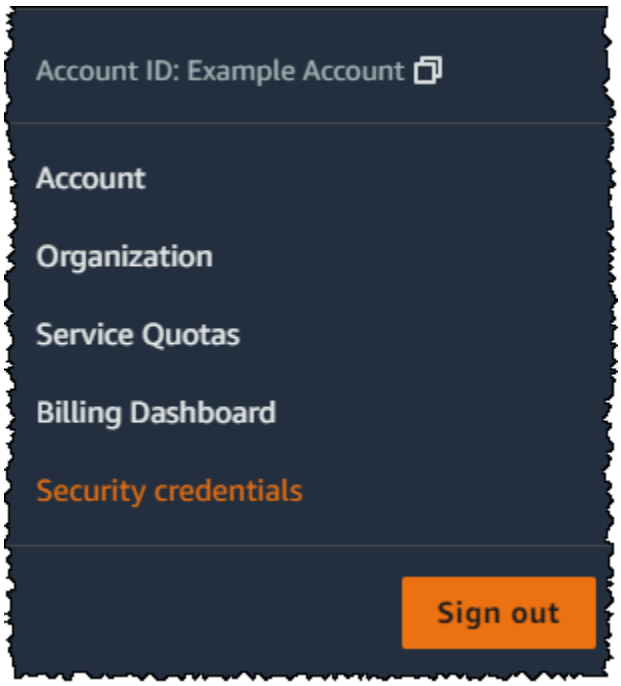
2. 在 Re-Sync With Our Servers (與我們的伺服器重新同步) 區段中，將裝置接下來連續產生的兩個代碼輸入 MFA code 1 (MFA 代碼 1) 和 MFA code 2 (MFA 代碼 2)。然後選擇 Re-sync authentication device (重新同步身分驗證裝置)。
3. 如果必要，再次輸入密碼，然後選擇登入。然後，使用您的 MFA 裝置完成登入。

登入後重新同步根使用者 MFA 裝置 (主控台)

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 IAM 主控台。在下一頁中，輸入您的密碼。

   > **Note**
   >
   > 根使用者無法登入 以 IAM 使用者身分登入 頁面。如果您看到 以 IAM 使用者身分登入 頁面，請選擇頁面底部附近的 使用根使用者電子郵件登入。如需以 root 使用者身分登入的說明，請參閱《使用指南》中的「以 root 使用者身分登入」AWS 登入 。 AWS Management Console

2. 在導覽列右側選擇您的帳戶名稱，然後選擇 安全憑證 。如有需要，選擇 Continue to Security Credentials (繼續至安全憑證)。



3. 展開頁面上的 Multi-factor authentication (MFA) (多重要素驗證 (MFA)) 區段。
4. 選擇裝置旁的選項按鈕，然後選取 Resync (重新同步)。
5. 在 Resync MFA device (重新同步 MFA 裝置) 對話方塊中，將裝置接下來連續產生的兩個代碼輸入 MFA code 1 (MFA 代碼 1) 和 MFA code 2 (MFA 代碼 2)。然後選擇 Resync (重新同步)。

> **Important**
>
> 產生代碼之後立即提交您的請求。如果在產生代碼後等待很長時間才提交請求，MFA 裝置會成功地與使用者建立關聯，但 MFA 裝置將不同步。會發生這種情況是因為定時式的一次性密碼 (TOTP) 在過了一小段時間後就會過期。

重新同步虛擬及硬體 MFA 裝置 (AWS CLI)

您可以從 AWS CLI重新同步虛擬及硬體 MFA 裝置。

重新同步 IAM 使用者的虛擬或硬體 MFA 裝置 (AWS CLI)

在命令提示字元中，發出 aws iam resync-mfa-device 命令：

- 虛擬 MFA 裝置：將裝置的 Amazon Resource Name (ARN) 指定為序號。

```
aws iam resync-mfa-device --user-name Richard --serial-number
 arn:aws:iam::123456789012:mfa/RichardsMFA --authentication-code1 123456 --
authentication-code2 987654
```

- 硬體 MFA 裝置：將硬體裝置的序號指定為序號。格式為廠商特定。例如，您可以從 Amazon 購買 gemalto 權杖。其序號通常是四個字母，後面接著四個數字。

```
aws iam resync-mfa-device --user-name Richard --serial-number ABCD12345678 --
authentication-code1 123456 --authentication-code2 987654
```

> **Important**
>
> 產生代碼之後立即提交您的請求。如果您產生代碼，然後因等太久而無法提交請求、請求會因代碼在不久後過期而失敗。

重新同步虛擬和硬體 MFA 裝置 (API)AWS

IAM 有一個執行同步化的 API 呼叫。在這種情況下，建議您提供您的虛擬及硬體 MFA 裝置使用者許可，讓其存取此 API 呼叫。然後根據該 API 呼叫建置工具，讓使用者在需要時能隨時重新同步他們的裝置。

為 IAM 使用者重新同步處理虛擬或硬體 MFA 裝置 (API)AWS

- 傳送 ResyncMFADevice 請求。

停用 MFA 裝置

如果您無法將多重要素驗證 (MFA) 裝置作為 IAM 使用者登入，請聯絡您的管理員尋求協助。

身為管理員，您可以停用另一個 IAM 使用者的裝置。這可讓使用者在不使用 MFA 的情況下登入。在更換 MFA 裝置或裝置暫時不可用時，您可以將此做為臨時解決方案。但是，我們建議您盡快為使用者啟用新的裝置。若要了解如何啟用新的 MFA 裝置，請參閱 the section called “啟用 MFA 裝置”。

> **Note**
>
> 如果您使用 API 或 AWS CLI 從中刪除使用者 AWS 帳戶，則必須停用或刪除使用者的 MFA 裝置。您將此變更做為移除使用者的過程的一部分。如需有關刪除使用者的詳細資訊，請參閱「管理 IAM 使用者」。

主題

- 停用 MFA 裝置 (主控台)
- 停用 MFA 裝置 (AWS CLI)
- 停用 MFA 裝置 (API)AWS

停用 MFA 裝置 (主控台)

為另一個 IAM 使用者停用 MFA 裝置 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 若要為使用者停用 MFA 裝置，請選擇要移除其 MFA 的使用者的名稱。
4. 選擇 Security credentials (安全憑證) 索引標籤。
5. 在 多重要素驗證 (MFA) 下方選擇 MFA 裝置旁的選項按鈕、選擇 移除，然後選擇 移除。

   裝置即會從中移除 AWS。在重新啟用並與使用 AWS 者或相關聯之前，它無法用於登入或 AWS 帳戶根使用者驗證要求。

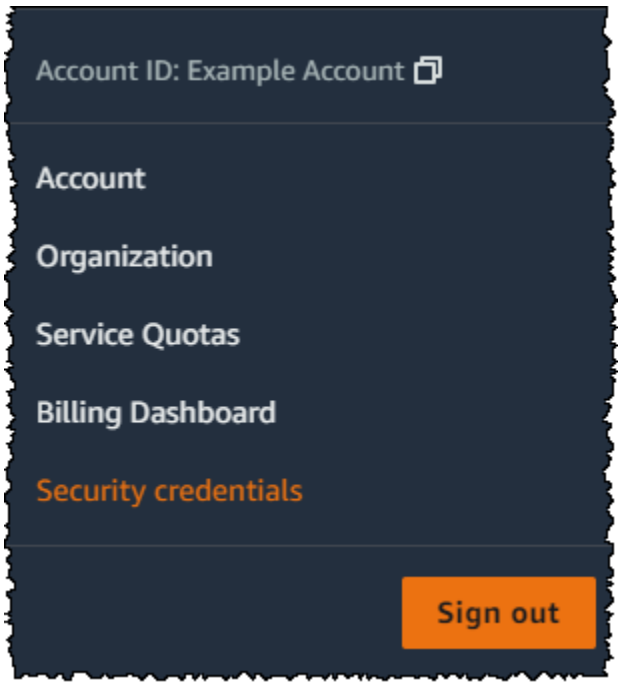
若要停用 AWS 帳戶根使用者 (主控台) 的 MFA 裝置

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 IAM 主控台。在下一頁中，輸入您的密碼。

   > **Note**
   >
   > 根使用者無法登入 以 IAM 使用者身分登入 頁面。如果您看到 以 IAM 使用者身分登入 頁面，請選擇頁面底部附近的 使用根使用者電子郵件登入。如需以 root 使用者身分登

入的說明，請參閱《使用指南》中的「以 root 使用者身分登入」AWS 登入 。 AWS Management Console

2. 在導覽列右側選擇您的帳戶名稱，然後選擇 安全憑證 。如有需要，選擇 Continue to Security Credentials (繼續至安全憑證)。



3. 在 Multi-factor authentication (MFA) (多重要素驗證 (MFA)) 區段中，選擇您想要停用的 MFA 裝置旁的選項按鈕，然後選擇 Remove (移除)。
4. 選擇移除。

   已為 AWS 帳戶停用 MFA 裝置。查看與您相關聯的電子郵件，以取 AWS 帳戶 得來自 Amazon Web Services 的確認訊息。該電子郵件會通知您，Amazon Web Services 多重要素驗證 (MFA) 已停用。訊息來自於 `@amazon.com` 或 `@aws.amazon.com`。

停用 MFA 裝置 (AWS CLI)

為 IAM 使用者停用 MFA 裝置 (AWS CLI)

- 執行此命令：`aws iam deactivate-mfa-device`

停用 MFA 裝置 (API)AWS

若要停用 IAM 使用者的 MFA 裝置 (AWS API)

- 呼叫此操作：DeactivateMFADevice

MFA 裝置遺失或停止運作時怎麼辦？

如果您的虛擬 MFA 設備或硬件 TOTP 令牌似乎運行正常，但您無法使用它來訪問您的 AWS 資源，則它可能與之不同步。 AWS如需同步虛擬 MFA 裝置或硬體 MFA 裝置的資訊，請參閱 重新同步虛擬及硬體 MFA 裝置。FIDO 安全性金鑰不會失去同步。

如果您的 AWS 帳戶根使用者 多因素身份驗證（MFA）設備丟失，損壞或無法正常工作，則可以恢復對帳戶的訪問權限。IAM 使用者必須聯絡管理員以停用裝置。

> **Important**
>
> 我們建議您為您的 IAM 使用者啟用多台 MFA 裝置，以確保在一台 MFA 裝置遺失或無法存取時繼續存取您的帳戶。您可以以目前受支援 MFA 類型的任意組合為您的 AWS 帳戶 根使用者和 IAM 使用者註冊最多八台 MFA 裝置。

復原根使用者 MFA 裝置

如果您的 AWS 帳戶根使用者 多因素身份驗證（MFA）設備丟失，損壞或無法正常工作，則可以使用其他註冊到同一設備的 MFA 設備登錄。 AWS 帳戶根使用者如果根使用者僅啟用了一台 MFA 裝置，您可以採用身分驗證的替代方法。這表示若無法藉由 MFA 裝置登入，您可以採用已用您的帳戶註冊的電子郵件和主要聯絡人電話號碼進行身分驗證，以藉此方式登入。

在使用其他身分驗證方法以根使用者身分登入之前，您必須有權存取與您的帳戶相關聯的電子郵件和主要聯絡人電話號碼。若需要更新主要聯絡人電話號碼，您可以使用管理員存取權 (而非根使用者) 以 IAM 使用者的身分登入。如需了解有關更新帳戶聯絡人資訊的更多指示，請參閱 AWS Billing 使用者指南中的編輯聯絡人資訊。如果無權存取電子郵件和主要聯絡人電話號碼，則您必須聯絡 AWS Support。

> **Important**
>
> 建議您保持電子郵件地址及聯絡人電話號碼與您的根使用者的關聯處於最新狀態，以成功地復原帳戶。如需詳細資訊，請參閱《AWS Account Management 參考指南》中的更新您 AWS 帳戶的主要連絡人。

使用驗證的替代因素登入 AWS 帳戶根使用者

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。
2. 在需要其他驗證頁面上，選取要用來進行驗證的 MFA 方法，然後選擇下一步。

   > **Note**
   >
   > 您可能會看到替代文字，例如使用 MFA 登入、對您的驗證裝置進行故障診斷，或對 MFA 進行故障診斷，但功能是相同的。您不能使用其他驗證方法來驗證您的帳戶電子郵件地址和主要聯絡人電話號碼，請聯絡 AWS Support 停用您的 MFA 裝置。

3. 視您使用的 MFA 類型而定，您會看到不同的頁面，但 MFA 疑難排解選項的功能相同。在需要其他驗證頁面或多重要素驗證頁面上，選擇 MFA 疑難排解。
4. 如果必要，再次輸入密碼，然後選擇 Sign in (登入)。
5. 在使用其他驗證要素登入區段中的驗證裝置疑難排解頁面上，選擇使用其他要素登入。
6. 在使用其他驗證要素登入頁面上，透過驗證電子郵件地址來驗證您的帳戶，然後選擇傳送驗證電子郵件。
7. 請查看與您相關聯的電子郵件，以 AWS 帳戶 取得來自 Amazon Web Services (recover-mfa-no-reply@verify .signin.aws) 的訊息。請遵循電子郵件中的指示進行。

   如果您沒看到帳戶中有電子郵件，請檢查您的垃圾郵件資料夾，或返回瀏覽器，然後選擇 Resend the email (重新傳送電子郵件)。
8. 驗證電子郵件地址後，您可以繼續驗證您的帳戶。若要驗證主要聯絡人電話號碼，請選擇 Call me now (立即呼叫我)。
9. 接聽來電， AWS 並在出現提示時，在電話鍵盤上輸入 AWS 網站上的 6 位數字號碼。

   如果沒有收到來電 AWS，請選擇 [登入] 再次登入主機，然後重新開始。或者，請參閱遺失或無法使用的多重要素驗證 (MFA) 裝置來聯絡支援中心以尋求協助。

10. 驗證您的電話號碼之後，您可以選擇 Sign in to the console (登入主控台) 登入您的帳戶。
11. 下一步取決於您使用的 MFA 類型：

    - 針對虛擬 MFA 裝置，請從您的裝置移除帳戶。然後，請前往 AWS Security Credentials (AWS 安全憑證) 頁面，並刪除舊 MFA 虛擬裝置實體，再建立新的實體。
    - 針對 FIDO 安全性金鑰，請前往 AWS Security Credentials (安全憑證) 頁面並停用舊 FIDO 安全性金鑰，然後再啟用新金鑰。
    - 針對硬體 TOTP 裝置，請聯絡第三方供應商，請其協助修復或更換裝置。您可以繼續使用其他身分驗證方法登入，直到收到新的裝置為止。擁有新的硬體 MFA 裝置之後，請前往 AWS Security Credentials (AWS 安全憑證) 頁面，並刪除舊 MFA 硬體裝置實體，再建立新的實體。

> **Note**
>
> 您無須將遺失或遭竊的 MFA 裝置更換成相同類型的裝置。例如，若您損壞了 FIDO 安全金鑰並訂購了新的，您可以使用虛擬 MFA 或硬體 TOTP 權杖，直到您收到新的 FIDO 安全金鑰為止。

> **Important**
>
> 如果您的 MFA 裝置遺失或遭竊，請在使用其他身分驗證方法登入並建立替代 MFA 裝置後變更根使用者密碼，以防攻擊者竊取驗證裝置並可能擁有您目前的密碼。如需詳細資訊，請參閱《AWS Account Management 參考指南》中的變更 AWS 帳戶根使用者的密碼。

復原 IAM 使用者 MFA 裝置

如果您是 IAM 使用者，且您的裝置遺失或停止運作，則無法自行復原。您必須聯絡管理員以停用裝置。然後，您就可以啟用新裝置。

以 IAM 使用者身分取得 MFA 裝置的相關協助

1. 請聯絡 AWS 系統管理員或其他提供 IAM 使用者使用者名稱和密碼的人員。管理員必須依 停用 MFA 裝置 中所述停用 MFA 裝置，如此您才可以登入。
2. 下一步取決於您使用的 MFA 類型：

- 針對虛擬 MFA 裝置，請從您的裝置移除帳戶。然後依 啟用虛擬多重要素驗證 (MFA) 裝置 (主控台) 所述啟用虛擬裝置。
- 針對 FIDO 安全金鑰，請聯絡第三方供應商，請其協助更換裝置。當您收到新的 FIDO 安全性金鑰時，請遵循 啟用金鑰或安全金鑰 (主控台) 中所述的程序來啟用它。
- 針對硬體 TOTP 裝置，請聯絡第三方供應商，請其協助修復或更換裝置。在擁有新的實體 MFA 裝置後，依 啟用硬體 TOTP 權杖 (主控台) 所述啟用裝置。

> **Note**
>
> 您無須將遺失或遭竊的 MFA 裝置更換成相同類型的裝置。您最多可以有八台任意組合的 MFA 裝置。例如，若您損壞了 FIDO 安全金鑰並訂購了新的，您可以使用虛擬 MFA 或硬體 TOTP 權杖，直到您收到新的 FIDO 安全金鑰為止。

3. 若您的 MFA 裝置遺失或遭竊，請同時變更您的 密碼，以防攻擊者竊取身分驗證裝置，同時還可能擁有您目前的密碼。如需詳細資訊，請參閱 管理 IAM 使用者密碼

設定受 MFA 保護的 API 存取

利用 IAM 政策，可以指定允許使用者呼叫的 API 操作。在一些情況下，您可能希望先要求使用者透過 AWS 多重要素驗證 (MFA) 進行身分驗證，然後才允許使用者執行尤其敏感的動作，藉以提高安全性。

例如，您可能擁有允許使用者執行 Amazon EC2 `RunInstances`、`DescribeInstances` 與 `StopInstances` 動作的政策。但是您可能想要限制破壞性的動作，例如，`TerminateInstances`並確保使用者只有在使用 AWS MFA 裝置進行驗證時才能執行該動作。

主題

- 概觀
- 使用案例：跨帳戶委派的 MFA 防護
- 使用案例：目前帳戶中 API 操作存取的 MFA 防護
- 使用案例：擁有以資源為基礎的政策之資源的 MFA 防護

概觀

新增 MFA 防護到 API 操作將包括以下任務：

1. 系統管理員會為每個需要發出需要 AWS MFA 驗證的 API 要求的使用者設定 MFA 裝置。此程序在為中的使用者啟用 MFA 裝置 AWS 中說明。
2. 系統管理員會為使用者建立原則，其中包含檢查使用者是否已透過 AWS MFA 裝置驗證的`Condition`元素。
3. 使用者呼叫其中一個支援 MFA 參數的 AWS STS API 作業 GetSessionToken，AssumeRole或視 MFA 保護的案例而定，如稍後所述。作為呼叫的一部分，使用者包含與其關聯的裝置的裝置識別碼。使用者也包含裝置產生之以時間為基礎的單次密碼 (TOTP)。在任一情況下，使用者都會取回稍後用來向 AWS發出其他請求的臨時安全憑證。

> **Note**
>
> 只有在服務支援臨時安全憑證時，才可使用該服務的 API 操作的 MFA 防護。如需這些服務的清單，請參閱使用暫時安全憑證存取 AWS。

如果授權失敗，則 AWS 返回拒絕訪問錯誤消息（就像對任何未經授權的訪問一樣）。使用 MFA 保護的 API 政策時，如果使用者嘗試在沒有有效 MFA 驗 AWS 證的情況下呼叫 API 作業，則拒絕存取政策中指定的 API 作業。如果 API 操作請求的時間戳記在政策中指定的允許範圍之外，也會拒絕該操作。使用者必須使用 MFA 代碼和裝置序號來請求新的臨時安全性憑證，並透過 MFA 重新進行身分驗證。

含有 MFA 條件的 IAM 政策

含有 MFA 條件的政策可連接到以下項目：

- IAM 使用者或群組
- 資源，例如 Amazon S3 儲存貯體、Amazon SQS 佇列或者 Amazon SNS 主題
- 可由使用者擔任的 IAM 角色的信任政策

可使用政策中的 MFA 條件來檢查以下屬性：

- 存在 - 如果只是驗證使用者是否已使用 MFA 進行身分驗證，請檢查 `aws:MultiFactorAuthPresent` 金鑰在 `Bool` 條件中是否為 `True`。只有在使用者使用短期憑證進行驗證時，索引鍵才會存在。長期憑證，例如存取金鑰，則不包括此鍵。
- 持續時間 - 如果您只希望在 MFA 身分驗證後的指定時間內授予存取權限，請使用數值條件類型將 `aws:MultiFactorAuthAge` 索引鍵的有效期與某個值 (如 3600 秒) 進行比較。請注意，如果未使用 MFA，則 `aws:MultiFactorAuthAge` 索引鍵不會顯示。

以下範例顯示 IAM 角色的信任政策，該政策包含一個 MFA 條件，用於測試是否存在 MFA 身分驗證。使用此原則時，來自`Principal`元素中 AWS 帳戶 指定的 (以有效 AWS 帳戶 ID 取代`ACCOUNT-B-ID`) 的使用者可以承擔此原則所附加的角色。不過，如果使用者使用 MFA 進行身分驗證，這類的使用者只能擔任該角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

如需有關 MFA 條件類型的詳細資訊，請參閱AWS 全域條件內容索引鍵、數位條件運算子 與 用於檢查條件索引鍵是否存在的條件運算子 。

在 GetSessionToken 和之間選擇 AssumeRole

AWS STS 提供兩種 API 作業，可讓使用者傳遞 MFA 資訊：`GetSessionToken`和`AssumeRole`. 使用者呼叫以獲取臨時安全憑證的 API 操作取決於適用於以下哪個案例。

針對以下案例使用 **`GetSessionToken`**：

- 呼叫存取資源的 API 作業與 AWS 帳戶 提出請求的 IAM 使用者相同。請注意，`GetSessionToken`請求中的臨時登入資料只有在登入資料請求中包含 MFA 資訊時，才能存取 IAM 和 AWS STS API 操作。由於 `GetSessionToken` 傳回的臨時憑證包含 MFA 資訊，因此您可以檢查由該憑證發出的個別 API 操作中的 MFA。
- 存取受到以資源為基礎且包含 MFA 條件的政策所保護的資源。

`GetSessionToken` 操作的目的是使用 MFA 驗證使用者的身分。您不能使用政策來控制驗證操作。

針對以下案例使用 **`AssumeRole`**：

- 呼叫存取相同的或不同的 AWS 帳戶中的資源的 API 操作。API 呼叫可以包含任何 IAM 或 AWS STS API。請注意，要保護存取，您可以在使用者擔任角色時強制執行 MFA。由 `AssumeRole` 傳回的臨時憑證未將 MFA 資訊包含在上下文中，因此您無法檢查 MFA 的單一 API 操作。這就是您必須使用 `GetSessionToken` 限制對受到以資源為基礎之政策保護的資源的存取的原因。

本文件稍後將提供有關如何實施這些使用案例的詳細資訊。

有關受 MFA 保護的 API 存取的要點

瞭解 API 操作的 MFA 防護有下列數個層面非常重要：

- MFA 防護僅透過使用臨時安全性憑證供，而該憑證必須使用 AssumeRole 或 GetSessionToken 來取得。
- 您無法搭配認證使用 MFA 保護的 API 存取權。 AWS 帳戶根使用者
- 無法搭配 U2F 安全性金鑰使用受 MFA 保護的 API 存取。
- 聯合使用者無法指派 MFA 裝置來搭配 AWS 服務使用，因此他們無法存取由 MFA 控制的 AWS 資源。(請查看下一要點。)
- 其他傳回臨時登入資料的 AWS STS API 作業不支援 MFA。對於AssumeRoleWithWebIdentity和AssumeRoleWithSAML，使用者會由外部提供者驗證，而且 AWS 無法判斷該提供者是否需要 MFA。對於 GetFederationToken，MFA 不一定要與特定使用者相關聯。
- 同樣地，長期憑證 (IAM 使用者存取金鑰和根使用者存取金鑰) 無法用於受 MFA 保護的 API 存取，因為此類憑證不會過期。
- 也可以在沒有 MFA 資訊的情況下呼叫 AssumeRole 與 GetSessionToken。在此情況下，呼叫者將取回臨時安全性憑證，但這些臨時憑證的工作階段資訊不會顯示使用 MFA 進行身分驗證的使用者。
- 若要建立 API 操作的 MFA 防護，可將 MFA 條件加入到政策。政策必須包含 aws:MultiFactorAuthPresent 條件索引鍵，才能強制使用 MFA。對於跨帳戶委派，該角色的信任政策必須包含條件索引鍵。
- 當您允許其他人存 AWS 帳戶 取您帳戶中的資源時，資源的安全性取決於受信任帳戶 (另一個帳戶，而不是您的帳戶) 的設定。即使您要求多重要素驗證，也是如此。有權建立虛擬 MFA 裝置的可信帳戶中的任何身分都可以建構 MFA 宣告，以滿足角色信任政策的該部分。在允許其他帳戶的成員存取需要多重要素驗證的 AWS 資源之前，您應該確定受信任帳戶的擁有者遵循安全性最佳做法。例如，信任的帳戶應該限制存取敏感 API 操作 (例如 MFA 裝置管理 API 操作) 至特定的信任身分。
- 如果政策包含 MFA 條件，則在以下情況下將拒絕請求：使用者未進行 MFA 身分驗證或使用者提供了無效的 MFA 裝置識別碼或無效的 TOTP。

使用案例：跨帳戶委派的 MFA 防護

在此案例中，您想要將存取權委派給其他帳戶中的 IAM 使用者，但前提是使用者必須透過 AWS MFA 裝置進行驗證。(如需建立跨帳戶委派的詳細資訊，請參閱 角色術語和概念。

假設您有一個帳戶 A (擁有要存取的資源的信任帳戶)，其 IAM 使用者 Anaya 擁有管理員許可。她希望對帳戶 B (可信任的帳戶) 中的使用者 Richard 授予存取權，但希望確保 Richard 在擔任該角色之前已使用 MFA 進行身分驗證。

1. 在信任帳戶 A 中，Anaya 會建立名為的 IAM 角色，`CrossAccountRole`並將角色信任政策中的主體設定為帳戶 B 的帳戶 ID，信任政策會授與動作的 AWS STS `AssumeRole`權限。Anaya 也將 MFA 條件加入到信任政策中，如以下範例中所示。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

2. Anaya 在該角色新增一個許可政策，以指定允許該角色執行的操作。具有 MFA 防護的角色許可政策與任何其他角色許可政策沒有差別。以下範例顯示 Anaya 新增到角色的政策；它允許假定的使用者在帳戶 A 中的 Books 資料表上執行任何 Amazon DynamoDB 動作。此政策也允許 `dynamodb:ListTables` 動作，而這是在主控台中執行動作的必要項目。

> **Note**
>
> 該許可政策不包含 MFA 條件。了解 MFA 身分驗證僅用於確定使用者是否可以擔任此角色這點非常重要。在使用者擔任此角色後，將不會進行進一步的 MFA 檢查。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TableActions",
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:*:ACCOUNT-A-ID:table/Books"
        },
        {
            "Sid": "ListTables",
```

```
            "Effect": "Allow",
            "Action": "dynamodb:ListTables",
            "Resource": "*"
        }
    ]
}
```

3. 在受信任的帳戶 B 中，管理員確保 IAM 使用者 Richard 已設定 AWS MFA 裝置，並且知道裝置的識別碼。如果是硬體 MFA 裝置，則裝置 ID 是序號，或如果是虛擬 MFA 裝置，裝置是 ARN。
4. 在帳戶 B 中，管理員將以下政策連接到使用者 Richard (或該使用者所在的群組)，該政策允許使用者呼叫 `AssumeRole` 動作。資源被設定到 Anaya 在第 1 步中所建立的角色 ARN。注意，該政策不包含 MFA 條件。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sts:AssumeRole"],
    "Resource": ["arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole"]
  }]
}
```

5. 在帳戶 B 中，Richard (或 Richard 正在執行的應用程式) 呼叫 `AssumeRole`。API 呼叫包含要擔任角色的 ARN (`arn:aws:iam::ACCOUNT-A-ID:role/CrossAccountRole`)、MFA 裝置的 ID 和 Richard 從其裝置中取得的目前 TOTP。

   當理查德打電話時`AssumeRole`， AWS 確定他是否具有有效的憑據，包括 MFA 的要求。如果是這種情況，Richard 將成功取得角色，並且可在使用角色的臨時憑證時對帳戶 A 中名為 Books 的表格執行任何 DynamoDB 動作。

   有關呼叫 `AssumeRole` 的程式之範例，請參閱 AssumeRole 使用 MFA 驗證進行呼叫。

使用案例：目前帳戶中 API 操作存取的 MFA 防護

在這個案例中，您應該確保您中的使用者只有在使用 AWS MFA 裝置驗證使用者時才 AWS 帳戶 能存取敏感的 API 作業。

假設您擁有帳戶 A，其中包含一組需要使用 EC2 執行個體的開發人員。普通開發人員可以使用執行個體，但他們未獲得 `ec2:StopInstances` 或 `ec2:TerminateInstances` 操作的許可。您希望

僅允許幾個可信任的使用者執行這些「破壞性」特權操作，因此您將 MFA 防護加入到允許這些敏感 Amazon EC2 動作的政策中。

在此使用案例中，使用者 Sofía 是可信任的使用者之一。使用者 Anaya 是帳戶 A 中的管理員。

1. 安娜亞確保 Sofía 配置了 AWS MFA 設備，並且索菲亞知道設備的 ID。如果是硬體 MFA 裝置，則裝置 ID 是序號，或如果是虛擬 MFA 裝置，裝置是 ARN。
2. Anaya 建立一個名為 `EC2-Admins` 的群組並將使用者 Sofía 加入到該群組中。
3. Anaya 將以下政策連接到 `EC2-Admins` 群組。此政策授予使用者呼叫 Amazon EC2 `StopInstances` 與 `TerminateInstances` 動作的許可，但前提是該使用者已使用 MFA 進行身分驗證。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Resource": ["*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

4.

> **Note**
>
> 要讓此政策生效，使用者必須先登出，然後再次登入。

   如果使用者 Sofía 需要停止或終止 Amazon EC2 執行個體，她 (或執行中的應用程式) 會呼叫 `GetSessionToken`。此 API 操作傳遞 MFA 裝置的 ID，以及 Sofía 從其裝置取得的目前 TOTP。
5. 使用者 Sofía (或 Sofía 正在使用的應用程式) 使用由 `GetSessionToken` 提供的臨時憑證來呼叫 Amazon EC2 `StopInstances` 或者 `TerminateInstances` 動作。

   有關呼叫 `GetSessionToken` 的程式之範例，請參閱本文件後述的 GetSessionToken 使用 MFA 驗證進行呼叫。

使用案例：擁有以資源為基礎的政策之資源的 MFA 防護

在此案例中，您是 S3 儲存貯體、SQS 佇列或 SNS 主題的擁有者。您想要確保來自任 AWS 帳戶 何存取資源的使用者都經過 AWS MFA 裝置驗證。

此使用案例介紹了一種提供跨帳戶 MFA 防護的方法，無需使用者先擔任角色。在此情況下，若符合三個條件，使用者便可存取資源。使用者必須通過 MFA 的身分驗證，能夠從 `GetSessionToken` 取得臨時安全性憑證，並且在資源政策所信任的帳戶中。

假設您在帳戶 A 中並建立一個 S3 儲存貯體。您想要將此值區的存取權授予位於數個不同的使用者 AWS 帳戶，但前提是這些使用者必須透過 MFA 驗證。

在此方案中，使用者 Anaya 是帳戶 A 中的管理員。使用者 Nikhil 是帳戶 C 中的 IAM 使用者。

1. 在帳戶 A 中，Anaya 建立一個名為 `Account-A-bucket` 的儲存貯體。
2. Anaya 將儲存貯體政策加入到儲存貯體。該政策允許帳戶 A、帳戶 B 或帳戶 C 中的所有使用者執行儲存貯體中的 Amazon S3 `PutObject` 和 `DeleteObject` 動作。該政策包含 MFA 條件。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": [
      "ACCOUNT-A-ID",
      "ACCOUNT-B-ID",
      "ACCOUNT-C-ID"
    ]},
    "Action": [
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::ACCOUNT-A-BUCKET-NAME/*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

> **Note**
>
> Amazon S3 (僅) 針對根帳戶存取提供「MFA 刪除」功能。在您設定儲存貯體的版本控制狀態時，可啟用 Amazon S3 MFA Delete 功能。Amazon S3 MFA Delete 功能不適用於 IAM 使用者，在管理時獨立於 MFA 防護的 API 存取。即使 IAM 使用者有刪除儲存貯體的許可，

但在啟用 Amazon S3 MFA Delete 功能時，也無法執行刪除。如需有關 Amazon S3 MFA Delete 的詳細資訊，請參閱 MFA Delete。

3. 在帳戶 C 中，管理員確定使用者 Nikhil 以 AWS MFA 裝置設定，而且知道裝置的 ID。如果是硬體 MFA 裝置，則裝置 ID 是序號，或如果是虛擬 MFA 裝置，裝置是 ARN。
4. 在帳戶 C 中，Nikhil (或該使用者正在執行的應用程式) 呼叫 `GetSessionToken`。此呼叫包括 MFA 裝置的 ID 或 ARN 以及 Nikhil 從其裝置中取得的目前 TOTP。
5. Nikhil (或 Nikhil 正在使用的應用程式) 使用由 `GetSessionToken` 傳回的暫時憑證來呼叫 Amazon S3 `PutObject` 動作，上傳檔案到 `Account-A-bucket`。

   有關呼叫 `GetSessionToken` 的程式之範例，請參閱本文件後述的 GetSessionToken 使用 MFA 驗證進行呼叫。

> **Note**
>
> `AssumeRole` 傳回的暫時憑證在此案例中則不會運作。雖然使用者可以提供 MFA 資訊以取得角色，`AssumeRole` 傳回的暫時憑證不會包含 MFA 資訊。有了這項資訊，才能符合政策中的 MFA 條件。

範本程式碼：使用多重要素驗證請求憑證

以下範例顯示如何呼叫 `GetSessionToken` 和 `AssumeRole` 操作，並傳遞 MFA 驗證參數。呼叫 `GetSessionToken` 不需要許可，但您必須擁有一個可讓您呼叫 `AssumeRole` 的政策。傳回的憑證隨後用於列出帳戶中的所有 S3 儲存貯體。

GetSessionToken 使用 MFA 驗證進行呼叫

以下範例說明如何呼叫 `GetSessionToken` 並傳遞 MFA 身分驗證資訊。然後，由 `GetSessionToken` 操作傳回的臨時安全憑證用於列出帳戶中的所有 S3 儲存貯體。

連接到執行此程式碼的使用者的政策 (或群組中的使用者)，會針對傳回的臨時憑證提供許可。對於此範本程式碼，政策必須授予使用者許可來請求 Amazon S3 `ListBuckets` 操作。

下列程式碼範例會示範如何使用`GetSessionToken`。

CLI

AWS CLI

為 IAM 身分取得一組短期憑證

下列 `get-session-token` 命令會為進行呼叫的 IAM 身分擷取一組短期憑證。產生的憑證可用於政策要求多重要素驗證 (MFA) 的請求。憑證會在產生後的 15 分鐘過期。

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

輸出：

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的請求臨時安全憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetSessionToken中的。

PowerShell

適用的工具 PowerShell

範例 1：傳回包含一段時間內有效之暫時認證的**Amazon.RuntimeAWSCredentials**執行個體。用來請求臨時登入資料的認證是從目前的 shell 預設值推斷出來的。若要指定其他認證，請使用-ProfileName 或-AccessKey /-SecretKey 參數。

```
Get-STSSessionToken
```

輸出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

範例 2：傳回包含有效期為一小時之臨時認證的**Amazon.RuntimeAWSCredentials**執行個體。用來提出要求的認證是從指定的設定檔取得。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

輸出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

範例 3：傳回包含有效期為一小時之臨時認證的**Amazon.RuntimeAWSCredentials**執行個體，該執行個體會使用與帳戶相關聯的 MFA 裝置識別號碼 (其認證在設定檔「myprofilename」中指定) 以及裝置提供的值。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

輸出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetSessionToken式參考中的。

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。

透過傳遞 MFA 字符取得工作階段字符，並使用它列出該帳戶的 Amazon S3 儲存貯體。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
```

```
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

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

AssumeRole 使用 MFA 驗證進行呼叫

以下範例說明如何呼叫 AssumeRole 並傳遞 MFA 身分驗證資訊。然後，由 AssumeRole 傳回的臨時安全憑證用於列出帳戶中的所有 Amazon S3 儲存貯體。

如需有關此案例的詳細資訊，請參閱 使用案例：跨帳戶委派的 MFA 防護。

下列程式碼範例會示範如何使用AssumeRole。

.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;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
```

```
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials);
```

```
            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AssumeRole中的。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}
```

```
##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
##############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        ech o"Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AssumeRole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。

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
```

```
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考[AssumeRole](#)中的。

CLI

AWS CLI

擔任角色

下列 `assume-role` 命令會為 IAM 角色 `s3-access-example` 擷取一組短期憑證。

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
```

```
    --role-session-name s3-access-example
```

輸出：

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

該命令的輸出包含存取金鑰、私密金鑰以及可用來向 AWS進行驗證的工作階段字符。

對於 AWS CLI 使用，您可以設置與角色關聯的具名配置文件。當您使用設定檔時， AWS CLI 會為您呼叫 assume-role 並管理認證。如需詳細資訊，請參閱 CLI 使用者指南中的 AWS CLI 中的AWS 使用 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AssumeRole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.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version": "2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
assume (for example, rn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
                """;

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

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
```

```
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
 exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考AssumeRole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。

建立用戶端。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

擔任 IAM 角色。

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";
```

```
import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AssumeRole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 設定和執行程式碼範例儲存庫。

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
```

```
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AssumeRole中的。

PowerShell

適用的工具 PowerShell

示例 1：返回一組臨時憑據（訪問密鑰，秘密密鑰和會話令牌），可用於一個小時來訪問請求用戶通常無法訪問的 AWS 資源。傳回的認證具有所承擔之角色的存取原則所允許的權限，以及所提供的原則 (您無法使用提供的原則來授與超過假定角色之存取原則所定義的權限)。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

範例 2：傳回一組臨時登入資料 (有效期為一小時)，這些登入資料具有與所假定角色之存取原則中所定義的相同權限。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

範例 3：傳回一組臨時認證，提供序號與用來執行指令程式之使用者認證相關聯的 MFA 產生的權杖。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

範例 4：傳回一組已擔任客戶帳戶中定義角色的臨時登入資料。對於第三方可以承擔的每個角色，客戶帳戶必須使用識別碼來建立角色，識別碼必須在每次假設角色時傳入-ExternalId 參數。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程AssumeRole式參考中的。

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。

擔任需要 MFA 字符的 IAM 角色，並使用暫時性憑證列出該帳戶的 Amazon S3 儲存貯體。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
```

```
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AssumeRole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 設定和執行程式碼範例儲存庫。

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
```

```
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
 assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考AssumeRole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 assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
```

```
        Ok(e) => {
            println!("UserID :                {}",
 e.user_id().unwrap_or_default());
            println!("Account:                {}",
 e.account().unwrap_or_default());
            println!("Arn    :                {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK AssumeRole中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
                    async throws -> STSClientTypes.Credentials {
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
                throw ServiceHandlerError.authError
```

```
            }

            return credentials
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK AssumeRole中的斯威夫特 API 參考。

尋找未使用的 AWS 憑

若要提高您的安全性 AWS 帳戶，請移除不需要的 IAM 使用者登入資料 (也就是密碼和存取金鑰)。例如，當使用者離開您的組織或不再需要 AWS 存取權時，請尋找他們正在使用的認證，並確定他們不再運作。在理想情況下，如果不再需要可刪除憑證。有需要時，您隨時可以在日後重新建立它們。至少您應變更密碼或停用存取金鑰，讓前任持有者無法再繼續存取。

當然，未使用的定義可能有所不同，通常表示在指定期間內未使用的憑證。

尋找未使用的密碼

您可以使用檢視 AWS Management Console 使用者的密碼使用資訊。如果您有大量的使用者，您可以使用主控台來下載憑證報告與有關每位使用者上次使用他們的主控台密碼的資訊。您也可以從 AWS CLI 或 IAM API 存取這些資訊。

尋找未使用的密碼 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 如果必要，請將 Console last sign-in (主控台上次登入) 欄位新增到使用者表格：
   a. 在最右側的表格上方，選擇設定圖示 ( )。
   b. 在 選取可見欄 中選取 主控台上次登入。
   c. 選擇 確認 返回使用者清單。

4. [主控台上次登入] 欄會顯示使用者上次 AWS 透過主控台登入的日期。您可以使用此資訊來尋找其密碼超過指定期間都沒有登入的使用者。該欄位對於其密碼從未登入的使用者顯示 Never (永不)。None (無) 指出使用者無密碼。最近未使用的密碼可能就是待移除的項目。

> **Important**
>
> 由於服務問題，上次使用密碼的資料不包含從 2018 年 5 月 3 日 22:50 (太平洋日光時間) 到 2018 年 5 月 23 日 14:08 (太平洋日光時間) 使用的密碼。這會影響 IAM 主控台中顯示的上次登入日期，以及 IAM 登入資料報告中的密碼上次使用日期，並由 GetUser API 作業傳回。如果使用者在受影響的時間登入，傳回的上次使用密碼的日期會是使用者最後一次在 2018 年 5 月 3 日之前登入的日期。對於在 2018 年 5 月 23 日 14:08 (太平洋日光時間) 之後登入的使用者，傳回的上次使用密碼日期會是準確的。
> 如果您使用上次使用的密碼資訊來識別未使用的認證以進行刪除，例如刪除過去 90 天內未登入的使用者，建議您調整評估時段，使其包含 2018 年 5 月 23 日之後的日期。 AWS 或者，如果您的用戶使用訪問密鑰以 AWS 編程方式訪問，則可以參考訪問密鑰上次使用的信息，因為它對所有日期都是準確的。

透過下載憑證報告尋找未使用的密碼 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Credential report (憑證報告)。
3. 選擇 Download Report (下載報告) 以下載名稱為 `status_reports_<date>T<time>.csv` 的逗號分隔值 (CSV) 檔案。第五個欄位包含 `password_last_used` 欄與日期或以下其中一項：
   - N/A (無) – 完全沒有指派密碼的使用者。
   - no_information (無資訊) – 自從 IAM 在 2014 年 10 月 20 日開始追蹤密碼使用期限，未使用他們密碼的使用者。

若要尋找未使用的密碼 (AWS CLI)

執行以下命令來尋找未使用的密碼：

- `aws iam list-users` 傳回使用者清單，每個都有一個 PasswordLastUsed 值。如果缺少值，則表示使用者沒有密碼或者自從 IAM 在 2014 年 10 月 20 日開始追蹤密碼使用期限，未使用密碼。

若要尋找未使用的密碼 (AWS API)

呼叫以下操作來尋找未使用的密碼：

- `ListUsers` 傳回使用者集合，每個選項都有 <PasswordLastUsed> 值。如果缺少值，則表示使用者沒有密碼或者自從 IAM 在 2014 年 10 月 20 日開始追蹤密碼使用期限，未使用密碼。

如需關於下載憑證報告的命令的資訊，請參閱取得憑證報告 (AWS CLI)。

尋找未使用的存取金鑰

您可以使用 AWS Management Console 來檢視使用者的存取金鑰使用資訊。如果您有大量的使用者，您可以使用主控台來下載憑證報告，以尋找每位使用者上次使用他們的存取金鑰的時間。您也可以從 AWS CLI 或 IAM API 存取這些資訊。

尋找未使用的存取金鑰 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 如有必要，請將 Access key last used (上次使用的存取金鑰) 欄位新增到使用者表格：
   a. 在最右側的表格上方，選擇設定圖示 ( )。
   b. 在 選取可見欄 中選取 存取金鑰上次使用時間。
   c. 選擇 確認 返回使用者清單。
4. [存取金鑰上次使用] 資料行會顯示自使用者上次 AWS 以程式設計方式存取以來的天數。您可以使用此資訊來尋找存取金鑰超過指定期間都沒有使用的使用者。該欄會向沒有存取金鑰的使用者顯示 –。最近未使用的存取金鑰可能就是待移除的項目。

透過下載憑證報告尋找未使用的存取金鑰 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Credential Report (憑證報告)。
3. 選擇 Download Report (下載報告) 以下載名稱為 `status_reports_<date>T<time>.csv` 的逗號分隔值 (CSV) 檔案。欄位 11 到 13 包含存取金鑰 1 上次使用的日期、區域和服務資訊。欄位

16 到 18 包含存取金鑰 2 的相同資訊。如果使用者沒有存取金鑰，或者使用者自從 IAM 在 2015 年 4 月 22 日開始追蹤存取金鑰使用期限均未使用存取金鑰，該值為 N/A (無)。

若要尋找未使用的存取金鑰 (AWS CLI)

執行以下命令來尋找未使用的存取金鑰：

- `aws iam list-access-keys` 傳回有關使用者的存取金鑰的資訊，包括 AccessKeyID。
- `aws iam get-access-key-last-used` 需要存取金鑰 ID 並會傳回輸出，其中包含上次使用存取金鑰的 LastUsedDate、Region，以及上次所請求服務的 ServiceName。如果缺少 LastUsedDate，則表示自從 IAM 在 2015 年 4 月 22 日開始追蹤存取金鑰使用期限均未使用存取金鑰。

若要尋找未使用的存取金鑰 (AWS API)

呼叫以下操作來尋找未使用的存取金鑰：

- `ListAccessKeys` 傳回與指定的使用者關聯之存取金鑰的 AccessKeyID 值清單。
- `GetAccessKeyLastUsed` 需要存取金鑰 ID 並會傳回值集合。包含有是上次使用的存取金鑰的 LastUsedDate、Region，以及上次所請求服務的 ServiceName。如果缺少值，則表示使用者沒有存取金鑰，或者自從 IAM 在 2015 年 4 月 22 日開始追蹤存取金鑰使用期限均未使用存取金鑰。

如需關於下載憑證報告的命令的資訊，請參閱取得憑證報告 (AWS CLI)。

取得 AWS 帳戶的憑證報告

您可以產生並下載憑證報告，其中會列出帳戶中的所有使用者，及其各種憑證的狀態，包括密碼、存取金鑰和 MFA 裝置。您可以從 AWS SDK 和命令列工具或 IAM API 取得登入資料報告。 AWS Management Console

您可以使用憑證報告協助您進行稽核和合規性工作。您可以使用報告來稽核憑證生命週期要求的效果，如密碼和存取金鑰更新。您可以向外部稽核員提供報告，或向稽核員授予許可，以便該人員直接下載報告。

您可以依照每四小時一次的頻率來產生憑證報告。當您請求報告時，IAM 會先檢查過去四個小時內是否 AWS 帳戶 已產生報告。若是如此，便會下載最新的報告。如果帳戶的最新報告是四小時前產生的，或者如果帳戶無先前報告，則 IAM 會產生並下載新報告。

主題

必要許可

建立和下載報告必須要有以下許可：

- 建立憑證報告：`iam:GenerateCredentialReport`
- 下載報告：`iam:GetCredentialReport`

了解報告格式

憑證報告採取逗號分隔值 (CSV) 檔案為格式。您可以使用常用試算表軟體打開 CSV 檔以執行分析，也可以建構應用程式來以程式設計方式使用 CSV 檔並執行自訂分析。

CSV 檔案內含下列欄位：

使用者

易讀的使用者名稱。

arn

使用者的 Amazon Resource Name (ARN)。如需有關 ARN 的詳細資訊，請參閱 IAM ARN。

user_creation_time

使用者建立的日期和時間，採用 ISO 8601 日期時間格式。

password_enabled

當使用者有密碼時，此值為 TRUE，否則，它是FALSE。的值始終 AWS 帳戶根使用者是`not_supported`。

password_last_used

上次使用 AWS 帳戶根使用者 或使用者密碼登入 AWS 網站的日期和時間，採用 ISO 8601 日期時間格式。 AWS 擷取使用者上次登入時間的 AWS Management Console網站為「 AWS 論壇」和

「AWS Marketplace」。如果密碼在 5 分鐘的時間範圍內多次使用，則僅在此欄位中記錄此期間內的第一次使用。

- 在以下情況中，此欄位的值為 no_information：
  - 從未使用使用者密碼。
  - 沒有與密碼關聯的登入資料，例如，在 IAM 於 2014 年 10 月 20 日開始追蹤此資訊後，使用者密碼未曾使用。
- 當使用者沒有密碼時，此欄位中的值是 N/A (不適用)。

> **Important**
>
> 由於服務問題，上次使用密碼的資料不包含從 2018 年 5 月 3 日 22:50 (太平洋日光時間) 到 2018 年 5 月 23 日 14:08 (太平洋日光時間) 使用的密碼。這會影響 IAM 主控台中顯示的上次登入日期，以及 IAM 登入資料報告中的密碼上次使用日期，並由 GetUser API 作業傳回。如果使用者在受影響的時間登入，傳回的上次使用密碼的日期會是使用者最後一次在 2018 年 5 月 3 日之前登入的日期。對於在 2018 年 5 月 23 日 14:08 (太平洋日光時間) 之後登入的使用者，傳回的上次使用密碼日期會是準確的。
>
> 如果您使用上次使用的密碼資訊來識別未使用的認證以進行刪除，例如刪除過去 90 天內未登入的使用者，建議您調整評估時段，使其包含 2018 年 5 月 23 日之後的日期。 AWS 或者，如果您的用戶使用訪問密鑰以 AWS 編程方式訪問，則可以參考訪問密鑰上次使用的信息，因為它對所有日期都是準確的。

password_last_changed

使用者密碼上次設定的日期和時間，採用 ISO 8601 日期時間格式。若使用者沒有密碼，此欄位中的值是 N/A (不適用)。 AWS 帳戶 (根) 的值永遠是not_supported。

password_next_rotation

如果帳戶的密碼政策要求密碼輪換，則此欄位包含使用者需要設定新密碼的日期和時間，採用 ISO 8601 日期時間格式。 AWS 帳戶 (根) 的值永遠是not_supported。

mfa_active

當使用者啟用多重要素驗證 (MFA) 裝置時，此值為 TRUE，否則為 FALSE。

access_key_1_active

當使用者有存取金鑰且存取金鑰狀態為 Active 時，此值為 TRUE，否則為 FALSE。

access_key_1_last_rotated

使用者的存取金鑰建立或前次變更的時間與日期，採用 ISO 8601 日期時間格式。若使用者沒有主動式存取金鑰，此欄位中的值是 N/A (不適用)。

access_key_1_last_used_date

使用者存取金鑰最近用於登入 AWS API 請求的日期與時間，採用 ISO 8601 日期時間格式。如果存取金鑰在 15 分鐘的時間範圍內多次使用，則僅在此欄位中記錄第一次使用。

在以下情況中，此欄位的值為 N/A (不適用)：

- 使用者無存取金鑰。
- 從未使用存取金鑰。
- 自 IAM 於 2015 年 4 月 22 日開始追蹤此資訊後，存取金鑰未曾使用。

access_key_1_last_used_region

最近使用過存取金鑰的 AWS 區域。如果存取金鑰在 15 分鐘的時間範圍內多次使用，則僅在此欄位中記錄第一次使用。

在以下情況中，此欄位的值為 N/A (不適用)：

- 使用者無存取金鑰。
- 從未使用存取金鑰。
- 存取金鑰最後一次使用的時間為 IAM 於 2015 年 4 月 22 日開始追蹤此資訊之前。
- 上次使用的服務沒有區域限制，如 Amazon S3。

access_key_1_last_used_service

最近使用存取金鑰存取的 AWS 服務。此欄位中的值會使用服務的命名空間 — 舉例來說，Amazon S3 為 s3、而 Amazon EC2 為 ec2。如果存取金鑰在 15 分鐘的時間範圍內多次使用，則僅在此欄位中記錄第一次使用。

在以下情況中，此欄位的值為 N/A (不適用)：

- 使用者無存取金鑰。
- 從未使用存取金鑰。
- 存取金鑰最後一次使用的時間為 IAM 於 2015 年 4 月 22 日開始追蹤此資訊之前。

access_key_2_active

當使用者有第二個存取金鑰且第二個存取金鑰狀態為 Active 時，此值為 TRUE。否則為 FALSE。

> **Note**
>
> 使用者最多可有兩個存取金鑰，首先更新金鑰，然後刪除先前的金鑰，從而使輪換變得更簡單。如需有關更新存取金鑰的詳細資訊，請參閱 更新存取金鑰。

access_key_2_last_rotated

使用者的第二個存取金鑰建立或前次更新的時間與日期，採用 ISO 8601 日期時間格式。若使用者沒有第二個主動式存取金鑰，此欄位中的值是 N/A (不適用)。

access_key_2_last_used_date

使用者第二個 AWS 存取金鑰最近用於簽署 API 要求時的日期和時間，採用 ISO 8601 日期時間格式。如果存取金鑰在 15 分鐘的時間範圍內多次使用，則僅在此欄位中記錄第一次使用。

在以下情況中，此欄位的值為 N/A (不適用)：

- 使用者沒有第二個存取金鑰。
- 從未使用第二個存取金鑰。
- 使用者的第二個存取金鑰最後一次使用的時間為 IAM 於 2015 年 4 月 22 日開始追蹤此資訊之前。

access_key_2_last_used_region

使用者的第二個最近使用過存取金鑰的 AWS 區域。如果存取金鑰在 15 分鐘的時間範圍內多次使用，則僅在此欄位中記錄第一次使用。在以下情況中，此欄位的值為 N/A (不適用)：

- 使用者沒有第二個存取金鑰。
- 從未使用第二個存取金鑰。
- 使用者的第二個存取金鑰最後一次使用的時間為 IAM 於 2015 年 4 月 22 日開始追蹤此資訊之前。
- 上次使用的服務沒有區域限制，如 Amazon S3。

access_key_2_last_used_service

最近使用使用者的第二個存取金鑰存取的 AWS 服務。此欄位中的值會使用服務的命名空間 — 舉例來說，Amazon S3 為 s3、而 Amazon EC2 為 ec2。如果存取金鑰在 15 分鐘的時間範圍內多次使用，則僅在此欄位中記錄第一次使用。在以下情況中，此欄位的值為 N/A (不適用)：

- 使用者沒有第二個存取金鑰。
- 從未使用第二個存取金鑰。

- 使用者的第二個存取金鑰最後一次使用的時間為 IAM 於 2015 年 4 月 22 日開始追蹤此資訊之前。

cert_1_active

當使用者有 X.509 簽署的憑證且該憑證的狀態為 `Active` 時，此值為 `TRUE`，否則為 `FALSE`。

cert_1_last_rotated

使用者的簽署憑證建立或前次變更的時間與日期，採用 ISO 8601 日期時間格式。若使用者沒有主動式簽署憑證，此欄位中的值是 `N/A` (不適用)。

cert_2_active

當使用者有第二個 X.509 簽署的憑證且該憑證的狀態為 `Active` 時，此值為 `TRUE`，否則為 `FALSE`。

> **Note**
>
> 使用者最多可有兩個 X.509 簽署憑證，讓輪換更容易。

cert_2_last_rotated

使用者的第二個簽署憑證建立或前次變更的時間與日期，採用 ISO 8601 日期時間格式。若使用者沒有第二個主動式簽署憑證，此欄位中的值是 `N/A` (不適用)。

取得憑證報告 (主控台)

您可以使用 AWS Management Console 將認證報告下載為逗號分隔值 (CSV) 檔案。

下載憑證報告 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Credential report (憑證報告)。
3. 選擇 Download Report (下載報告)。

取得憑證報告 (AWS CLI)

下載憑證報告 (AWS CLI)

1. 產生認證報告。 AWS 儲存單一報告。如果報告存在，產生憑證報告會覆寫先前的報告。`aws iam generate-credential-report`
2. 檢視上次產生的報告：`aws iam get-credential-report`

取得認證報告 (AWS API)

若要下載憑證報告 (AWS API)

1. 產生認證報告。 AWS 儲存單一報告。如果報告存在，產生憑證報告會覆寫先前的報告。`GenerateCredentialReport`
2. 檢視上次產生的報告：`GetCredentialReport`

將 IAM 搭配使用 CodeCommit：Git 登入資料、安全殼層金鑰和 AWS 存取金鑰

CodeCommit 是在 AWS 雲端託管私有 Git 儲存庫的受管理版本控制服務。若要使用 CodeCommit，請將 Git 用戶端設定為與 CodeCommit 儲存庫通訊。作為此組態的一部分，您提供 CodeCommit 可用於驗證您的 IAM 登入資料。IAM 支援三 CodeCommit 種登入資料類型：

- Git 認證，一個由 IAM 產生的使用者名稱和密碼組，可用來透過 HTTPS 與 CodeCommit 儲存庫通訊。
- SSH 金鑰是本機產生的公開-私密 key pair，您可以與 IAM 使用者建立關聯，以透過 SSH 與 CodeCommit 儲存庫通訊。
- AWS 存取金鑰，您可以 AWS CLI 將其與隨附的認證協助程式搭配使用，以透過 HTTPS 與 CodeCommit 儲存庫進行通訊。

> **Note**
>
> 您無法使用 SSH 金鑰或 Git 憑證來存取另一個 AWS 帳戶中的儲存庫。要了解如何為另一個 IAM 使用者和群組設定存AWS CodeCommit 放 CodeCommit 庫的存取權限 AWS 帳戶，請參閱使用AWS CodeCommit 者指南中的使用角色設定存放庫的跨帳戶存取。

如需有關每個選項的詳細資訊，請參閱下列各節。

使用 Git 認證和 HTTPS CodeCommit（推薦）

使用 Git 憑證後，您可以為 IAM 使用者產生靜態的使用者名稱和密碼對，然後將這些憑證用於 HTTPS 連線。您還可以將這些憑證用於支援靜態的 Git 憑證的任何第三方工具或整合開發環境 (IDE)。

由於這些憑證對於所有受支援的作業系統都是通用的，並且與大多數憑證管理系統、開發環境和其他軟體開發工具相容，因此建議使用此方法。您可以隨時重設 Git 憑證的密碼。您還可以使憑證處於非作用中，或者在不再需要時刪除它們。

> **Note**
>
> 您無法為 Git 憑證選擇自己的使用者名稱或密碼。IAM 會為您產生這些登入資料，以協助確保它們符合中的安全標準 AWS 和安全存放庫 CodeCommit。您只能在產生憑證時作一次下載。請確定將憑證儲存在安全的位置。如有需要，您可以隨時重設密碼，但這樣做會使用舊密碼設定的任何連線無效。在連線之前，必須重新設定連線以使用新密碼。

如需詳細資訊，請參閱下列主題：

- 若要建立 IAM 使用者，請參閱 在您的中建立 IAM 使用者 AWS 帳戶。
- 若要使用產生和使用 Git 認證 CodeCommit，請參閱使用者指南中的〈使用 Git 認證的 HTTPS 使用AWS CodeCommit 者〉。

> **Note**
>
> 產生 Git 憑證後變更 IAM 使用者的名稱不會變更 Git 憑證的使用者名稱。使用者名稱和密碼保持不變，且仍然有效。

若要更新服務特定的憑證

1. 除了目前正在使用的設定之外，還要建立第二個特定於服務的憑證設定。
2. 更新您的所有應用程式，以使用新設定的憑證並驗證應用程式是否正常工作。
3. 將原始憑證的狀態變更為「非作用中」。
4. 確認您的所有應用程式仍在運作。

5. 刪除非作用中特定於服務的憑證

使用安全殼層金鑰和 SSH CodeCommit

透過 SSH 連線，您可以在本機電腦上建立 Git 並 CodeCommit 用於 SSH 驗證的公開和私密金鑰檔案。您將公有金鑰與 IAM 使用者關聯，並將私有金鑰存放在本機電腦上。如需詳細資訊，請參閱下列主題：

- 若要建立 IAM 使用者，請參閱 在您的中建立 IAM 使用者 AWS 帳戶。
- 若要建立 SSH 公有金鑰，並將其與 IAM 使用者關聯的詳細資訊，請參閱在 Linux、macOS 或 Unix 上的 SSH 連線，或查看 AWS CodeCommit 使用者指南中的在 Windows 上的 SSH 連線。

> **Note**
>
> 公有金鑰必須以 ssh-rsa 格式或 PEM 格式編碼。公有金鑰的最小位元長度為 2048 位元，最長為 16384 位元。這與您上傳的檔案大小不同。例如，您可以產生 2048 位元索引鍵，產生的 PEM 檔案長度為 1679 位元組。如果以其他格式或大小提供公有金鑰，則會看到錯誤訊息，指出該金鑰格式無效。

使用 HTTPS 與 AWS CLI 憑證助手和 CodeCommit

作為使用 Git 登入資料進行 HTTPS 連線的替代方法，您可以在 Git 需要驗證以與 CodeCommit 儲存庫互動時，允許 Git 使用 IAM 使用者登入資料的加密簽署版本或 Amazon EC2 執行個體角色。 AWS 對於不需要 IAM 使用者的 CodeCommit 儲存庫，這是唯一的連線方法。這也是使用聯合身分存取和臨時憑證的唯一方法。如需詳細資訊，請參閱下列主題：

- 若要進一步了解有關聯合身分存取的詳細資訊，請參閱 身分提供者與聯合 和 對外部驗證的使用者提供存取權 (聯合身分)。
- 要了解有關臨時登入資料的更多資訊，請參閱IAM 中的暫時安全憑證和暫時存取 CodeCommit存放庫。

AWS CLI 憑證協助程式與其他認證協助程式系統 (例如「鑰匙串存取」或「Windows 認證管理」) 不相容。當使用憑證協助程式設定 HTTPS 連線時，會有額外的組態考量。如需詳細資訊，請參閱使用者指南中的 < 在 Windows 上使用 AWS CLI 認證協助程式 > 在 Linux、macOS 或 Unix 上AWS CodeCommit 使用 AWS CLI 認證協助程式的 HTTPS 連線進行 HTTPS 連線。

使用 IAM 與 Amazon Keyspaces (適用於 Apache Cassandra)

Amazon Keyspaces (適用於 Apache Cassandra) 是一種可擴展、高可用性且受管的 Apache Cassandra 相容資料庫服務。您可以透過或以程式設計方式存取 Amazon Keyspaces。 AWS Management Console若要使用服務特定憑證以程式設計方式存取 Amazon Keyspaces，您可以使用 `cqlsh` 或開放原始碼 Cassandra 驅動程式。服務特定的憑證包括使用者名稱和密碼，例如 Cassandra 用於身分驗證和存取管理的使用者名稱和密碼。對於每位使用者，每個支援的服務最多可以有兩組服務特定憑證。

要使用訪問密鑰以編程方式 AWS 訪問 Amazon 密鑰空間，您可以使用 AWS SDK， AWS Command Line Interface （ AWS CLI ）或開源卡桑德拉驅動程序與 Sigv4 插件。如需進一步了解，請參閱《Amazon Keyspaces (適用於 Apache Cassandra) 開發人員指南》中的 Connecting programmatically to Amazon Keyspaces (以程式設計方式連接到 Amazon Keyspaces)。

> **Note**
>
> 如果您打算只透過主控台與 Amazon Keyspaces 互動，則不需要產生服務特定的憑證。如需詳細資訊，請參閱《Amazon Keyspaces (適用於 Apache Cassandra) 開發人員指南》中的使用主控台存取 Amazon Keyspaces。

如需有關存取 Amazon Keyspaces 所需許可的詳細資訊，請參閱 Amazon Keyspaces (適用於 Apache Cassandra) 開發人員指南中的 Amazon Keyspaces (適用於 Apache Cassandra) 以身分為基礎的政策範例。

產生 Amazon Keyspaces 憑證 (主控台)

您可以使用為您的 IAM AWS Management Console 使用者產生 Amazon Keyspaces (適用於 Apache 卡桑德拉) 登入資料。

產生 Amazon Keyspaces 服務特定的憑證 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)，然後選擇需要憑證的使用者名稱。
3. 在 Security Credentials (安全憑證) 索引標籤的 Credentials for Amazon Keyspaces (for Apache Cassandra) (Amazon Keyspaces (適用於 Apache Cassandra) 憑證) 上，選擇 Generate credentials (產生憑證)。

4. 您的服務特定憑證現在可供使用。這是唯一可以檢視或下載密碼的機會。您稍後無法進行復原。不過，您可以隨時重設密碼。請將使用者和密碼儲存在安全的位置，因為稍後需要用到。

產生 Amazon Keyspaces 憑證 (AWS CLI)

您可以使用為您的 IAM AWS CLI 使用者產生 Amazon Keyspaces (適用於 Apache 卡桑德拉) 登入資料。

產生 Amazon Keyspaces 服務特定的憑證 (AWS CLI)

- 使用下列命令：
  - AWS IAM create-service-specific-credential

生成 Amazon Keyspaces 憑據（ AWS API ）

您可以使用 AWS API 為您的 IAM 用戶生成 Amazon Keyspaces（用於阿帕奇卡桑德拉）憑據。

若要產生 Amazon Keyspaces 服務特定登入資料 (API)AWS

- 請完成下列操作：
  - CreateServiceSpecificCredential

在 IAM 中管理伺服器憑證

若要在中啟用 HTTPS 連線到您的網站或應用程式 AWS，您需要 SSL/TLS 伺服器憑證。處理 AWS Certificate Manager (ACM) 可支援區域中的憑證時，我們建議您使用 ACM 來佈建、管理和部署您的伺服器憑證。在不支援的區域中，您必須使用 IAM 作為憑證管理員。如需了解 ACM 支援哪些區域，請參閱 AWS 一般參考 中的 AWS Certificate Manager 端點和配額。

ACM 為佈建、管理和部署您伺服器憑證的首選工具。使用 ACM，您可以要求憑證或將現有的 ACM 或外部憑證部署到 AWS 資源。ACM 提供的憑證為免費且會自動續約。在支援的區域中，您可以使用 ACM 從主控台或以程式設計方式管理伺服器憑證。如需有關 ACM 的詳細資訊，請參閱 AWS Certificate Manager 使用者指南。如需有關請求 ACM 憑證的詳細資訊，請參閱 AWS Certificate Manager 使用者指南中的請求公有憑證或請求私有憑證。如需有關將第三方憑證匯入 ACM 的詳細資訊，請參閱《AWS Certificate Manager 使用者指南》中的匯入憑證。

只有當您必須在 ACM 不支援的區域中支援 HTTPS 連接時，才應使用 IAM 作為憑證管理器。IAM 會安全地加密您的私有金鑰並將加密的版本儲存在 IAM SSL 憑證存放區中。IAM 支援在所有區域部署伺服器憑證，但您必須向外部供應商取得憑證才能搭配使用 AWS。不能將 ACM 憑證上傳到 IAM。此外，也無法從 IAM 主控台管理憑證，

如需有關如何將第三方憑證上傳到 IAM 的詳細資訊，請參閱下列主題。

目錄

上傳伺服器憑證 (AWS API)

若要將伺服器憑證上傳至 IAM，您必須提供憑證及相符的私有金鑰。當憑證並非自我簽署憑證時，您還必須提供憑證鏈。(上傳自簽署憑證時無需憑證連結。) 在上傳憑證前，請確保您已具有所有這些項目且滿足以下條件：

- 憑證在上傳時必須有效。您不能在憑證有效期開始 (憑證的 `NotBefore` 日期) 之前或憑證有效期到期 (憑證的 `NotAfter` 日期) 之後上傳憑證。
- 私有金鑰必須為未加密。您無法上傳受密碼或複雜密碼保護的私有金鑰。有關解密已加密的私有金鑰的說明資訊，請參閱 故障診斷。
- 憑證、私有金鑰和憑證鏈全都必須採用 PEM 編碼。有關將這些項目轉換為 PEM 格式的說明資訊，請參閱 故障診斷。

若要使用 IAM API 上傳憑證，請傳送UploadServerCertificate請求。以下範例顯示如何使用 AWS Command Line Interface (AWS CLI) 執行此作業。該範例假設如下：

- PEM 編碼的憑證存放在名為 `Certificate.pem` 的檔案中。
- PEM 編碼的憑證鏈存放在名為 `CertificateChain.pem` 的檔案中。
- PEM 編碼的未加密私有金鑰存放在名為 `PrivateKey.pem` 的檔案中。

- (選用) 您要使用鍵值組來標記伺服器憑證。例如，您可以新增標籤金鑰 Department 和標籤值 Engineering，以協助您識別和整理憑證。

若要使用下列範例命令，請將這些檔案名稱取代為您自己的檔案名稱。*ExampleCertificate*以上傳憑證的名稱取代。如果您要標記憑證，請以您自己的值取代*ExampleKey*和*ExampleValue*標記金鑰值配對。在連續的一行中輸入命令。為方便閱讀，以下範例包含分行符號和多餘的空格。

```
aws iam upload-server-certificate --server-certificate-name ExampleCertificate
                                  --certificate-body file://Certificate.pem
                                  --certificate-chain file://CertificateChain.pem
                                  --private-key file://PrivateKey.pem
                                  --tags '{"Key": "ExampleKey", "Value":
 "ExampleValue"}'
```

如果上述命令執行成功，則它將傳回上傳憑證的相關中繼資料，包括其 Amazon Resource Name (ARN)、易記名稱、識別符 (ID)、過期日期、標籤等。

> **Note**
>
> 如果您要上傳伺服器憑證以搭配 Amazon 使用 CloudFront，則必須使用--path選項指定路徑。路徑必須以 /cloudfront 開頭，且必須包含結尾反斜線 (例如，/cloudfront/test/)。

若要使用上 AWS Tools for Windows PowerShell 傳憑證，請使用「發佈 IAM」。ServerCertificate

擷取伺服器憑證 (AWS API)

若要使用 IAM API 擷取憑證，請傳送要GetServerCertificate求。以下範例顯示如何使用 AWS CLI執行此作業。以要擷取的憑證名稱取代*ExampleCertificate*。

```
aws iam get-server-certificate --server-certificate-name ExampleCertificate
```

如果上述命令執行成功，則它將傳回憑證、憑證連結 (如果已上傳一個) 和有關憑證的中繼資料。

> **Note**
>
> 在您上傳後，無法從 IAM 下載或擷取私有金鑰。

若要使用擷 AWS Tools for Windows PowerShell 取憑證，請使用 Get-IAM ServerCertificate。

列出伺服器憑證 (AWS API)

若要使用 IAM API 列出您上傳的伺服器憑證，請傳送ListServerCertificates請求。以下範例顯示如何使用 AWS CLI執行此作業。

```
aws iam list-server-certificates
```

如果上述命令執行成功，將傳回包含有關每個憑證的中繼資料的清單。

若要使用列 AWS Tools for Windows PowerShell 出上傳的伺服器憑證，請使用 Get-ServerCertificates IAM。

標記和取消標記伺服器憑證 (AWS API)

您可以將標籤連接到您的 IAM 資源，以整理和控制對其的存取。若要使用 IAM API 標記現有的伺服器憑證，請傳送TagServerCertificate要求。以下範例顯示如何使用 AWS CLI執行此作業。

```
aws iam tag-server-certificate --server-certificate-name ExampleCertificate
                               --tags '{"Key": "ExampleKey", "Value":
 "ExampleValue"}'
```

上述命令成功時，不會傳回任何輸出。

若要使用 IAM API 取消標記伺服器憑證，請傳送UntagServerCertificate要求。以下範例顯示如何使用 AWS CLI執行此作業。

```
aws iam untag-server-certificate --server-certificate-name ExampleCertificate
                               --tag-keys ExampleKeyName
```

上述命令成功時，不會傳回任何輸出。

重新命名伺服器憑證或更新其路徑 (AWS API)

若要使用 IAM API 重新命名伺服器憑證或更新其路徑，請傳送UpdateServerCertificate要求。以下範例顯示如何使用 AWS CLI執行此作業。

若要使用以下範例指令，請將取代舊與新的憑證名稱與憑證路徑，並在連續的一行中輸入命令。為方便閱讀，以下範例包含分行符號和多餘的空格。

```
aws iam update-server-certificate --server-certificate-name ExampleCertificate
                                  --new-server-certificate-name CloudFrontCertificate
                                  --new-path /cloudfront/
```

當前述命令成功時，不會傳回任何輸出。

若要使用重新命名伺服器憑證或更新其路徑，請使用 Update-ServerCertificate IAM。 AWS Tools for Windows PowerShell

刪除伺服器憑證 (AWS API)

若要使用 IAM API 刪除伺服器憑證，請傳送DeleteServerCertificate要求。以下範例顯示如何使用 AWS CLI執行此作業。

若要使用下列範例命令，請*ExampleCertificate*以要刪除的憑證名稱取代。

```
aws iam delete-server-certificate --server-certificate-name ExampleCertificate
```

當前述命令成功時，不會傳回任何輸出。

若要使用刪 AWS Tools for Windows PowerShell 除伺服器憑證，請使用移除 IAM。ServerCertificate

故障診斷

您必須先確保憑證、私有金鑰和憑證連結均使用 PEM 編碼，然後才能將憑證上傳到 IAM。您還必須確保私有金鑰為未加密。請參閱以下範例。

Example PEM 編碼憑證範例

```
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

Example PEM 編碼的未加密私有金鑰範例

```
-----BEGIN RSA PRIVATE KEY-----
Base64-encoded private key
-----END RSA PRIVATE KEY-----
```

Example PEM 編碼的憑證鏈範例

憑證鏈包含一或多個憑證。您可以使用文字編輯器、Windows 的 copy 命令，或 Linux cat 命令，將憑證檔案串連為憑證鏈。當您包含多個憑證時，每個憑證必須認證先前的憑證。您可以透過串連憑證 (包含上一個根 CA 憑證) 來完成此動作。

以下範例包含三個憑證，但您的憑證鏈可包含更多或更少憑證。

```
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

如果這些項目在上傳到 IAM 時未採用正確的格式，您可以使用 OpenSSL 將其轉換為正確的格式。

將憑證或憑證鍊從 DER 轉換為 PEM

使用 OpenSSL x509 命令，如下列範例。在以下範例命令中，將 *Certificate.der* 替換為包含您的 DER 編碼的憑證檔案的名稱。以偏好的輸出檔案名稱取代 *Certificate.pem*，以包含 PEM 編碼的憑證。

```
openssl x509 -inform DER -in Certificate.der -outform PEM -out Certificate.pem
```

若要將私有金鑰從 DER 轉換為 PEM

使用 OpenSSL rsa 命令，如下列範例。在以下範例命令中，將 *PrivateKey.der* 替換為包含您的 DER 編碼的私有金鑰檔案的名稱。以偏好的輸出檔案名稱取代 *PrivateKey.pem*，以包含 PEM 編碼的私有金鑰。

```
openssl rsa -inform DER -in PrivateKey.der -outform PEM -out PrivateKey.pem
```

解密已加密的私有金鑰 (移除密碼或密碼短語)

使用 OpenSSL rsa 命令，如下列範例。若要使用以下範例命令，將 *EncryptedPrivateKey.pem* 替換為包含您的已加密私有金鑰檔案的名稱。以偏好的輸出檔案名稱取代 *PrivateKey.pem*，以包含 PEM 編碼的未加密私有金鑰。

```
openssl rsa -in EncryptedPrivateKey.pem -out PrivateKey.pem
```

將憑證 bundle 從 PKCS#12 (PFX) 轉換為 PEM

使用 OpenSSL pkcs12 命令，如下列範例。在以下範例命令中，將 *CertificateBundle.p12* 替換為包含您的 PKCS#12 編碼的憑證 bundle 的名稱。以偏好的輸出檔案名稱取代 *CertificateBundle.pem*，以包含 PEM 編碼的憑證 bundle。

```
openssl pkcs12 -in CertificateBundle.p12 -out CertificateBundle.pem -nodes
```

將憑證 bundle 從 PKCS#7 轉換為 PEM

使用 OpenSSL pkcs7 命令，如下列範例。在以下範例命令中，將 *CertificateBundle.p7b* 替換為包含您的 PKCS#7 編碼的憑證 bundle 的名稱。以偏好的輸出檔案名稱取代 *CertificateBundle.pem*，以包含 PEM 編碼的憑證 bundle。

```
openssl pkcs7 -in CertificateBundle.p7b -print_certs -out CertificateBundle.pem
```

IAM 使用者群組

IAM 使用者群組是 IAM 使用者的集合。使用者群組可讓您指定多個使用者的許可，從而可以更輕鬆地管理這些使用者的許可。例如，您可以擁有一個名為 Admins 的使用者群組，並為該使用者群組提供典型的管理員許可。該使用者群組中的任何使用者都自動擁有 Admins 群組許可。如果新使用者加入您的組織並需要管理員權限，您可以透過將使用者新增到 Admins 使用者群組來指派適當的許可。如果某人在您的組織中變更工作，而不是編輯該使用者的許可，您可以將他們從舊使用者群組中移除，並將他們新增到適當的新使用者群組中。

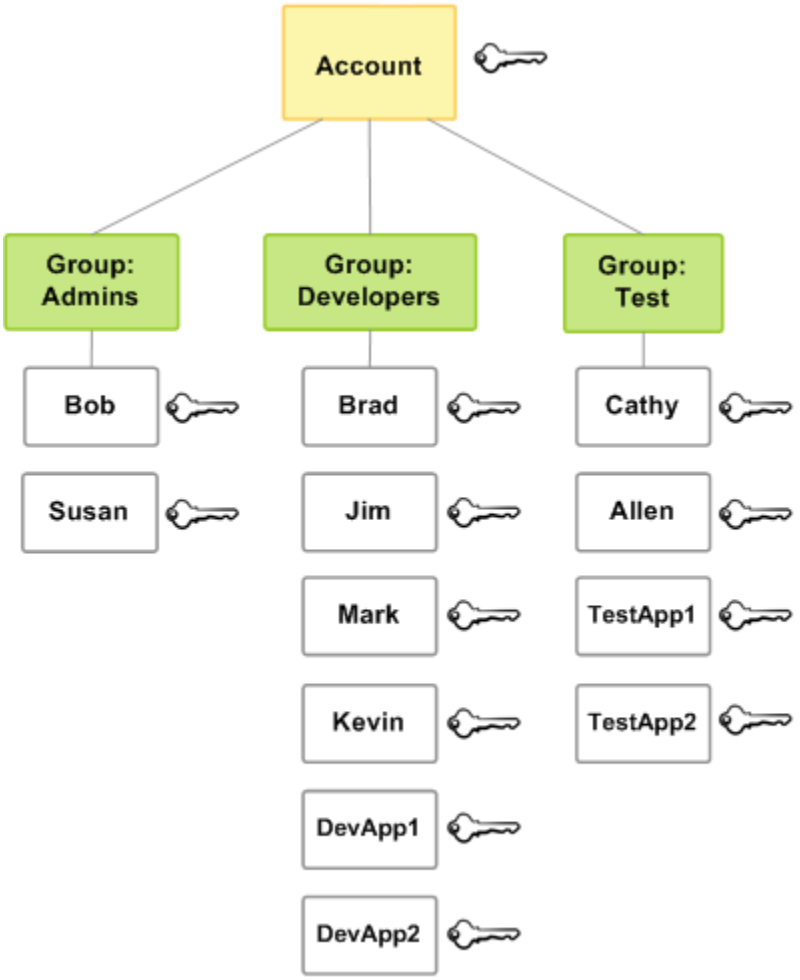
您可以將身分型政策連接至使用者群組，以便使用者群組中的所有使用者都會收到該政策的許可。您無法將使用者群組識別為政策 (例如資源型政策) 中的 `Principal`，因為群組與許可 (而非驗證) 相關，

並且主體是經過驗證的 IAM 實體。如需有關政策類型的詳細資訊，請參閱 以身分為基礎和以資源為基礎的政策。

以下是使用者群組的一些重要特性：

- 使用者群組可以包含許多使用者，使用者可以屬於多個使用者群組。
- 使用者群組不能為巢狀；群組只能包含使用者，而不包含其他使用者群組。
- 沒有預設使用者群組自動包含 AWS 帳戶中的所有使用者。如果您想擁有這樣的使用者群組，您必須建立它並指派每個新使用者到該群組。
- 中 IAM 資源的數量和大小 (例如群組數目以及使用者可以加入的群組數目) 會受到限制。 AWS 帳戶如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

下圖顯示一個小型公司的簡單範例。該公司擁有者為使用者建立 Admins 使用者群組，以便隨著公司的成長建立和管理其他使用者。所以此 Admins 使用者群組會建立 Developers 使用者群組和 Test 使用者群組。這些使用者群組都包含與 (Jim、Brad、 DevApp 1 等) 互動的使用者 AWS (人類和應用程式)。每個使用者都有一組個別的安全憑證。在這個範例中，每個使用者均屬於單一使用者群組。不過，使用者可屬於多個使用者群組。



建立 IAM 使用者群組

> **Note**
>
> 最佳作法是，建議您要求人類使用者與身分識別提供者使用同盟，才能 AWS 使用臨時登入資料進行存取。如果遵循最佳實務，則您不用管理 IAM 使用者和群組。相反地，您的使用者和群組會在外部進行管理， AWS 並且能夠以同盟身分存取 AWS 資源。聯合身分識別是來自企業使用者目錄、Web 身分識別提供者、 AWS Directory Service、Identity Center 目錄的使用者，或使用透過身分識別來源提供的認證存取 AWS 服務的任何使用者。聯合身分使用由其身分提供者定義的群組。如果您使用的是 AWS IAM Identity Center，請參閱使用指南中的在 IAM 身分中心中管理身分，以取得有關在 IAM 身分中心建立使用AWS IAM Identity Center 者和群組的資訊。

若要設定使用者群組，您需要建立群組。然後，根據您期望群組中的使用者執行的工作，來提供群組許可。最後，將使用者新增到群組。

如需有關建立使用者群組所需的許可資訊，請參閱 存取 IAM 資源所需的許可。

建立 IAM 使用者群組和連接政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中選擇 User groups (使用者群組)，然後選擇 Create group (建立群組)。
3. 針對 User group name (使用者群組名稱)，請輸入群組名稱。

> **Note**
>
> AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。群組名稱可以是長達 128 個字母、數字以及這些字元的組合：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、@ 符號、底線 (_) 以及連字號 (-)。名稱在帳戶中必須是唯一的。它們無法透過大小寫進行區分。例如，您無法建立名為 **ADMINS** 和 **admins** 的群組。

4. 在使用者清單中，針對您要新增到群組的每個使用者，選取其核取方塊。
5. 在政策清單中，對於您要套用到群組所有成員的每個政策選取核取方塊。
6. 選擇 Create group (建立群組)。

若要建立 IAM 使用者群組 (AWS CLI 或 AWS API)

請使用下列其中一個：

- AWS CLI：aws iam create-group
- AWS API：CreateGroup

管理 IAM 使用者群組

Amazon Web Services 提供管理 IAM 使用者群組的多種工具。有關在使用者群組中新增與移除使用者所需的許可之相關資訊，請參閱 存取 IAM 資源所需的許可。

主題

- 列出 IAM 使用者群組

- 在 IAM 使用者群組中新增和移除使用者
- 將政策連接至 IAM 使用者群組
- 重新命名 IAM 使用者群組
- 刪除 IAM 使用者群組

列出 IAM 使用者群組

您可以列出帳戶中的所有使用者群組，列出使用者群組中的使用者，以及列出使用者所屬的使用者群組。如果您使用 AWS CLI 或 AWS API，則可以列出具有特定路徑前綴的所有用戶組。

列出您帳戶中的所有使用者群組

執行下列任何一項：

- AWS Management Console：在導覽窗格中，選擇 User groups (使用者群組)。
- AWS CLI：aws iam list-groups
- AWS API：ListGroups

若要列出特定使用者群組中的使用者

執行下列任何一項：

- AWS Management Console：在導覽窗格中，選擇 User groups (使用者群組)，選擇群組名稱，然後選擇 Users (使用者) 索引標籤。
- AWS CLI：aws iam get-group
- AWS API：GetGroup

列出使用者所屬的所有使用者群組

執行下列任何一項：

- AWS Management Console：在導覽窗格中，選擇 Users (使用者)，選擇使用者名稱，然後選擇 Groups (群組) 標籤。
- AWS CLI：AWS list-groups-for-user
- AWS API：ListGroupsForUser

在 IAM 使用者群組中新增和移除使用者

使用使用者群組來將相同許可一次套用到多個使用者。您可以接著從 IAM 使用者群組新增或移除使用者。這在有人員加入和離開您的組織時很實用

檢視政策存取

變更政策的許可之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

在使用者群組中新增或移除使用者 (主控台)

您可以使用在 AWS Management Console 使用者群組中新增或移除使用者。

將使用者新增至 IAM 使用者群組 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 User groups (使用者群組)，然後選擇群組的名稱。
3. 選擇 Users (使用者) 索引標籤，然後選擇 Add Users (新增使用者)。選取要新增之使用者旁的核取方塊。
4. 選擇 Add users (新增使用者)。

從 IAM 群組移除使用者 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 User groups (使用者群組)，然後選擇群組的名稱。
3. 選擇 Users (使用者) 索引標籤。選取要移除之使用者旁的核取方塊，然後選擇 Remove users (移除使用者)。

在使用者群組中新增或移除使用者 (AWS CLI)

您可以使用在 AWS CLI 使用者群組中新增或移除使用者。

將使用者新增至 IAM 使用者群組 (AWS CLI)

- 使用下列命令：

- AWS IAM add-user-to-group

從 IAM 使用者群組移除使用者 (AWS CLI)

- 使用下列命令：
  - AWS IAM remove-user-from-group

新增或移除使用者群組 (AWS API) 中的使用者

您可以使用 AWS API 新增或移除使用者群組中的使用者。

若要將使用者新增至 IAM 群組 (AWS API)

- 請完成下列操作：
  - AddUserToGroup

若要從 IAM 使用者群組 (AWS API) 移除使用者

- 請完成下列操作：
  - RemoveUserFromGroup

將政策連接至 IAM 使用者群組

您可以將AWS 受管理策略 (也就是提供的預先寫入的策略) 附加 AWS至使用者群組，如下列步驟所述。若要連接客戶受管政策 ，即具有您建立的自訂許可的政策，必須先建立政策。如需建立客戶受管政策的詳細資訊，請參閱 建立 IAM 政策。

如需有關許可和政策的詳細資訊，請參閱 AWS 資源存取管理。

將政策連接至使用者群組 (主控台)

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 User groups (使用者群組)，然後選擇群組的名稱。
3. 選擇 Permissions (許可) 標籤。

4. 選擇 [新增權限]，然後選擇 [附加原則]
5. 連接至使用者群組的目前政策會顯示在目前的許可政策清單。在 Other permissions policies (其他許可政策) 清單中，選取要連接的政策名稱旁的核取方塊。您可以使用搜尋方塊來依類型和政策名稱篩選政策清單。
6. 選取要附加至 IAM 使用者群組的政策，然後選擇 [附加政策]。

若要將政策附加至使用者群組 (AWS CLI 或 AWS API)

執行下列任何一項：

- AWS CLI：AWS attach-group-policy
- AWS API：AttachGroupPolicy

重新命名 IAM 使用者群組

當您更改使用者群組名稱或路徑時，會發生以下情況：

- 連接到使用者群組的所有政策繼續採用新名稱。
- 使用者群組保留新名稱下的所有使用者。
- 使用者群組的唯一 ID 保持不變。如需有關唯一 ID 的詳細資訊，請參閱 唯一識別碼。

IAM 不會自動更新將該使用者群組視為資源以使用新名稱的政策。因此，當您重新命名使用者群組時，請務必謹慎。在重新命名使用者群組之前，必須手動檢查所有政策，以尋找按名稱提及該使用者群組的任何政策。例如，假設 Bob 是組織測試部分的經理。Bob 擁有政策連接到他的 IAM 使用者實體，可讓他從測試使用者群組中新增和移除使用者。如果管理員更改使用者群組的名稱 (或更改群組的路徑)，則管理員還需更新連接到 Bob 的政策以使用新名稱或新路徑。否則，Bob 將無法在該使用者群組中新增或移除使用者。

尋找將使用者群組參考為資源的政策：

1. 從 IAM 主控台的導覽窗格中，選擇 Policies (政策)。
2. 以 Type (類型) 資料欄排序，尋找 Customer managed (客戶受管) 自訂政策。
3. 選擇要編輯的政策的政策名稱。
4. 選擇許可索引標籤，然後選擇摘要。
5. 從服務清單中選擇 IAM，如果存在。
6. 在 Resource (資源) 欄位中尋找使用者群組的名稱。

7. 選擇編輯，可在政策中變更您的使用者群組名稱。

變更 IAM 使用者群組名稱

執行下列任何一項：

- AWS Management Console：在導覽窗格中，選擇 User groups (使用者群組)，然後選擇群組名稱。選擇 Edit (編輯)。輸入新的使用者群組名稱，然後選擇 Save changes (儲存變更)。
- AWS CLI：aws iam update-group
- AWS API：UpdateGroup

刪除 IAM 使用者群組

當您刪除中的使用者群組時 AWS Management Console，主控台會自動移除所有群組成員、卸離所有連結的受管理策略，以及刪除所有內嵌政策。不過，由於 IAM 不會自動刪除將該使用者群組作為資源參考的政策，因此刪除使用者群組時必須小心。在刪除使用者群組之前，必須手動檢查所有政策，以尋找按名稱提及該群組的任何政策。例如，測試團隊管理員 John 擁有與其 IAM 使用者實體相連的政策，可讓他從測試使用者群組中新增和移除使用者。如果管理員刪除該群組，則管理員還必須刪除連接到 John 的政策。否則，如果管理員重新建立已刪除的群組，並為其指定相同的名稱，則 John 的許可將保持不變，即使他已離開測試團隊。

尋找將使用者群組參考為資源的政策

1. 從 IAM 主控台的導覽窗格中，選擇 Policies (政策)。
2. 以 Type (類型) 資料欄排序，尋找 Customer managed (客戶受管) 自訂政策。
3. 選擇要刪除的政策的政策名稱。
4. 選擇許可索引標籤，然後選擇摘要。
5. 從服務清單中選擇 IAM，如果存在。
6. 在 Resource (資源) 欄位中尋找使用者群組的名稱。
7. 選擇刪除可刪除政策。
8. 輸入政策名稱以確認刪除政策，然後選擇刪除。

相反地，當您使用 AWS CLI PowerShell、Windows 工具或 AWS API 刪除使用者群組時，必須先移除群組中的使用者。然後刪除內嵌在該使用者群組的所有內嵌政策。接下來，分開連接到該群組的所有受管政策。只有這樣才能刪除使用者群組本身。

刪除 IAM 使用者群組 (主控台)

您可以從 AWS Management Console中刪除 IAM 使用者群組。

刪除 IAM 使用者群組 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 User groups (使用者群組)。
3. 在使用者群組清單中，選取要刪除的使用者群組名稱旁的核取方塊。您可以使用搜尋方塊，依類型、許可和使用者群組名稱來篩選使用者群組清單。
4. 選擇 Delete (刪除)。
5. 在確認方塊中，如果您要刪除單一使用者群組，請輸入使用者群組名稱，然後選擇 Delete (刪除)。如果您想刪除多個使用者群組，請輸入要刪除的使用者群組數量，後接 **`user groups`**，然後選擇 Delete (刪除)。例如，如果您要刪除三個使用者群組，請輸入 **`3 user groups`**。

刪除 IAM 使用者群組 (AWS CLI)

您可以從 AWS CLI中刪除 IAM 使用者群組。

刪除 IAM 使用者群組 (AWS CLI)

1. 從使用者群組中移除所有使用者。
   - aws iam 獲取組（以獲取用戶組中的用戶列表）和 aws iam remove-user-from-group（從用戶組中刪除用戶）
2. 刪除內嵌在該使用者群組的所有內嵌政策。
   - aws iam list-group-policies（獲取用戶組的內聯政策列表）和 aws iam delete-group-policy（刪除用戶組的內聯政策）
3. 分開到該使用者群組的所有受管政策。
   - aws iam list-attached-group-policies（獲取附加到用戶組的受管政策列表）和 aws iam detach-group-policy（從用戶組中分離受管政策）
4. 刪除使用者群組
   - aws iam delete-group

刪除身分與存取權管理使用者群組AWS

您可以使用 AWS API 刪除 IAM 使用者群組。

若要刪除 IAM 使用者群組 (AWS API)

1. 從使用者群組中移除所有使用者。
   - GetGroup（以獲取用戶組中的用戶列表）和 RemoveUserFromGroup（從用戶組中刪除用戶）
2. 刪除內嵌在該使用者群組的所有內嵌政策。
   - ListGroupPolicies(取得使用者群組內嵌政策的清單) 和 DeleteGroupPolicy(刪除使用者群組的內嵌政策)
3. 分開到該使用者群組的所有受管政策。
   - ListAttachedGroupPolicies(取得附加至使用者群組的受管理策略清單) 和 DetachGroupPolicy(從使用者群組中斷連結受管理的策略)
4. 刪除使用者群組
   - DeleteGroup

IAM 角色

IAM 角色是您可以在帳戶中建立的另一種 IAM 身分，具有特定的許可。IAM 角色類似於 IAM 使用者，因為它是具有權限政策的 AWS 身分，可決定身分可以執行和不能在其中執行的操作 AWS。但是，角色的目的是讓需要它的任何人可代入，而不是單獨地與某個人員關聯。此外，角色沒有與之關聯的標準長期憑證，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。

您可以使用角色將存取權委派給通常無法存取 AWS 資源的使用者、應用程式或服務。例如，您可能想要授與 AWS 帳戶中的使用者存取他們通常沒有的資源，或授與使用者 AWS 帳戶 存取另一個帳戶中資源的權限。或者，您可能希望允許移動應用程序使用 AWS 資源，但不想在應用程序中嵌入 AWS 密鑰（這些密鑰可能難以更新，並且用戶可以在其中提取它們）。有時候，您想要將 AWS 存取權授予已在以外定義身分的使用者 AWS，例如在您的公司目錄中定義的使用者。或者，您可能需要向第三方授予存取您帳戶的許可，讓他們可以對您的資源執行稽核。

對於這些案例，您可以使用 IAM 角色委派對 AWS 資源的存取權。本節介紹各種角色和它們的不同使用方式，如何從不同方式中選出適合的時機與方法，以及如何建立、管理、切換到 (或擔任) 和刪除角色。

> **Note**
>
> 第一次創建時 AWS 帳戶，默認情況下不會創建任何角色。為帳戶新增服務時，這些服務可能會新增服務連結角色，以支援其使用案例。
> 服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
> 在您刪除服務連結角色之前，您必須先刪除這些角色的相關資源。這可保護您的資源，避免您不小心移除資源的存取許可。
> 如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

主題

角色術語和概念

以下是一些基本術語，可協助您開始使用角色。

角色

您可以在帳戶中建立的 IAM 身分具有特定的許可。IAM 角色與 IAM 使用者有些相似處。角色和使用者都是具備許可政策的 AWS 身分，可決定身分在 AWS中可執行和不可執行的操作。但是，角色的目的是讓需要它的任何人可代入，而不是單獨地與某個人員關聯。此外，角色沒有與之關聯的標準長期憑證，例如密碼或存取金鑰。反之，當您擔任角色時，其會為您的角色工作階段提供臨時安全性憑證。

角色可由以下項目使用：

- 與角色相同 AWS 帳戶 的 IAM 使用者

- 與角色不同 AWS 帳戶 的 IAM 使用者
- 由 AWS 如 Amazon 彈性計算雲（亞馬遜 EC2）提供的 Web 服務
- 透過外部身分提供者 (IdP) 服務進行驗證的外部使用者，並且該服務與 SAML 2.0 或 OpenID Connect 或自訂建置的身分經紀人相容。

AWS 服務角色

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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。

AWS EC2 執行個體的服務角色

應用程式在 Amazon EC2 執行個體上執行的特殊類型服務角色可代為在您的帳戶中執行動作。此角色會在啟動時指派至 EC2 執行個體。在該執行個體上執行的應用程式可取得臨時安全憑證，並執行允許該角色執行的動作。如需有關使用 EC2 執行個體的服務角色的詳細資訊，請參閱 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可。

AWS 服務連結角色

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

> **Note**
>
> 如果當服務開始支援服務連結的角色時您已經在使用服務，您可能會收到一封電子郵件，宣告您帳戶中的新角色。在這種情況下，服務會自動在您的帳戶中建立服務連結的角色。您不需要採取任何動作來支援此角色，而且您不應手動刪除它。如需詳細資訊，請參閱 顯示在我的 AWS 帳戶中的新角色。

如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。如需詳細資訊，請參閱 使用服務連結角色。

角色鏈結

角色鏈結是指當您使用角色透過 AWS CLI 或 API 擔任第二個角色時。例如，`RoleA` 擁有擔任 `RoleB` 的許可。您可以透過在 AssumeRole API 作業中使`RoleA`用其長期使用者憑證來啟用 User1 假設。這會傳回 `RoleA` 短期憑證。透過角色鏈結，您可以使用 `RoleA` 的短期憑證，啟用 User1 擔任 `RoleB`。

當您擔任角色時，您可以傳遞工作階段標籤，並將標籤設為可轉移。可轉移工作階段標籤會傳遞到角色鏈中所有的後續工作階段。若要進一步了解工作階段標籤，請參閱 傳遞工作階段標籤 AWS STS。

角色鏈結可將您的 AWS CLI 或 AWS API 角色工作階段限制為最多一小時。當您使用 AssumeRoleAPI 作業來擔任角色時，您可以使用DurationSeconds參數指定角色工作階段的持續時間。您可以指定參數值高達 43200 秒 (12 小時)，這取決於您角色的最大工作階段持續時間設定 。然而，如果在您使用角色鏈結來擔任角色時，並提供大於一小時的 DurationSeconds 參數值，則操作失敗。

AWS 不會將使用角色授予在 EC2 執行個體上執行的應用程式的許可作為角色鏈結。

委派

授予許可給某人，以允許存取您控制的資源。委派涉及設定兩個帳戶間的信任。第一個是擁有資源的帳戶 (信任帳戶)。第二個是包含需要存取資源使用者的帳戶 (受信任帳戶)。受信任帳戶和信任帳戶可以是以下任何一個：

- 相同帳戶。
- 都在您的組織的控制之下的個別帳戶。
- 由不同組織擁有的兩個帳戶。

若要委派許可來存取資源，您要在具有連接兩個政策的信任帳戶中建立 IAM 角色。許可政策會授予角色的使用者所需的許可，以對資源執行預定的任務。信任政策會指定允許哪些受信任帳戶成員可擔任角色。

建立信任政策時，您無法在主體元素中指定萬用字元 (*) 做為 ARN 的一部分。信任政策會連接到信任帳戶中的角色，並且是許可的二分之一。另外一半是連接到受信任帳戶中使用者的許可政策，其允許切換為該使用者或擔任該角色 。擔任角色的使用者會暫時放棄其自己的許可，改為接受該角色的許可。當使用者退出或停止使用該角色時，會恢復原有的使用者許可。其他稱為外部 ID 的參數有助於確保在不是由相同組織控制的帳戶之間安全使用角色。

聯合

建立外部身分識別提供者與之間的信任關係 AWS。用戶可以登錄到 OIDC 提供商，例如使用亞馬遜，臉書，谷歌或任何與 OpenID Connect（OIDC）兼容的 IdP 登錄。使用者也可以登入企業身分系統，該系統與安全性聲明標記語言 (SAML) 2.0 相容，例如 Microsoft Active Directory Federation Services。當您使用 OIDC 和 SAML 2.0 設定這些外部身分識別提供者之間的信任關係時 AWS，會將使用者指派給 IAM 角色。使用者也會收到允許使用者存取您 AWS 資源的臨時認證。

聯合身分使用者

您可以使用企業使用者目錄或 OIDC 提供者的現有身分，而不是建立 IAM 使用者。 AWS Directory Service這些稱為聯合使用者。 AWS 透過身分識別提供者要求存取時，會將角色指派給聯合身分使用者。如需有關聯合身分使用者的詳細資訊，請參閱 聯合身分使用者和角色。

信任政策

JSON 政策文件，您會在其中定義您信任擔任角色的主體。角色信任政策是在 IAM 中連接至角色的以資源為基礎的必要政策。您在信任政策中可指定的主體包含使用者、角色、帳戶和服務。

許可政策

使用 JSON 格式的許可文件，您會在其中定義角色可以使用哪些動作和資源。文件的撰寫會根據 IAM 政策語言的規則。

許可界限

一種進階功能，可供您使用政策，限制以身分為基礎的政策可以授予角色的最大許可。您不能將許可界限用到服務連結的角色。如需詳細資訊，請參閱 IAM 實體的許可界限。

Principal

中 AWS 可執行動作和存取資源的實體。主體可以是 IAM AWS 帳戶根使用者使用者或角色。您可以授予許可以兩種方法之一來存取資源：

- 您可以將許可政策連接到使用者 (直接或間接透過群組) 或角色。
- 對於那些支援資源類型政策的服務，您可以在連接到資源之政策的 `Principal` 元素中識別主體。

如果您引用 AWS 帳戶 作為主體，則通常指該帳戶內定義的任何本金。

> **Note**
>
> 您不能在角色的信任政策中使用萬用字元 (*) 來比對部分主體名稱或 ARN。如需詳細資訊，請參閱 AWS 政策元素：Principal。

跨帳戶存取的角色

將一個帳戶中的資源存取權，授予不同帳戶中受信任主體的角色。角色是授予跨帳戶存取的主要方式。不過，一些 AWS 服務允許您直接將政策連接到資源 (而不是使用角色做為代理)。這些策略稱為以資源為基礎的策略，您可以使用它們來授與主參與者對資源的另一個 AWS 帳戶 存取權。其中一些資源包括 Amazon Simple Storage Service (S3) 儲存貯體、S3 Glacier 保存庫、Amazon

Simple Notification Service (SNS) 主題以及 Amazon Simple Queue Service (SQS) 佇列。若要了解哪些服務支援以資源為基礎的政策，請參閱 AWS 與 IAM 搭配使用的服務。如需有關以資源為基礎的政策詳細資訊，請參閱 IAM 中的跨帳戶資源存取。

常見的角色方案：使用者、應用程式和服務

與大多數 AWS 功能一樣，您通常有兩種使用角色的方法：在 IAM 主控台中以互動方式，或以程式設計方式使用 Windows PowerShell 適用的工具或 API。 AWS CLI

- 使用 IAM 主控台的帳戶中的 IAM 使用者可以切換到角色，以臨時使用主控台中角色的許可。使用者放棄其原始許可並取得指派給該角色的許可。當使用者退出角色時，將恢復其原始許可。
- 提供的應用程式或服務 AWS (例如 Amazon EC2) 可以透過請求用於向其發出程式設計請求的角色請求臨時安全登入資料來擔任角色。 AWS您以這種方式使用角色，這樣您就不必為需要存取資源的每個實體分享或維護長期安全憑證 (例如，透過建立 IAM 使用者)。

> **Note**
>
> 本指南互換使用切換到角色和擔任角色字詞。

使用角色的最簡單方法是授予 IAM 使用者切換到您在自己或另一個 AWS 帳戶中建立之角色的許可。他們可以使用 IAM 主控台輕鬆切換角色，以使用您通常不希望他們擁有的許可，然後退出角色以放棄這些許可。這有助於防止意外存取或修改敏感資源。

如需角色的更複雜用途，例如授予存取應用程式和服務，或聯合身分外部使用者，您可以呼叫 `AssumeRole` API。這個 API 呼叫會傳回一組臨時憑證，應用程式可以在後續 API 呼叫中使用這些憑證。嘗試使用臨時憑證的動作只能透過相關的角色授予許可。應用程式不需要像主控台中的使用者般「退出」角色；相反，應用程式只是停止使用臨時憑證並繼續使用原始憑證進行呼叫。

同盟使用者使用身分識別提供者 (IdP) 的認證登入。 AWS 然後提供臨時認證給受信任的 IdP，以傳遞給用戶以包含在後續 AWS 資源請求中。這些憑證提供授予指定角色的許可。

本節概述以下案例：

- 在您擁有的 IAM 使用者中為 IAM 使用者提供存取權，以存取您擁有的另一 AWS 帳戶 個帳戶中的資源
- 提供對非 AWS 工作負載的存取權
- 將存取權提供給第三方擁有之 AWS 帳戶 中的 IAM 使用者

- 提供對資 AWS 源所 AWS 提供服務的存取
- 將存取權提供給外部驗證使用者 (聯合身分)

在您擁有的另一個 AWS 帳戶 IAM 使用者中提供存取權

您可以授與 IAM 使用者切換到您的角色 AWS 帳戶 或您擁有的其他角色中定義 AWS 帳戶 的角色的權限。

> **Note**
>
> 如果要授予對您未擁有或無法控制的帳戶的存取許可，請參閱本主題後面的 提供訪問由第三方 AWS 帳戶 擁有。

假設您擁有一個對組織來說至關重要的 Amazon EC2 執行個體。您可以使用這些許可來建立角色，而非直接授予使用者終止執行個體的許可。然後，允許管理員可以在需要終止執行個體時切換為該角色。這麼做，可為執行個體加入以下幾層保護：

- 您必須向使用者明確授予擔任該角色的許可。
- 您的使用者必須使用或 API 主動切換至角色， AWS Management Console 或使用 AWS CLI 或 AWS API 擔任該角色。
- 您可以為角色加入多重要素驗證 (MFA) 保護，僅限登入 MFA 裝置的使用者才能擔任該角色。若要了解如何配置角色以使擔任角色的使用者必須先使用多重要素驗證 (MFA) 進行身分驗證，請參閱 設定受 MFA 保護的 API 存取。

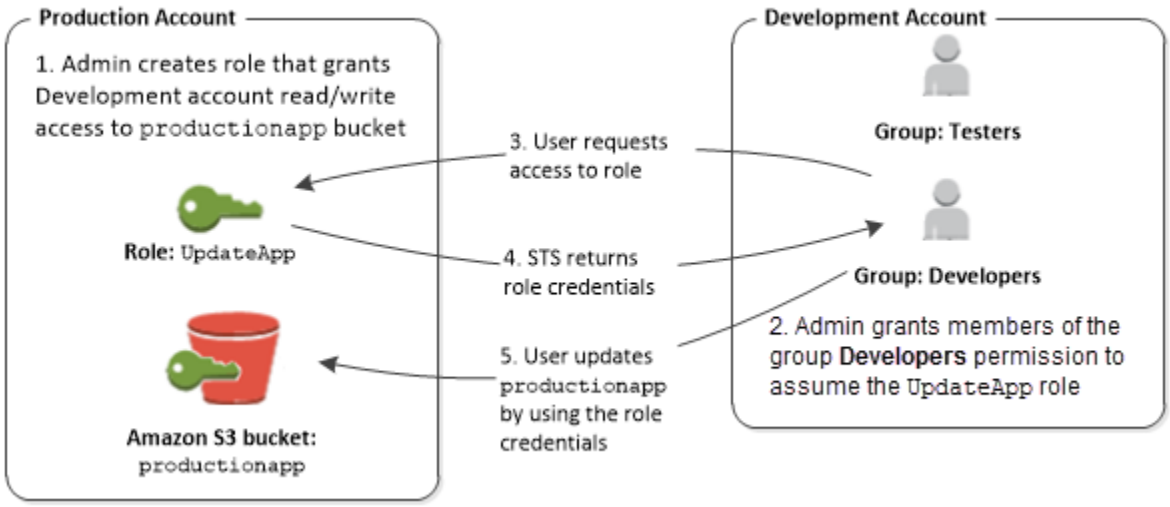
我們建議使用此方法強制實施最低權限。也就是僅限於特定任務需要時，才能使用升級的許可。藉由角色，您可以幫助防止意外更改敏感環境，如果您將它們與審核合併以協助確保僅在需要時才使用角色，將會有極大幫助。

在您出於此目的建立角色時，可在該角色的信任政策的 `Principal` 元素中依照 ID 指定其使用者需要存取許可的帳戶。隨後可以向這些其他帳戶中的特定使用者授予切換到角色的許可。若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。

一個帳戶中的使用者可以切換為相同或不同帳戶中的角色。使用角色過程中，使用者只能執行角色允許的操作並且只能存取角色允許的資源；其原始使用者許可處於暫停狀態。使用者退出角色時，恢復原始使用者許可。

使用不同的開發和生產帳戶的範例方案

想像一下，您的組織有多個可 AWS 帳戶 以將開發環境與生產環境隔離開來。開發帳戶中的使用者有時可能需要存取生產帳戶中的資源。例如在將更新從開發環境推廣到生產環境時，可能就需要跨帳戶存取許可。儘管您可以為在兩個帳戶中工作的使用者建立單獨的身分 (和密碼)，多個帳戶的憑證管理還是會為身分管理帶來難題。在以下圖表中，所有使用者都透過開發帳戶進行管理，但部分開發人員需要對生產帳戶進行有限存取。開發帳戶有兩個群組：測試人員和開發人員，每個群組有其專屬的政策。



1. 生產帳戶中的一名管理員使用 IAM 在該帳戶中建立 UpdateApp 角色。在角色中，管理員定義信任政策，該政策將開發帳戶指定為 Principal，這表示開發帳戶中的授權使用者可以使用 UpdateApp 角色。管理員也可以為角色定義許可政策，該政策指定名為 productionapp 之 Amazon S3 儲存貯體的讀取和寫入許可。

   然後，管理員將與需要擔任角色的任何人共用該角色的帳號和名稱。該資訊是角色的帳號和名稱 (針對 AWS 主控台使用者) 或 Amazon 資源名稱 (ARN) (用於 AWS CLI 或 AWS API 存取)。角色 ARN 類似於 arn:aws:iam::123456789012:role/UpdateApp，其中角色名為 UpdateApp，而且角色使用帳戶號碼 123456789012 所建立。

   > **Note**
   >
   > 管理員可以選擇是否配置角色，以便擔任角色的使用者必須先使用多重要素驗證 (MFA) 進行身分驗證。如需詳細資訊，請參閱 設定受 MFA 保護的 API 存取。

2. 在開發帳戶中，管理員向開發人員群組的成員授予切換為角色的許可。這是通過授予開發人員組調用UpdateApp角色的 AWS Security Token Service （ AWS STS ）AssumeRoleAPI 的權限來完成的。開發帳戶中的開發人員群組的所有 IAM 使用者現在都可以切換為生產帳戶中的 UpdateApp 角

色。不在開發人員群組中的其他使用者無權切換為該角色，因此無法存取生產帳戶中的 S3 儲存貯體。

3. 使用者請求切換為該角色：
   - AWS 控制台：用戶在導航欄上選擇帳戶名稱，然後選擇切換角色。使用者指定帳戶 ID (或別名) 和角色名稱。或者，使用者可以按一下管理員在電子郵件中發送的連結。透過該連結，使用者可以前往已填寫詳細資訊的 Switch Role (切換角色) 頁面。
   - AWS API/AWS CLI：開發帳戶「開發人員」群組中的使用者呼叫`AssumeRole`函數以取得角色的`UpdateApp`認證。使用者將 `UpdateApp` 角色的 ARN 指定為呼叫的一部分。如果測試人員群組中的使用者發出相同請求，請求將失敗，因為測試人員沒有針對 `AssumeRole` 角色 ARN 呼叫 `UpdateApp` 的許可。
4. AWS STS 返回臨時憑據：
   - AWS 控制台：使用角色的信任策略 AWS STS 驗證請求，以確保請求來自受信任的實體（它是：開發帳戶）。驗證後，將暫時的安全登入資料 AWS STS傳回主 AWS 控台。
   - API/CLI：根據角色的信任策略 AWS STS 驗證請求，以確保請求來自受信任的實體（它是：開發帳戶）。驗證後，將臨時安全登入資料 AWS STS傳回應用程式。
5. 臨時認證允許訪問資 AWS 源：
   - AWS 控制台： AWS 控制台代表用戶使用臨時憑據進行所有後續控制台操作（在此情況下）讀取和寫入`productionapp`值區。主控台無法存取生產帳戶中的任何其他資源。使用者退出角色時，使用者的許可恢復為切換為角色之前所擁有的原始許可。
   - API/CLI：應用程式使用臨時安全性憑證更新 `productionapp` 儲存貯體。應用程式只能使用臨時安全性憑證讀取和寫入 `productionapp` 儲存貯體，無法存取生產帳戶的任何其他資源。應用程式不必退出角色，只需在後續 API 呼叫中停止使用臨時憑證並使用原始憑證。

其他資訊

如需詳細資訊，請參閱下列內容：

- IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶

提供對非 AWS 工作負載的存取權

IAM 角色是指派許可的物件 AWS Identity and Access Management (IAM)。當您假設該角色使用 IAM 身分或外部的身分時 AWS，它會為您提供角色工作階段的臨時安全登入資料。您可能在資料中心或其他基礎架構中執行的工作負載需 AWS 要存取您的 AWS 資源。您可以使用「隨處角色」(IAM AWS Identity and Access Management 角色隨處) 來驗證非 AWS 工作負載，而不必建立、分發和管理長期

存取金鑰。IAM 角色 Anywhere 使用憑證授權單位 (CA) 提供的 X.509 憑證來驗證身分，並透過 IAM 角色提供 AWS 服務 的臨時登入資料安全地提供存取權。

若要使用 IAM Roles Anywhere，您可使用 AWS Private Certificate Authority 設定 CA，或者使用您 PKI 基礎設施中的 CA。設定 CA 後，您可以在 IAM Roles Anywhere 中建立名為信任錨點的物件，以便在 IAM Roles Anywhere 與 CA 之間建立信任，從而進行身分驗證。然後您可以設定現有的 IAM 角色，或建立信任 IAM Roles Anywhere 服務的新角色。當您的非 AWS 工作負載使用信任錨點使用 IAM 角色隨處進行身份驗證時，他們可以為您的 IAM 角色取得臨時登入資料以存取您的 AWS 資源。

如需設定 IAM Roles Anywhere 的詳細資訊，請參閱《IAM Roles Anywhere 使用者指南》中的 What is AWS Identity and Access Management Roles Anywhere (什麼是 IAM Roles Anywhere)。

提供訪問由第三方 AWS 帳戶 擁有

當第三方需要您組織 AWS 資源的存取權時，您可以使用角色將存取權委派給他們。例如，第三方可能提供一種用於管理您的 AWS 資源之服務。使用 IAM 角色，您可以授與這些第三方存取您的 AWS 資源，而無需共用您的 AWS 安全登入資料。相反地，第三方 AWS 可以透過假設您在 AWS 帳戶. 若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。

為了建立他們可以代入的角色，第三方必須為您提供以下資訊：

- 第三方的 AWS 帳戶 識別碼。當您定義角色的信任原則時，您可以指定其 AWS 帳戶 識別碼作為主參與者。
- 與角色唯一關聯的外部 ID。外部 ID 可以是只有您和第三方知道的任何識別碼。例如，您可以使用您與該第三方之間的發票 ID，但不要使用能被猜到的內容，例如第三方的電話號碼。為角色定義信任政策時，必須指定該 ID。第三方在代入角色時必須提供該 ID。如需有關外部 ID 的詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。
- 協力廠商使用您的 AWS 資源所需的權限。定義角色的許可政策時，必須指定這些許可。這個政策定義了他們可以執行哪些操作以及可以存取哪些資源。

建立完角色後，您必須向第三方提供該角色的 Amazon Resource Name (ARN)。他們需要使用您的角色的 ARN 來代入該角色。

> **Important**
>
> 當您授與第三方存取您的 AWS 資源時，他們可以存取您在策略中指定的任何資源。他們使用的資源費用將由您支付。請確保適當地限制他們對資源的使用。

將 AWS 資源存取權授予第三方時，如何使用外部 ID

有時，您需要授予第三方訪問您的 AWS 資源（委託訪問權限）。此方案的一個重要層面是外部 ID，外部 ID 是一條可選資訊，您可在 IAM 角色信任政策中使用該資訊來指定誰能擔任該角色。

> **Important**
>
> AWS 不會將外部 ID 視為密碼。在中建立密碼 (例如存取 key pair 或密碼) 之後 AWS，您將無法再次檢視它們。有權查看角色的任何人都可以看到該角色的外部 ID。

在您支援具有不同 AWS 帳戶的多重租用戶環境中，我們建議您每 AWS 帳戶個使用一個外部 ID。此 ID 應該是由第三方產生的隨機字串。

如要請求第三方在取得角色時提供外部 ID，請使用您選擇的外部 ID 來更新角色的信任政策。

若要在擔任角色時提供外部 ID，請使用 AWS CLI 或 AWS API 來擔任該角色。如需詳細資訊，請參閱 STS AssumeRoleAPI 作業或 STS 假設角色 CLI 作業。

例如，假設您決定聘請名為 Example Corp 的第三方公司來監控您的 AWS 帳戶 並幫助優化成本。為了追蹤您的每日支出，Example Corp 需要存取您的 AWS 資源。Example Corp 也可監控其他客戶的許多其他 AWS 帳戶。

請不要向 Example Corp 提供對您 AWS 帳戶中的 IAM 使用者和其長期憑證的存取權。請改用 IAM 角色及其臨時安全性憑證。IAM 角色提供了一種機制，可讓第三方存取您的 AWS 資源，而無需共用長期登入資料 (例如 IAM 使用者存取金鑰)。

您可以使用 IAM 角色在您 AWS 帳戶 和範例公司帳戶之間建立信任關係。建立此關係後，Example Corp 帳戶的成員可以呼叫 AWS Security Token Service AssumeRoleAPI 以取得臨時安全登入資料。然後，Example Corp 成員可以使用認證來存取您帳戶中的 AWS 資源。

> **Note**
>
> 如需有關您可呼叫以取得臨時安全性登入資料之 AssumeRole 和其他 AWS API 作業的詳細資訊，請參閱請求暫時性安全憑證。

以下是此方案的更多詳細資訊：

1. 您聘請了 Example Corp，此公司將為您建立獨有的客戶識別碼。他們為您提供了這個唯一的客戶 ID 和他們的 AWS 帳戶 號碼。您需要此資訊來在下一個步驟中建立 IAM 角色。

> **Note**
>
> 實施例公司可以使用他們想要的任何字符串值 ExternalId，只要它是唯一的每個客戶。該值可以是客戶帳戶，甚至可以是一個隨機字串，只要沒有兩個客戶擁有相同的值即可。該值不是「機密」。實施例公司必須提供 ExternalId 價值給每個客戶。關鍵在於，該值必須由 Example Corp 而非由其客戶產生，以確保每個外部 ID 都是唯一的。

2. 您可以登入 AWS 並建立 IAM 角色，讓範例公司存取您的資源。與任何 IAM 角色類似，該角色具有兩個政策：許可政策和信任政策。角色的信任政策指定擔任該角色的對象。在我們的範例案例中，原則會指定範例公司的 AWS 帳戶 數目`Principal`。這允許來自此帳戶的身分擔任該角色。此外，您新增 `Condition` 元素到信任政策。此 `Condition` 測試 `ExternalId` 內容索引鍵，以確保它與 Example Corp 的獨有客戶 ID 一致。例如：

```
   "Principal": {"AWS": "Example Corp's AWS ## ID"},
   "Condition": {"StringEquals": {"sts:ExternalId": "Unique ID Assigned by Example
Corp"}}
```

3. 該角色的許可政策指定該角色允許某個人執行哪些操作。例如，您可以指定該角色允許某人只能管理您的 Amazon EC2 和 Amazon RDS 資源，但不能管理您的 IAM 使用者或群組。在我們的範例方案中，您使用許可政策為 Example Corp 授予帳戶中的所有資源的唯讀存取許可。
4. 建立完角色後，為 Example Corp 提供該角色的 Amazon Resource Name (ARN) (ARN)。
5. 當實施例公司需要訪問您的 AWS 資源時，公司的某人調用 AWS `sts:AssumeRole` API。呼叫包括要承擔之角色的 ARN，以及對應至其客戶 ID 的 ExternalId 參數。

如果請求來自使用 Example Corp 的某個人 AWS 帳戶，並且角色 ARN 和外部 ID 是正確的，則請求成功。然後，它會提供 Example Corp 可用來存取角色允許的 AWS 資源的臨時安全登入資料。

換言之，當角色政策包括外部 ID 時，任何需要擔任該角色的人都必須是該角色中的主體，還必須包括正確的外部 ID。

為什麼要使用外部 ID？

抽象地說，外部 ID 允許正擔任該角色的使用者聲明所操作的環境。它還為帳戶擁有者提供一種方法來允許僅在特定情況下擔任該角色。外部 ID 的主要功能是解決並防止 混淆代理人問題。

我何時應使用外部 ID？

在以下情況下使用外部 ID：

- 您是 AWS 帳戶 擁有者，並且已為第三方設定角色，該第三方除了可存取您的其他人 AWS 帳戶 之外的角色。您應要求第三方提供其在擔任您的角色時包含的外部 ID。然後，在您角色的信任政策中檢查該外部 ID。這樣做可確保外部方僅在代表您執行操作時才能擔任您的角色。
- 在前述情況下，您代表不同客戶 (如 Example Corp) 擔任角色。您應該為每個客戶分配一個唯一的外部 ID 並指導他們將該外部 ID 加入到其角色的信任政策。然後，您必須確保在代入角色的請求中始終包含正確的外部 ID。

  您可能已為您的每個客戶提供一個獨有識別碼，而且此獨有 ID 足以用作外部 ID。該外部 ID 不是您要明確建立或分別追蹤所需的特殊值 (僅用於此目的)。

  您應始終在您的 `AssumeRole` API 呼叫中指定外部 ID。此外，在客戶為您提供角色 ARN 時，請測試是否能在含有/不含有正確外部 ID 的情況下擔任該角色。如果可在沒有正確外部 ID 的情況下擔任角色，則請勿在您的系統中儲存該客戶的角色 ARN。等待該客戶將角色信任政策更新為要求提供正確的外部 ID。這樣一來，您協助您的客戶執行了正確的操作，並幫助您和客戶避免了混淆代理人問題。

提供對 AWS 服務的存取

許多 AWS 服務都要求您使用角色來控制該服務可以存取的內容。服務會擔任代您執行動作的角色稱為服務角色。當角色做為服務的專業用途時，它被歸類為 EC2 執行個體的服務角色或服務連結角色。請參閱每個服務的 AWS 文件，以查看它是否使用角色，以及了解如何指派角色以供服務使用。

有關如何創建角色以委派對由提供的服務的訪問權限的更多內容 AWS，敬請參閱建立角色以將許可委派給 AWS 服務。

混淆代理人問題

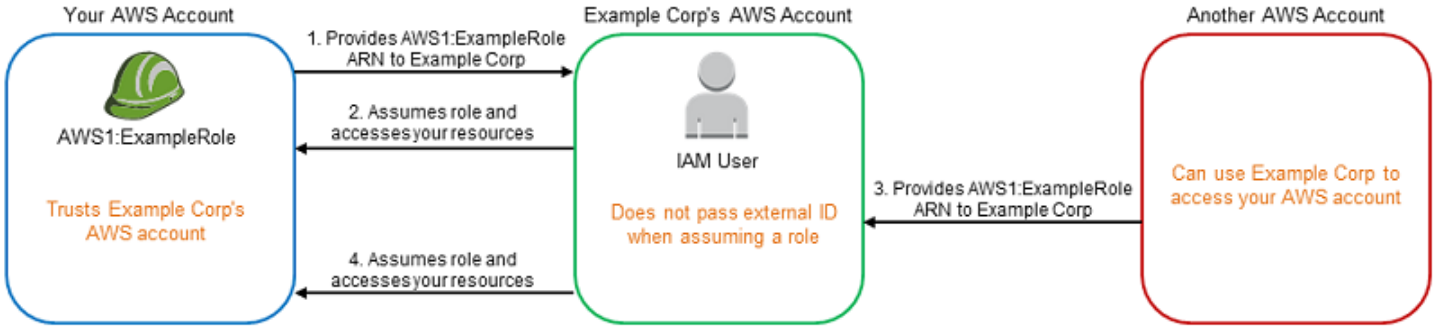
混淆代理人問題屬於安全性議題，其中沒有執行動作許可的實體可以強制具有更多許可的實體執行該動作。為了防止這種情況發生，在您 AWS 提供第三方（稱為跨帳戶）或其他 AWS 服務（稱為跨服務）訪問您帳戶中的資源時，提供幫助您保護帳戶的工具。

有時，您可能需要授予第三方對您 AWS 資源的訪問權限（委託訪問權限）。例如，假設您決定聘請名為 Example Corp 的第三方公司來監控您的 AWS 帳戶 並幫助優化成本。為了追蹤您的每日支出，Example Corp 需要存取您的 AWS 資源。實施例公司還監視許多其他客戶 AWS 帳戶 的其他客戶。您可以使用 IAM 角色在您 AWS 帳戶 和範例公司帳戶之間建立信任關係。此方案的一個重要層面是外部 ID，外部 ID 是一條可選資訊，您可在 IAM 角色信任政策中使用該資訊來指定誰能擔任該角色。外部 ID 的主要功能是解決並防止「混淆代理人」問題。

在中 AWS，跨服務模擬可能會導致混淆的副問題。在某個服務 (呼叫服務) 呼叫另一個服務 (被呼叫服務) 時，可能會發生跨服務模擬。可以操縱呼叫服務來使用其許可，以其不應有存取許可的方式對其他客戶的資源採取動作。

預防跨帳戶混淆代理人

下圖說明了跨帳戶混淆代理人問題。



此案例假設如下：

- AWS 1 是你的 AWS 帳戶.
- AWS 1：ExampleRole是您帳戶中的角色。此角色的信任政策透過將 Example Corp 的 AWS 帳戶指定為可擔任該角色的帳戶來信任 Example Corp。

將發生以下情況：

1. 當您開始使用實施例公司的服務時，您將 AWS 1: 的 ARN 提供ExampleRole給示例公司
2. 範例公司使用該角色 ARN 取得臨時安全登入資料，以存取. AWS 帳戶這樣一來，您將信任 Example Corp 做為可代表您執行操作的「代理人」。
3. 另一位 AWS 客戶也開始使用實施例公司的服務，並且該客戶還提供了 AWS 1 的 ARN：ExampleRole例如公司使用。據推測，其他客戶學習或猜測了 AWS 1: ExampleRole，這不是一個秘密。
4. 當其他客戶要求 Example Corp 存取其帳戶中的 AWS 資源時，Example Corp 會使用 AWS 1: 存ExampleRole取您帳戶中的資源。

這就是其他客戶可對您的資源進行未授權存取的方式。由於此客戶能夠誘使 Example Corp 無意中操作您的資源，因此 Example Corp 現在是一個「混淆代理人」。

Example Corp 可以透過要求在角色的信任政策中包含 `ExternalId` 條件檢查來解決混淆代理人問題。Example Corp 為每個客戶生成唯一的 `ExternalId` 值，並在其請求中使用該值來擔任此角色。

因此，ExternalId 值必須在 Example Corp 的客戶中具備唯一性，並由 Example Corp 而非其客戶控制。這就是您從 Example Corp 取得該 ID 且不能自行提供該 ID 的原因。這樣可以防止 Example Corp 成為混淆的副手並授予對其他帳戶 AWS 資源的訪問權限。

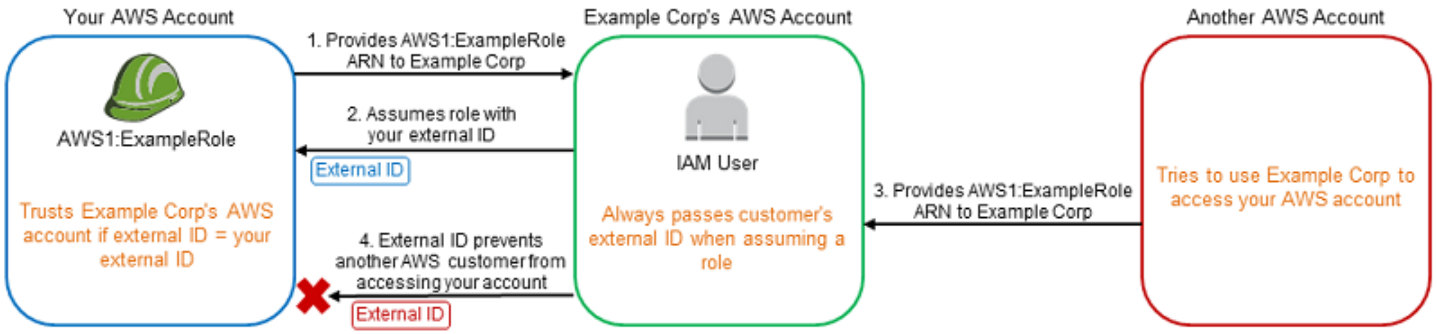
在我們的方案中，假設 Example Corp 為您提供的獨有識別碼是 12345，而為另一個客戶提供的識別碼是 67890。這些識別碼已針對此方案進行簡化。通常，這些識別碼為 GUID。假定這些識別碼在 Example Corp 的客戶之間是獨有的，它們將是用於外部 ID 的有意義的值。

Example Corp 將為您提供外部 ID 值 12345。然後，您必須將一個 Condition 元素加入到角色的信任政策，該政策要求 sts:ExternalId 值為 12345，如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "AWS": "Example Corp's AWS Account ID"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "12345"
      }
    }
  }
}
```

此原則中的條件元素允許範例公司只有在 AssumeRole API 呼叫包含 12345 的外部識別碼值時，才會擔任角色。示例公司確保每當它擔任代表客戶的角色時，它始終包含該客戶的外部 ID 值在 AssumeRole 呼叫中。即使其他客戶提供了示例公司與您的 ARN，它也無法控制示例公司在其請求中包含的外部 ID。 AWS這有助於防止未經授權的客戶取得對您的資源的存取權限。

下圖說明此程序。



1. 和以前一樣，當您開始使用實施例公司的服務時，您將 AWS 1: 的 ARN 提供ExampleRole給示例公司
2. 當範例公司使用該角色 ARN 承擔角色 AWS 1: 時ExampleRole，範例公司會在 AssumeRole API 呼叫中包含您的外部識別碼 (12345)。外部 ID 符合角色的信任原則，因此 AssumeRole API 呼叫成功，Example Corp 會取得臨時安全登入資料，以存取. AWS 帳戶
3. 另一位 AWS 客戶也開始使用 Example Corp 的服務，和以前一樣，該客戶還提供了 AWS 1 的 ARN：ExampleRole例如公司使用。
4. 但是這一次，當實施例公司試圖承擔角色 AWS 1: 時ExampleRole，它提供了與其他客戶（67890）關聯的外部 ID。其他客戶無法更改此外部 ID。Example Corp 這樣做是因為另一個客戶請求使用該角色，因此 67890 表示 Example Corp 正在其中操作的環境。因為您將具有自己外部識別碼 (12345) 的條件新增至信任原則 AWS 1: ExampleRole，因此 AssumeRole API 呼叫會失敗。該其他客戶不能對您帳戶中的資源進行未經授權的存取 (由圖表中的紅色 "X" 表示)。

該外部 ID 說明阻止任何其他客戶誘使 Example Corp 無意中存取您的資源。

預防跨服務混淆代理人

建議您在資源型政策中使用 aws:SourceArn、aws:SourceAccount、aws:SourceOrgID 或 aws:SourceOrgPaths 全域條件內容鍵，將服務具備的許可限定於特定資源。用於僅aws:SourceArn將一個資源與跨服務存取相關聯。用於aws:SourceAccount讓該帳號中的任何資源與跨服務使用相關聯。用於aws:SourceOrgID允許組織內任何帳號的任何資源與跨服務使用相關聯。用於aws:SourceOrgPaths將 AWS Organizations 路徑中帳號的任何資源與跨服務使用相關聯。如需有關使用和了解路徑的詳細資訊，請參閱了解 AWS Organizations 實體路徑。

防範混淆代理人問題的最精細的方法是在資源型政策中使用 aws:SourceArn 全域條件內容鍵和資源的完整 ARN。如果不知道資源的完整 ARN，或者如果您指定了多個資源，請使用 aws:SourceArn 全域條件內容鍵，同時使用萬用字元 (*) 表示 ARN 的未知部分。例如：。arn:aws:*servicename*:*:*123456789012*:*

如果 aws:SourceArn 值不包含帳戶 ID (例如 Amazon S3 儲存貯體 ARN)，您必須同時使用 aws:SourceAccount 和 aws:SourceArn 來限制許可。

若要大規模防範混淆代理人問題，請在資源型政策中使用 aws:SourceOrgID 或 aws:SourceOrgPaths 全域條件內容鍵和資源的組織 ID 或組織路徑。當您新增、移除或移動組織中的帳戶時，包含 aws:SourceOrgID 或 aws:SourceOrgPaths 鍵的政策將會自動包含正確的帳戶，您無需手動更新政策。

對於 non-service-linked 角色信任原則，信任原則中的每個服務都會執行iam:PassRole動作，以確認角色與呼叫服務位於相同的帳戶中。因此，不需要搭配這些信任政策使用 aws:SourceAccount、aws:SourceOrgID 或 aws:SourceOrgPaths。在信任政策中使用 aws:SourceArn 可讓您指定可擔任角色的資源，例如 Lambda 函數 ARN。某些新建立角色aws:SourceArn的使 AWS 服務 用aws:SourceAccount和信任原則，但帳戶中的現有角色並不需要使用金鑰。

> **Note**
>
> AWS 服務 與 AWS Key Management Service 使用 KMS 金鑰授權整合的不支援aws:SourceArn、aws:SourceAccountaws:SourceOrgID、或aws:SourceOrgPaths條件金鑰。如果 AWS 服務 透過 KMS 金鑰授權也使用金鑰，則在 KMS 金鑰原則中使用這些條件金鑰會導致非預期的行為。

跨服務混淆副預防 AWS Security Token Service

許多 AWS 服務需要您使用角色來允許服務代表您存取其他服務的資源。服務會擔任代您執行動作的角色稱為服務角色。角色需要兩個政策：指定允許承擔角色的主體的角色信任政策；以及指定角色可執行動作的許可政策。角色信任政策是 IAM 中唯一的資源型政策類型。其他人則 AWS 服務 具有以資源為基礎的政策，例如 Amazon S3 儲存貯體政策。

當服務代表您擔任角色時，必須允許服務主體執行角色信任政策中的 sts:AssumeRole 動作。當服務呼叫時sts:AssumeRole，會 AWS STS 傳回服務主體用來存取角色權限原則所允許之資源的一組暫時安全性登入資料。當服務在您的帳戶中擔任角色時，您可以在角色信任政策中包含 aws:SourceArn、aws:SourceAccount、aws:SourceOrgID 或 aws:SourceOrgPaths 全域條件內容鍵，以將對角色的存取限定於僅由預期資源產生的請求。

例如，在中 AWS Systems Manager Incident Manager，您必須選擇允許事件管理員代表您執行 Systems Manager 自動化文件的角色。自動化文件可以包含 CloudWatch 警示或事件所起始之 EventBridge事件的自動回應計畫。在下列角色信任政策範例中，您可以使用 aws:SourceArn 條件

鍵，以根據事件記錄的 ARN 限制對服務角色的存取。只有從回應計劃資源 myresponseplan 建立的事件記錄能夠使用此角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "Service": "ssm-incidents.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:ssm-incidents:*:111122223333:incident-
record/myresponseplan/*"
      }
    }
  }
}
```

> **Note**
>
> 並非所有服務都與 AWS STS 支援aws:SourceArnaws:SourceAccountaws:SourceOrgID、或aws:SourceOrgPaths條件金鑰整合。在具有不支援整合的 IAM 信任政策中，使用這些鍵可能會導致非預期行為。

對外部驗證的使用者提供存取權 (聯合身分)

您的使用者可能已在以外擁有身分識別 AWS，例如在您的公司目錄中。如果這些使用者需要使用 AWS 資源 (或使用存取這些資源的應用程式)，那麼這些使用者也需要 AWS 安全性認證。您可以使用 IAM 角色為從您的組織或第三方身分提供者 (IdP) 聯合身分的使用者指定許可。

> **Note**
>
> 作為安全最佳實務，建議您使用聯合身分在 IAM Identity Center 中管理使用者存取權，而不是建立 IAM 使用者。若要了解需要 IAM 使用者的特定情形，請參閱建立 IAM 使用者 (而非角色) 的時機。

使用 Amazon Cognito 聯合行動或以 Web 為基礎的應用程式的使用者

如果您建立可存取 AWS 資源的行動或基於 Web 的應用程式，應用程式需要安全認證才能向其發出程式設計要 AWS求。對於大多數行動應用程式藍本，建議您使用 Amazon Cognito。您可以將此服務與 iOS 版AWS 行動 SDK 以及適用於 Android 和 Fire OS 的AWS 行動 SDK 搭配使用，為使用者建立唯一身分，並對其進行驗證，以便安全存取您的 AWS 資源。Amazon Cognito 支援與下一個部分中列出的身分提供者相同的身分提供者，並且還支援開發人員驗證身分和未經身分驗證 (訪客) 的存取。Amazon Cognito 還提供 API 操作以同步使用者資料，如此可在使用者於不同裝置之間移動時進行保留。如需詳細資訊，請參閱 針對行動應用程式使用 Amazon Cognito。

使用公有身分服務提供者或 OpenID Connect 來聯合使用者

在可能的情況下，將 Amazon Cognito 用於行動和以 Web 為基礎的應用程式案例。Amazon Cognito 會為您完成大部分的 behind-the-scenes 工作與公有身分提供者服務。它適用於相同的第三方服務，並且還支援匿名登入。但是，對於更進階的案例，您可以直接使用 Login with Amazon、Facebook、Google 或與 OpenID Connect (OIDC) 相容的任何 IdP 等第三方服務。如需使用其中一項服務使用 OIDC 聯盟的詳細資訊，請參閱。OIDC 聯盟

使用 SAML 2.0 聯合使用者

如果您的組織已經使用支援 SAML 2.0 (安全性宣告標記語言 2.0) 的身分識別提供者軟體套件，您可以在您的組織之間建立信任，做為身分識別提供者 (IdP) 和 AWS 服務提供者。然後，您可以使用 SAML 為使用者提供呼叫 API 作業的聯合單一登入 (SSO) AWS Management Console 或聯合存取權。 AWS 例如，如果您的公司使用 Microsoft Active Directory 和 Active Directory Federation Services，則可以使用 SAML 2.0 聯合。如需有關使用 SAML 2.0 聯合使用者的詳細資訊，請參閱SAML 2.0 聯合身分。

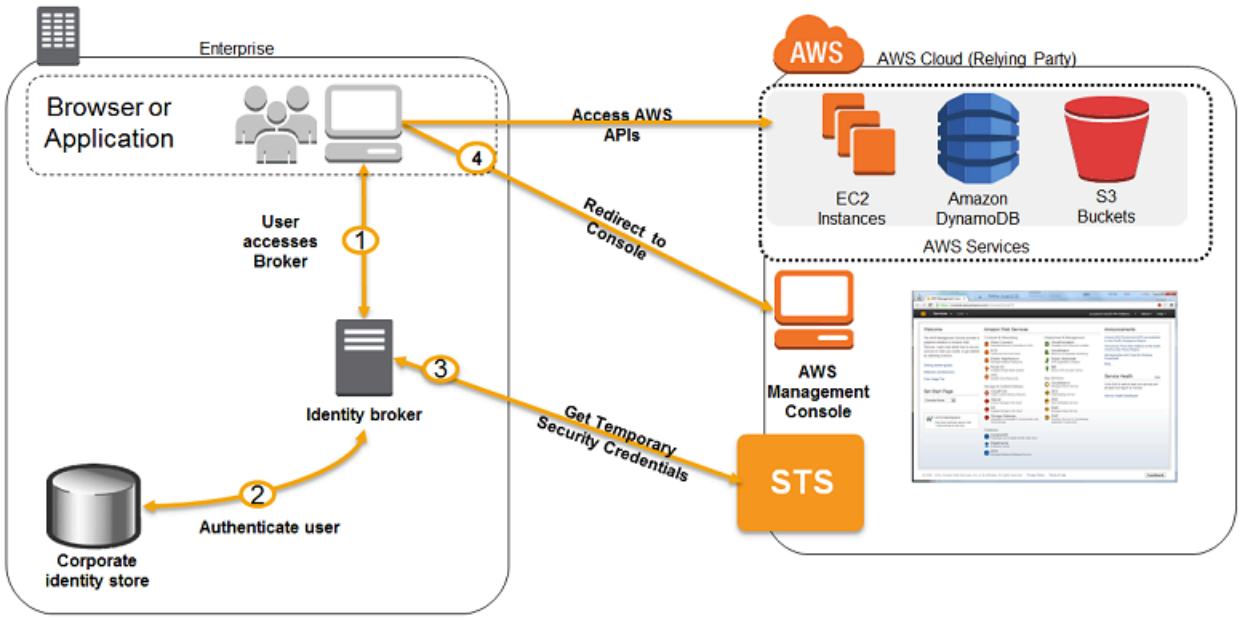
透過建立自訂身分經紀人應用程式來聯合使用者

如果您的身分存放區與 SAML 2.0 不相容，則可以建置自訂身分經紀人應用程式以執行類似的功能。Broker 應用程式會驗證使用者、要求使用者的臨時認證 AWS，然後將它們提供給使用者以存取 AWS 資源。

例如，Example Corp. 有許多員工需要執行存取公司 AWS 資源的內部應用程式。員工已經在公司身分和身分驗證系統中擁有身分，而 Example Corp. 不想要為每個公司員工建立個別的 IAM 使用者。

Bob 是範例公司的開發人員為了讓範例公司內部應用程式能夠存取公司的 AWS 資源，Bob 開發自訂的身分識別代理應用程式。該應用程式驗證員工是否已登入到現有的 Example Corp. 身分和身分驗證系統，該系統可能使用 LDAP、Active Directory 或其他系統。然後，身分經紀人應用程式取得員工的臨時安全憑證。此案例與前一個案例類似 (使用自訂驗證系統的行動應用程式)，不同之處在於需要存取 AWS 資源的應用程式全部在企業網路中執行，而且公司擁有現有的驗證系統。

若要取得臨時安全憑證，身分經紀人應用程式將呼叫 AssumeRole 或 GetFederationToken 以取得臨時安全憑證，具體取決於 Bob 想要如何管理使用者政策以及臨時憑證何時過期。(如需有關這些 API 操作間差異的詳細資訊，請參閱 IAM 中的暫時安全憑證 和 控制臨時安全安全憑證的許可)。該調用返回由訪問密鑰 ID，秘密 AWS 訪問密鑰和會話令牌組成的臨時安全憑據。身分經紀人應用程式讓這些臨時安全憑證可供內部公司應用程式使用。然後，應用程式可以使用臨時憑證直接呼叫 AWS 。該應用程式快取憑證，直到過期，然後請求一組新的臨時憑證。下圖說明此情況。



此案例具有以下屬性：

- 身分經紀人應用程式有權存取 IAM 的權杖服務 (STS) API 來建立臨時安全憑證。
- 身分經紀人應用程式能夠驗證員工是否在現有身分驗證系統中進行了身分驗證。
- 使用者可以取得暫時 URL，讓他們存取「 AWS 管理主控台」(稱為單一登入)。

如需建立臨時安全性憑證檔案的詳細資訊，請參閱 請求暫時性安全憑證。如需聯合身分使用者取得 AWS 管理主控台存取權的詳細資訊，請參閱啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console。

使用服務連結角色

服務連結角色是一種獨特的 IAM 角色類型，可直接連結到 AWS 服務。服務連結角色由服務預先定義，並包含服務代表您呼叫其他 AWS 服務所需的所有權限。連結的服務也定義您如何建立、修改和刪除服務連結的角色。服務可能會自動建立或刪除角色。做為服務中精靈或程序一部分，它也許可讓您建

立、修改或刪除角色。或者，它可能要求您使用 IAM 來建立或刪除角色。不論採用何種方式，服務連結角色可簡化設定服務流程，因為您不必手動新增服務許可，以代表您完成動作。

> **Note**
>
> 請記得，服務角色與服務連結角色不同。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。如需更多資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務。服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

連結的服務定義其服務連結角色的許可，除非另有定義，否則僅有該服務可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

在您刪除角色之前，您必須首先刪除它們的相關資源。這可保護您的資源，避免您不小心移除資源的存取許可。

> **Tip**
>
> 如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

服務連結角色許可

您必須設定 IAM 實體 (使用者或角色) 的許可，允許使用者或角色建立或編輯服務連結角色。

> **Note**
>
> 服務連結角色的 ARN 包括服務主體，這在下列政策中以 *SERVICE-NAME*.amazonaws.com 形式指出。請勿嘗試猜測服務主體，因為它區分大小寫，而且格式可能會因 AWS 服務而異。若要檢視服務的服務主體，請參閱該服務連結的角色文件。

允許 IAM 實體建立特定服務連結角色

將下列政策新增至需要建立服務連結角色的 IAM 實體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*",
            "Condition": {"StringLike": {"iam:AWSServiceName": "SERVICE-
NAME.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
        }
    ]
}
```

若要允許 IAM 實體建立任何服務連結角色

將下列陳述式新增至需要建立服務連結角色的 IAM 實體的許可政策，或包含所需政策的任何服務角色。此政策陳述式不允許 IAM 實體連接政策到角色。

```
{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

若要允許 IAM 實體編輯任何服務連結角色的說明

將下列陳述式新增至需要編輯服務連結角色說明或任何服務角色的 IAM 實體的許可政策。

```
{
    "Effect": "Allow",
    "Action": "iam:UpdateRoleDescription",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
```

```
}
```

若要允許 IAM 實體刪除特定服務連結角色

將下列陳述式新增至需要刪除服務連結角色的 IAM 實體的許可政策。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
}
```

允許 IAM 實體刪除任何服務連結角色

將下列陳述式新增至需要刪除服務連結角色 (但並非服務角色) 的 IAM 實體許可政策。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允許 IAM 實體將現有角色傳遞到服務

某些 AWS 服務可讓您將現有角色傳遞給服務，而不是建立新的服務連結角色。若要執行此操作，使用者必須擁有傳遞角色給服務的許可。在需要傳遞角色之 IAM 實體的許可政策中，新增下列陳述式。透過這個政策陳述式，實體也可以檢視角色清單，並從中選擇要傳遞的角色。如需詳細資訊，請參閱 授予使用者將角色傳遞至 AWS 服務的許可。

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": "*"
```

```
},
{
  "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
  "Effect": "Allow",
  "Action": [ "iam:PassRole" ],
  "Resource": "arn:aws:iam::account-id:role/my-role-for-XYZ"
}
```

具有服務連結角色的間接許可

可將服務連結角色授予的許可間接轉移給其他使用者和角色。服務使用服務連結角色時，該 AWS 服務連結角色可以使用其本身的權限來呼叫其他 AWS 服務。這表示使用者和角色 (具有呼叫使用服務連結角色之服務的許可) 可能會間接存取該服務連結角色所能存取的服務。

例如，當您建立 Amazon RDS 資料庫執行個體時，RDS 的服務連結角色如果尚未存在，則會自動建立。這個服務連結角色可讓 RDS 代表您呼叫亞馬 Amazon EC2、Amazon SNS、Amazon CloudWatch 日誌和亞馬遜 Kinesis。如果您允許帳戶中的使用者和角色修改或建立 RDS 資料庫，他們或許可以透過撥打 RDS 與 Amazon EC2、Amazon SNS、Amazon CloudWatch 日誌日誌和 Amazon Kinesis 資源間接互動，因為 RDS 會使用其服務連結角色存取這些資源。

建立服務連結角色

您用來建立服務連結角色的方法取決於服務。在某些情況下，您不需要手動建立一個服務連結角色。例如，當您在服務中完成特定動作 (例如建立資源)，該服務可能會為您建立服務連結角色。或者，您若在服務開始支援服務連結角色之前已在使用該服務，則服務可能已在您的帳戶中自動建立角色。如需進一步了解，請參閱 顯示在我的 AWS 帳戶中的新角色。

在其他情況下，服務可能支援使用服務主控台、API 或 CLI 手動建立服務連結角色。如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。若要了解服務是否支援建立服務連結角色，請選擇是連結以檢視該服務的服務連結角色的文件。

如果服務不支援建立角色，則可以使用 IAM 來建立服務連結角色。

> **Important**
>
> 服務連結角色算作您的 AWS 帳戶中的 IAM 角色限制，但是如果您已達到限制，仍然可以在您的帳戶中建立服務連結角色。只有服務連結角色可以超過限制。

建立服務連結角色 (主控台)

在 IAM 中建立服務連結角色之前，了解連結的服務是否自動建立服務連結角色，此外，了解是否您可以從服務主控台、API 或 CLI 建立角色。

建立服務連結角色 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色。然後，選擇 Create role (建立角色)。
3. 選擇 AWS Service (服務) 角色類型。
4. 選擇服務的使用案例。服務會定義使用案例，以包含服務所需的信任政策。然後選擇下一步。
5. 選擇一或多個許可政策以連接至角色。根據您選取的使用案例，服務可能可以執行下列任何操作：

   - 定義角色使用的許可。
   - 可讓您從有限的一組許可中進行選擇。
   - 可讓您從任何許可中進行選擇。
   - 可讓您目前無法選取政策、稍後建立政策，然後將它們連接至角色.

   選取可指派您希望角色擁有之許可政策旁的核取方塊，然後選擇 Next (下一步)。

   > **Note**
   >
   > 您指定的許可適用於任何使用角色的實體。角色預設沒有任何許可。

6. 針對 Role name (角色名稱)，服務會定義角色名稱自訂程度。如果服務定義角色的名稱，則此選項無法編輯。在其他情況下，服務可能會定義角色的字首，並且允許您輸入選用後綴。

   如果可能，輸入要新增至預設名稱的角色名稱後綴。此後綴可協助您識別此角色的用途。角色名稱在您的 AWS 帳戶內必須是獨一無二的。它們無法透過大小寫進行區分。例如，您無法建立名為 **`<service-linked-role-name>_SAMPLE`** 和 **`<service-linked-role-name>_sample`** 的角色。因為有各種實體可能會參照角色，所以您無法在建立角色之後編輯角色名稱。
7. (選用) 在 Description (說明) 中，編輯新服務連結角色的說明。
8. 您不能在建立角色時，將標籤連接至服務連結的角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
9. 檢閱角色，然後選擇 Create role (建立角色)。

建立服務連結角色 (AWS CLI)

在 IAM 中建立服務連結角色之前，了解連結的服務是否自動建立服務連結角色，並且了解是否您可以從服務的 CLI 建立角色。如果不支援服務 CLI，您可以使用 IAM 命令，使用服務擔任該角色所需的信任政策與內嵌政策來建立服務連結角色。

建立服務連結角色 (AWS CLI)

執行以下命令：

```
aws iam create-service-linked-role --aws-service-name SERVICE-NAME.amazonaws.com
```

建立服務連結角色 (AWS API)

在 IAM 中建立服務連結角色之前，了解連結的服務是否自動建立服務連結角色，並且了解是否您可以從服務的 API 建立角色。如果不支援服務 API，您可以使用 AWS API 建立服務連結角色，其中包含信任原則和內嵌政策，以供服務擔任該角色。

若要建立服務連結角色 (AWS API)

使用 CreateServiceLinkedRole API 呼叫。在請求中指定 *SERVICE_NAME_URL*.amazonaws.com 的服務名稱。

例如，要建立 Lex Bots 服務連結的角色，使用 lex.amazonaws.com。

編輯服務連結角色

您用來編輯服務連結角色的方法取決於服務。有些服務也許可讓您從服務主控台、API 或 CLI 編輯服務連結角色的許可。不過，因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。您可以從 IAM 主控台、API 或 CLI 編輯任何角色的說明。

如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。若要了解服務是否支援編輯服務連結角色，請選擇是連結以檢視該服務的服務連結角色的文件。

編輯服務連結角色描述 (主控台)

您可以使用 IAM 主控台來編輯服務連結角色的說明。

編輯服務連結角色的說明 (主控台)

1. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。

2. 選擇要修改之角色的名稱。
3. 在 Role description (角色說明) 的最右邊，選擇 Edit (編輯)。
4. 在方塊中輸入新的描述，然後選擇 Save (儲存)。

編輯服務連結角色描述 (AWS CLI)

您可以使用中的 IAM 命令 AWS CLI 來編輯服務連結角色的說明。

變更服務連結角色的說明 (AWS CLI)

1. (選用) 若要檢視角色的目前說明，請執行下列命令：

   ```
   aws iam get-role --role-name ROLE-NAME
   ```

   透過 CLI 命令，使用角色名稱 (而非 ARN) 來參照角色。例如，如果角色具有下列 ARN：`arn:aws:iam::123456789012:role/myrole`，請將角色參照為 **`myrole`**。
2. 若要更新服務連結角色的說明，請執行下列命令：

   ```
   aws iam update-role --role-name ROLE-NAME --description OPTIONAL-DESCRIPTION
   ```

編輯服務連結角色說明 (AWS API)

您可以使用 AWS API 編輯服務連結角色的說明。

若要變更服務連結角色 (AWS API) 的說明

1. (選用) 若要檢視角色的目前說明，請呼叫以下操作並指定角色的名稱：

   AWS API：GetRole
2. (選用) 若要更新角色的說明，請呼叫以下操作並指定角色的名稱 (和選用描述)：

   AWS API：UpdateRole

刪除服務連結角色

您用來建立服務連結角色的方法取決於服務。在某些情況下，您不需要手動刪除一個服務連結角色。例如，當您在服務中完成特定動作 (例如移除資源)，該服務可能會為您刪除服務連結角色。

在其他情況下，服務可能支援從服務主控台、API 或 AWS CLI手動刪除服務連結角色。

如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。若要了解服務是否支援刪除服務連結角色，請選擇是連結以檢視該服務的服務連結角色的文件。

如果服務不支援刪除角色，您可以從 IAM 主控台、API 或 AWS CLI刪除服務連結角色。若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，務必清除您的服務連結角色，之後才能將其刪除。

清除服務連結角色

您必須先確認服務連結角色沒有作用中的工作階段，並移除該角色使用的資源，之後才能使用 IAM 將其刪除。

檢查服務連結角色是否於 IAM 主控台有作用中的工作階段

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。然後，選擇服務連結角色的名稱 (而非核取方塊)。
3. 在所選角色的 Summary (摘要) 頁面中，選擇 Access Advisor (存取 Advisor) 分頁。
4. 在 Access Advisor (存取 Advisor) 分頁中，檢閱服務連結角色的近期活動。

> **Note**
>
> 如果您不確定服務是否正在使用服務連結角色，您可以嘗試刪除該角色。如果服務正在使用該角色，則刪除會失敗，而您可以檢視正在使用該角色的區域。如果服務正在使用該角色，您必須先等到工作階段結束，才能刪除該角色。您無法撤銷服務連結角色的工作階段。

若要移除服務連結角色所使用的資源

如需哪些服務支援使用服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找 Service-Linked Role (服務連結角色) 欄中顯示 Yes (是) 的服務。若要了解服務是否支援刪除服務連結角色，請選擇是連結以檢視該服務的服務連結角色的文件。請參閱該服務的文件，以了解如何移除服務連結角色所使用的資源。

刪除服務連結角色 (主控台)

您可以使用 IAM 主控台刪除服務連結角色。

刪除服務連結角色 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。然後，選擇您要刪除的角色名稱旁的核取方塊，而非名稱或資料列本身。
3. 在頁面頂端的 Role actions (角色動作) 中選擇 Delete (刪除)。
4. 在確認對話方塊中，複查上次存取的資訊，其中顯示每個選取角色上次存取 AWS 服務的時間。這可協助您確認角色目前是否作用中。如果您想要繼續進行，請選擇 Yes, Delete (是，刪除) 來提交服務連結角色以進行刪除。
5. 查看 IAM 主控台通知，監視服務連結角色刪除的進度。因為 IAM 服務連結角色刪除不同步，所以在您提交角色進行刪除之後，刪除任務可能會成功或失敗。

   - 如果任務成功，則會從清單中移除角色，而且成功通知會出現在頁面頂端。
   - 如果任務失敗，您可以從通知中選擇 View details (檢視詳細資訊) 或 View Resources (檢視資源)，以了解刪除失敗的原因。如果刪除因角色使用服務資源而失敗，則服務傳回該資訊時，通知會包含資源清單。您接著可以清除資源，並重新提交刪除。

     > **Note**
     >
     > 根據服務所傳回的資訊，您可能需要重複此程序數次。例如，您的服務連結角色可能會使用六個資源，而且您的服務可能傳回其中五項的相關資訊。如果您清除五個資源，並重新提交刪除角色，則刪除會失敗，而且服務會報告還有一個資源。服務可能會傳回所有資源、其中一些資源，或未報告任何資源。

   - 如果任務失敗，而且通知未包含資源清單，則服務可能未傳回該資訊。若要了解如何清除該服務的資源，請參閱AWS 與 IAM 搭配使用的服務。請在表格中找到您的服務，然後選擇 Yes (是) 連結，檢視該服務的服務連結角色文件。

刪除服務連結角色 (AWS CLI)

您可以使用的 IAM 命令 AWS CLI 來刪除服務連結角色。

刪除服務連結角色 (AWS CLI)

1. 如果您知道要刪除的服務連結角色的名稱，請輸入以下命令以列出您帳戶中的角色：

   ```
   aws iam get-role --role-name role-name
   ```

   透過 CLI 命令，使用角色名稱 (而非 ARN) 來參照角色。例如，如果角色具有下列 ARN：`arn:aws:iam::123456789012:role/myrole`，請將角色參照為 **myrole**。

2. 因為無法刪除正在使用或具有相關聯資源的服務連結角色，所以您必須提交刪除要求。如果不符合這些條件，則可以拒絕該請求。您必須從回應中擷取 `deletion-task-id`，以檢查刪除任務的狀態。輸入下列命令，以提交服務連結角色刪除要求：

   ```
   aws iam delete-service-linked-role --role-name role-name
   ```

3. 輸入下列命令，以檢查刪除任務的狀態：

   ```
   aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   刪除任務的狀態可以是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果刪除失敗，則呼叫會傳回失敗原因，以進行疑難排解。如果刪除因角色使用服務資源而失敗，則服務傳回該資訊時，通知會包含資源清單。您接著可以清除資源，並重新提交刪除。

   > **Note**
   >
   > 根據服務所傳回的資訊，您可能需要重複此程序數次。例如，您的服務連結角色可能會使用六個資源，而且您的服務可能傳回其中五項的相關資訊。如果您清除五個資源，並重新提交刪除角色，則刪除會失敗，而且服務會報告還有一個資源。服務可能會傳回所有資源、其中一些資源，或未報告任何資源。若要了解如何清除未報告任何資源之服務的資源，請參閱AWS 與 IAM 搭配使用的服務。請在表格中找到您的服務，然後選擇 Yes (是) 連結，檢視該服務的服務連結角色文件。

刪除服務連結角色 (AWS API)

您可以使用 AWS API 刪除服務連結角色。

若要刪除服務連結角色 (AWS API)

1. 若要提交服務連結角色的刪除要求，請致電DeleteServiceLinkedRole。在請求中，指定角色名稱。

   因為無法刪除正在使用或具有相關聯資源的服務連結角色，所以您必須提交刪除要求。如果不符合這些條件，則可以拒絕該請求。您必須從回應中擷取 `DeletionTaskId`，以檢查刪除任務的狀態。
2. 要檢查刪除狀態，請致電GetServiceLinkedRoleDeletionStatus。在請求中，指定 `DeletionTaskId`。

   刪除任務的狀態可以是 `NOT_STARTED`、`IN_PROGRESS`、`SUCCEEDED` 或 `FAILED`。如果刪除失敗，則呼叫會傳回失敗原因，以進行疑難排解。如果刪除因角色使用服務資源而失敗，則服務傳回該資訊時，通知會包含資源清單。您接著可以清除資源，並重新提交刪除。

   > **Note**
   >
   > 根據服務所傳回的資訊，您可能需要重複此程序數次。例如，您的服務連結角色可能會使用六個資源，而且您的服務可能傳回其中五項的相關資訊。如果您清除五個資源，並重新提交刪除角色，則刪除會失敗，而且服務會報告還有一個資源。服務可能會傳回所有資源、其中一些資源，或未報告任何資源。若要了解如何清除未報告任何資源之服務的資源，請參閱AWS 與 IAM 搭配使用的服務。請在表格中找到您的服務，然後選擇 Yes (是) 連結，檢視該服務的服務連結角色文件。

建立 IAM 角色

若要建立角色，您可以使用 AWS Management Console AWS CLI、適用於視窗 PowerShell的工具或 IAM API。

如果您使用 AWS Management Console，精靈會引導您完成建立角色的步驟。精靈的步驟略有不同，具體取決於您要為 AWS 服務 AWS 帳戶、為或聯合使用者建立角色。

主題

- 建立角色以委派許可給 IAM 使用者
- 建立角色以將許可委派給 AWS 服務
- 針對第三方身分提供者建立角色 (聯合身分)
- 使用自訂信任政策建立角色 (主控台)

- 委派存取權限的政策範例

建立角色以委派許可給 IAM 使用者

您可以使用 IAM 角色委派對 AWS 資源的存取權。使用 IAM 角色，您可以在信任帳戶和其他 AWS 受信任帳戶之間建立信任關係。信任帳戶擁有要存取的資源，而受信任帳戶包含需要存取資源的使用者。不過，可以讓另一個帳戶在您的帳戶中擁有資源。例如，信任的帳戶可能允許信任的帳戶來建立新的資源，例如在 Amazon S3 儲存貯體中建立新物件。在這種情況下，建立資源的帳戶擁有資源，並控制誰可以存取該資源。

建立信任關係後，IAM 使用者或來自受信任帳戶的應用程式即可使用 AWS Security Token Service (AWS STS) `AssumeRole`API 操作。此作業會提供暫時的安全性登入資料，讓您能夠存取帳戶中的 AWS 資源。

兩個帳戶可都由您控制，或者含有使用者的帳戶可由第三方進行控制。如果具有使用者的其他帳戶 AWS 帳戶 是您無法控制的帳戶，則您可以使用該`externalId`屬性。外部 ID 可以是您和第三方帳戶管理員之間商定的任何文字或數字。此選項會自動新增條件到信任政策，讓使用者只在請求包含正確的 `sts:ExternalID` 時才擔任該角色。如需詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。

如需有關如何使用角色委派許可的資訊，請參閱角色術語和概念。如需使用服務角色以允許服務存取您帳戶中的資源的資訊，請參閱建立角色以將許可委派給 AWS 服務。

建立 IAM 角色 (主控台)

您可以使用 AWS Management Console 建立 IAM 使用者可以承擔的角色。例如，假設您的組織有多個可 AWS 帳戶 以將開發環境與生產環境隔離開來。如需有關建立可讓開發帳戶中的使用者存取生產帳戶中資源的角色的高級別資訊，請參閱使用不同的開發和生產帳戶的範例方案。

建立角色 (主控台)

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 AWS 帳戶 角色類型。
4. 若要為您的帳戶建立角色，請選取 This account (此帳號)。若要為其他帳戶建立角色，請選擇 Another AWS 帳戶 (另一個)，然後輸入您要對其授予資源存取權的 Account ID (帳戶 ID)。

指定帳戶的管理員可以授予許可給該帳戶中的任何 IAM 使用者來擔任此角色。若要執行此操作，管理員要將政策連接到授予 `sts:AssumeRole` 動作之許可的使用者或群組。該政策必須指定角色的 ARN 為 `Resource`。

5. 如果您將許可授予您不控制之帳戶的使用者，而且使用者將以程式設計方式擔任此角色，則請選取 Require external ID (需要外部 ID)。外部 ID 可以是您和第三方帳戶管理員之間商定的任何文字或數字。此選項會自動新增條件到信任政策，讓使用者只在請求包含正確的 `sts:ExternalID` 時才擔任該角色。如需詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。

   > **Important**
   >
   > 選擇此選項只能透過 AWS CLI、Windows PowerShell 工具或 AWS API 來限制角色的存取。這是因為您無法使用 AWS 主控台切換到信任原則中具有`externalId`條件的角色。不過，您可以程式設計方式建立存取這類存取，即透過編寫指令碼或使用相關開發套件的應用程式。如需詳細資訊和範例指令碼，請參閱如何在 AWS 安全性部落格 AWS Management Console中啟用跨帳戶存取。

6. 如果您想限制角色為使用多重要素驗證 (MFA) 登入的使用者，請選取 Require MFA (需要 MFA)。這會新增條件到角色的信任政策，以檢查 MFA 登入。想要擔任該角色的使用者必須從設定的 MFA 裝置使用臨時的一次性密碼登入。未經 MFA 身分驗證的使用者無法擔任角色。如需有關 MFA 的詳細資訊，請參閱 在中使用多因素身份驗證（MFA）AWS

7. 選擇下一步。

8. IAM 在您的帳戶中包含受 AWS 管政策和客戶受管政策的清單。選取用於許可政策的政策，或者選擇 Create policy (建立政策) 以開啟新的瀏覽器標籤，並從頭建立新的政策。如需詳細資訊，請參閱 建立 IAM 政策。在您建立政策後，關閉該標籤並返回您的原始標籤。選取您要擔任角色的任何人具有之許可政策旁的核取方塊。如果您希望，您目前可以不選取政策，稍後再將政策連接到角色。角色預設沒有任何許可。

9. (選用) 設定許可界限。這是進階功能。

   開啟 Set permissions boundary (設定許可界限) 區段，並選擇 Use a permissions boundary to control the maximum role permissions (使用許可界限來控制角色許可上限)。選取用於許可界限的政策。

10. 選擇下一步。

11. 針對 Role name (角色名稱)，輸入您的角色名稱。角色名稱在您的 AWS 帳戶. 角色名稱用在政策中或作為 ARN 的一部分時，角色名稱區分大小寫。當主控台中的客戶顯示角色名稱時 (例如在登

入程序期間)，角色名稱不區分大小寫。因為有各種實體可能會參考此角色，所以建立角色之後，您就無法編輯其名稱。

12. (選用) 在 Description (說明) 中，輸入新角色的說明。
13. 在 Step 1: Select trusted entities (步驟 1：選取受信任的實體) 或者 Step 2: Add permissions (步驟 2：新增許可) 區段中選擇 Edit (編輯)，可編輯使用案例和角色許可。您將返回之前的頁面進行編輯。
14. (選用) 藉由連接標籤作為鍵值對，將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
15. 檢閱角色，然後選擇 Create role (建立角色)。

> **Important**
>
> 請記住，這只是所需設定的前半部。您也必須以信任的帳戶許可提供給個別使用者，以切換到主控台中的角色，或程式化方式擔任角色。如需有關此步驟的詳細資訊，請參閱 向使用者授予切換角色的許可。

建立 IAM 角色 (AWS CLI)

從中建立角色 AWS CLI 需要多個步驟。當您使用主控台建立角色時，許多步驟都是為您完成的，但是 AWS CLI 您必須明確地自行執行每個步驟。您必須建立角色，然後為該角色指派許可政策。或者，您也可以設定角色的許可界限。

若要為跨帳戶存取建立角色 (AWS CLI)

1. 建立角色：aws iam create-role
2. 將受管許可政策附加到角色：aws iam attach-role-policy

   或

   為角色建立內嵌許可政策：aws iam put-role-policy
3. (選用) 透過連接標籤來將自訂屬性新增至該角色：aws iam tag-role

   如需詳細資訊，請參閱 管理 IAM 角色 (AWS CLI 或 AWS API) 上的標籤。
4. （可選）設置角色的許可邊界：aws iam put-role-permissions-boundary

   許可界限控制角色可以擁有的許可上限。權限界限是一 AWS 項進階功能。

以下範例顯示前兩個最常見步驟，可在簡單的環境中建立一個跨帳戶角色。此範例允許 123456789012 帳戶中的任何使用者擔任角色，並檢視 example_bucket Amazon S3 儲存貯體。此範例也假定您使用執行 Windows 的用戶端電腦，並且已使用您的帳戶憑證和區域設定您的命令列界面。若要取得更多資訊，請參閱〈規劃 AWS 指令行介面〉。

當您建立角色時，這個範例在第一個命令包含下列信任政策。此信任政策允許 123456789012 帳戶中的使用者使用 AssumeRole 操作來擔任角色，但只在使用者採用 SerialNumber 和 TokenCode 參數以提供 MFA 身分驗證時。如需有關 MFA 的詳細資訊，請參閱 在中使用多因素身份驗證（MFA）AWS。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
          "Action": "sts:AssumeRole",
          "Condition": { "Bool": { "aws:MultiFactorAuthPresent": "true" } }
      }
  ]
}
```

Important

如果您的 Principal 元素包含特定 IAM 角色或使用者的 ARN，則該 ARN 會在儲存政策時轉換為唯一的主體 ID。如果有人希望藉由刪除並重新建立角色或使用者來提升許可時，這麼做可有助於減少此類風險。您通常不會在主控台中看到此 ID，因為在顯示信任政策時還會反向轉換回 ARN。不過，如果您刪除角色或使用者，則主參與者識別碼會出現在主控台中，因為 AWS 無法再將其對應回 ARN。因此，如果您刪除並重新建立了信任政策的 Principal 元素所引用的使用者或角色，您必須編輯角色來替換 ARN。

當您使用第二個命令的政策，您必須將現有受管政策連接到角色。下列許可政策允許任何擔任角色的人只對 example_bucket Amazon S3 儲存貯體執行 ListBucket 動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
```

```
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::example_bucket"
        }
    ]
}
```

若要建立此 Test-UserAccess-Role 角色，您必須先將之前的信任政策以名稱 trustpolicyforacct123456789012.json 儲存到 policies 資料夾 (在您的本機 C: 磁碟機)。然後使用名稱將先前的 AWS 帳戶 權限策略另存為客戶管理策略PolicyForRole。然後，您可以使用以下命令來建立角色，並連接受管政策。

```
# Create the role and attach the trust policy file that allows users in the specified
 account to assume the role.
$ aws iam create-role --role-name Test-UserAccess-Role --assume-role-policy-document
 file://C:\policies\trustpolicyforacct123456789012.json

# Attach the permissions policy (in this example a managed policy) to the role to
 specify what it is allowed to do.
$ aws iam attach-role-policy --role-name Test-UserAccess-Role --policy-arn
 arn:aws:iam::123456789012:policy/PolicyForRole
```

> **Important**
>
> 請記住，這只是所需設定的前半部。您還必須提供受信任帳戶中的個別使用者許可，以切換到該角色。如需有關此步驟的詳細資訊，請參閱 向使用者授予切換角色的許可。

建立角色並授與其執行 AWS 工作或存取 AWS 資源的權限後，123456789012帳戶中的任何使用者都可以擔任該角色。如需詳細資訊，請參閱 切換到 IAM 角色 (AWS CLI)。

建立身分與存取權管理角色 (AWS API)

從 AWS API 建立角色需要多個步驟。當您使用主控台建立角色時，有許多步驟會自動為您完成，但是使用 API 的話，您必須自行明確執行每個步驟。您必須建立角色，然後為該角色指派許可政策。或者，您也可以設定角色的許可界限。

在程式碼中建立角色 (AWS API)

1. 建立角色：CreateRole

對於角色的信任政策，您可以指定一個檔案位置。

2. 將受管理的權限原則附加至角色：AttachRolePolicy

   或

   建立角色的內嵌權限原則：PutRolePolicy

   > **Important**
   >
   > 請記住，這只是所需設定的前半部。您還必須提供受信任帳戶中的個別使用者許可，以切換到該角色。如需有關此步驟的詳細資訊，請參閱 向使用者授予切換角色的許可。

3. (選擇性) 透過附加標籤將自訂屬性新增至使用者：TagRole

   如需詳細資訊，請參閱 管理 IAM 使用者 (AWS CLI 或 AWS API) 的標籤。

4. (選擇性) 設定角色的權限範圍：PutRolePermissionsBoundary

   許可界限控制角色可以擁有的許可上限。權限界限是一 AWS 項進階功能。

建立角色並授與其執行 AWS 工作或存取 AWS 資源的權限後，您必須將權限授與帳戶中的使用者，以允許他們擔任該角色。如需有關擔任角色的詳細資訊，請參閱 切換到身分與存取權管理角色 (AWS API)。

建立 IAM 角色 (AWS CloudFormation)

如需在中建立 IAM 角色的詳細資訊 AWS CloudFormation，請參閱《AWS CloudFormation 使用者指南》中的資源和屬性參考和範例。

如需中 IAM 範本的詳細資訊 AWS CloudFormation，請參閱《AWS CloudFormation 使用者指南》中的AWS Identity and Access Management 範本片段。

建立角色以將許可委派給 AWS 服務

許多 AWS 服務都要求您使用角色來允許服務代表您存取其他服務中的資源。服務會擔任代您執行動作的角色稱為服務角色。當角色做為服務的專業用途時，它被歸類為 EC2 執行個體的服務角色 (舉例來說) 或服務連結角色。若要查看使用服務連結的角色支援哪些服務，或者服務是否支援任何形式的臨時憑證的詳細資訊，請參閱 AWS 與 IAM 搭配使用的服務。若要了解個別服務如何使用角色，請在表格中選擇服務名稱以查看該服務的文件。

設定PassRole權限時，您應確定使用者未傳遞角色所擁有的權限超過您希望使用者擁有的角色。例如，愛麗絲可能不被允許執行任何 Amazon S3 動作。如果愛麗絲可以將角色傳遞給允許 Amazon S3 動作的服務，則該服務可以在執行任務時代表愛麗絲執行 Amazon S3 動作。

如需有關角色如何協助您委派許可的詳細資訊，請參閱 角色術語和概念。

服務角色許可

您必須設定許可，允許 IAM 實體 (使用者或角色) 建立或編輯服務角色。

> **Note**
>
> 服務連結角色的 ARN 包括服務主體，這在下列政策中以 *SERVICE-NAME*.amazonaws.com 形式指出。不要嘗試猜測服務主體，因為它是區分大小寫，且格式可以因各 AWS 服務而異。若要檢視服務的服務主體，請參閱該服務連結的角色文件。

允許 IAM 實體建立特定服務角色

將下列政策新增至需要建立服務角色的 IAM 實體。此政策可讓您建立所指定服務且具有特定名稱的服務角色。然後，您可以將受管或內嵌政策連接至該角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
        }
    ]
}
```

允許 IAM 實體建立任何服務角色

AWS 建議您只允許系統管理使用者建立任何服務角色。具有建立角色和連接任何政策之許可的人員可以提升自己的許可。反之，建立一個政策，讓他們只建立所需的角色，或讓系統管理員代表他們建立服務角色。

若要附加可讓系統管理員存取您的整個原則 AWS 帳戶，請使用AdministratorAccess AWS 受管理的原則。

允許 IAM 實體編輯服務角色

將下列政策新增至需要編輯服務角色的 IAM 實體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EditSpecificServiceRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:DeleteRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "iam:ListRolePolicies",
                "iam:PutRolePolicy",
                "iam:UpdateRole",
                "iam:UpdateRoleDescription"
            ],
            "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
        },
        {
            "Sid": "ViewRolesAndPolicies",
            "Effect": "Allow",
            "Action": [
                "iam:GetPolicy",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

允許 IAM 實體刪除特定服務角色

將下列陳述式新增至需要刪除所指定服務角色之 IAM 實體的許可政策。

```
{
    "Effect": "Allow",
    "Action": "iam:DeleteRole",
    "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
}
```

允許 IAM 實體刪除任何服務角色

AWS 建議您只允許系統管理使用者刪除任何服務角色。反之，建立一個政策，只允許他們刪除所需的角色，或讓系統管理員代表他們刪除服務角色。

若要附加可讓系統管理員存取您的整個原則 AWS 帳戶，請使用AdministratorAccess AWS 受管理的原則。

建立 AWS 服務的角色 (主控台)

您可以使用 AWS Management Console 來建立服務的角色。因為有些服務支援多個服務角色，所以請參閱服務的 AWS 文件，以查看要選擇的使用案例。您可以了解如何為角色指派必要的信任和許可政策，以便服務可以代表您擔任角色。您用來控制角色的許可的步驟可能有所不同，端視服務如何定義使用案例，以及是否建立服務連結的角色而定。

若要建立 AWS 服務 (IAM 主控台) 的角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇服務，然後選擇使用案例。服務會定義使用案例，以包含服務所需的信任政策。
5. 選擇下一步。
6. 對於權限原則，選項取決於您選取的使用案例：
   - 如果服務定義了角色的權限，您就無法選取權限原則。
   - 從一組有限的權限原則中進行選取。
   - 從所有權限原則中選取。
   - 選取 [無權限原則]，建立角色後建立原則，然後將原則附加至角色。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。

a. 開啟 [設定權限界限] 區段，然後選擇 [使用權限界限] 控制最大角色權限。

IAM 在您的帳戶中包含受 AWS 管政策和客戶管理政策的清單。

b. 選取用於許可界限的政策。

8. 選擇下一步。
9. 對於角色名稱，選項取決於服務：

- 如果服務定義了角色名稱，您就無法編輯角色名稱。
- 如果服務定義了角色名稱的前置詞，您可以輸入選擇性的尾碼。
- 如果服務未定義角色名稱，您可以命名角色。

> **Important**
>
> 命名角色時，請注意下列事項：
>
> - 角色名稱在您的內部必須是唯一的 AWS 帳戶，並且不能根據大小寫將其唯一。
>
> 例如，請勿建立同時命名為**PRODROLE**和的角色**prodrole**。當角色名稱用於策略中或作為 ARN 的一部分時，角色名稱會區分大小寫，但是當主控台中的客戶 (例如在登入程序期間) 顯示角色名稱時，角色名稱不區分大小寫。
> - 您無法在建立角色之後編輯該角色的名稱，因為其他實體可能會參照該角色。

10. (選擇性) 在說明中，輸入角色的說明。
11. (選擇性) 若要編輯角色的使用案例和權限，請在步驟 1：選取信任的實體或步驟 2：新增權限區段中，選擇編輯。
12. (選擇性) 若要協助識別、組織或搜尋角色，請將標籤新增為鍵值配對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

為服務建立角色 (AWS CLI)

從中建立角色 AWS CLI 需要多個步驟。當您使用主控台建立角色時，許多步驟都是為您完成的，但是 AWS CLI 您必須明確地自行執行每個步驟。您必須建立角色，然後為該角色指派許可政策。如果您使用的服務是 Amazon EC2，則還必須建立執行個體描述檔並新增該角色。或者，您也可以設定角色的許可界限。

若要從中建立 AWS 服務的角色 AWS CLI

1. 以下 `create-role` 命令會建立名為 Test-Role 的角色，並將信任政策連接至該角色：

   ```
   aws iam create-role --role-name Test-Role --assume-role-policy-document
   file://Test-Role-Trust-Policy.json
   ```

2. 將受管許可政策附加到角色：aws iam attach-role-policy。

   例如，下列 `attach-role-policy` 命令會將名為 `ReadOnlyAccess` 的 AWS 受管政策連接至名為 `ReadOnlyRole` 的 IAM 角色：

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   ReadOnlyAccess --role-name ReadOnlyRole
   ```

   或

   為角色建立內嵌許可政策：aws iam put-role-policy

   若要新增嵌許可政策，請參閱以下範例：

   ```
   aws iam put-role-policy --role-name Test-Role --policy-name
   ExamplePolicy --policy-document file://AdminPolicy.json
   ```

3. (選用) 透過連接標籤來將自訂屬性新增至該角色：aws iam tag-role

   如需詳細資訊，請參閱 管理 IAM 角色 (AWS CLI 或 AWS API) 上的標籤。

4. （可選）設置角色的許可邊界：aws iam put-role-permissions-boundary

   許可界限控制角色可以擁有的許可上限。權限界限是一 AWS 項進階功能。

如果您要將該角色與 Amazon EC2 或其他使用 Amazon EC2 的 AWS 服務搭配使用，則必須將該角色存放在執行個體設定檔中。執行個體描述檔是角色的容器，可以在啟動時連接到 Amazon EC2 執行個體。執行個體設定檔只能包含一個角色，並且無法增加該限制。如果您使用建立角色 AWS Management Console，則系統會以與角色相同的名稱為您建立執行個體設定檔。如需有關執行個體描述檔的詳細資訊，請參閱 使用執行個體設定檔。如需有關如何啟動具有角色的 EC2 執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的控制對 Amazon EC2 資源的存取。

建立執行個體設定檔並將角色存放在其中 (AWS CLI)

1. 建立執行個體設定檔：aws iam create-instance-profile

2. 將角色添加到實例配置文件：aws iam add-role-to-instance 配置文件

下面的示 AWS CLI 例命令集演示了創建角色和附加權限的前兩個步驟。它還說明兩個建立執行個體設定檔和新增角色至描述檔的步驟。此範例信任允許 Amazon EC2 服務擔任角色並檢視 example_bucket Amazon S3 儲存貯體的政策。此範例也假定您在執行 Windows 的用戶端電腦上執行，並且已使用您的帳戶憑證和區域設定您的命令列界面。若要取得更多資訊，請參閱〈規劃 AWS 指令行介面〉。

當您建立角色時，這個範例在第一個命令包含下列信任政策。此信任政策允許 Amazon EC2 服務擔任該角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "ec2.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

當您使用第二個命令的政策，您必須將許可政策連接到角色。以下範例許可政策允許角色僅在 example_bucket Amazon S3 儲存貯體上執行 ListBucket 動作。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

若要建立此 Test-Role-for-EC2 角色，您必須先將之前的信任政策以名稱 trustpolicyforec2.json 和之前名為 permissionspolicyforec2.json 的許可政策儲存到您本機 policies 磁碟機的 C: 目錄。然後，您可以使用以下命令來建立角色、連接政策、建立執行個體設定檔，並新增角色到執行個體設定檔。

```
# Create the role and attach the trust policy that allows EC2 to assume this role.
$ aws iam create-role --role-name Test-Role-for-EC2 --assume-role-policy-document
 file://C:\policies\trustpolicyforec2.json
```

```
# Embed the permissions policy (in this example an inline policy) to the role to
 specify what it is allowed to do.
$ aws iam put-role-policy --role-name Test-Role-for-EC2 --policy-name Permissions-
Policy-For-Ec2 --policy-document file://C:\policies\permissionspolicyforec2.json

# Create the instance profile required by EC2 to contain the role
$ aws iam create-instance-profile --instance-profile-name EC2-ListBucket-S3

# Finally, add the role to the instance profile
$ aws iam add-role-to-instance-profile --instance-profile-name EC2-ListBucket-S3 --
role-name Test-Role-for-EC2
```

啟動 EC2 執行個體時，如果您使用 AWS 主控台，請在 [設定執行個體詳細資訊] 頁面中指定執行個體設定檔名稱。如果您使用 `aws ec2 run-instances` CLI 命令，指定 `--iam-instance-profile` 參數。

為服務建立角色 (AWS API)

從 AWS API 建立角色需要多個步驟。當您使用主控台建立角色時，有許多步驟會自動為您完成，但是使用 API 的話，您必須自行明確執行每個步驟。您必須建立角色，然後為該角色指派許可政策。如果您使用的服務是 Amazon EC2，則還必須建立執行個體描述檔並新增該角色。或者，您也可以設定角色的許可界限。

若要建立 AWS 服務的角色 (AWS API)

1. 建立角色：CreateRole

   對於角色的信任政策，您可以指定一個檔案位置。

2. 將受管理的權限原則附加至角色：AttachRole原則

   或

   建立角色的內嵌權限原則：PutRole原則

3. (選擇性) 透過附加標籤將自訂屬性新增至使用者：TagRole

   如需詳細資訊，請參閱 管理 IAM 使用者 (AWS CLI 或 AWS API) 的標籤。

4. (選擇性) 設定角色的權限範圍：PutRolePermissionsBoundary

   許可界限控制角色可以擁有的許可上限。權限界限是一 AWS 項進階功能。

如果您要將該角色與 Amazon EC2 或其他使用 Amazon EC2 的 AWS 服務搭配使用，則必須將該角色存放在執行個體設定檔中。執行個體設定檔是角色的容器。每個執行個體設定檔只能包含一個角色，並且無法增加該限制。如果您在中建立角色 AWS Management Console，則系統會以與角色相同的名稱為您建立執行個體設定檔。如需有關執行個體描述檔的詳細資訊，請參閱 使用執行個體設定檔。如需如何使用角色啟動 Amazon EC2 執行個體的詳細資訊，請參閱 Amazon EC2 使用者指南中的控制對 Amazon EC2 資源的存取。

若要建立執行個體設定檔並將角色儲存在其中 (AWS API)

1. 建立執行個體設定檔：CreateInstance設定檔
2. 將角色新增至執行個體設定檔：設AddRoleToInstance定檔

針對第三方身分提供者建立角色 (聯合身分)

您可以使用身分識別提供者，而不是在 AWS 帳戶. 透過身分識別提供者 (IdP)，您可以管理以外的使用者身分識別， AWS 並授與這些外部使用者身分識別權限，以存取您帳戶中的 AWS 資源。如需有關聯合身分與身分提供者的詳細資訊，請參閱 身分提供者與聯合。

為聯合身分使用者建立角色 (主控台)

用於為聯合身分使用者建立角色的程序將根據您選擇的第三方提供者決定：

- 對於 OpenID Connect（OIDC），請參閱。為 OpenID Connect 聯盟（控制台）創建角色
- 關於 SAML 2.0 的詳細資訊，請參閱為 SAML 2.0 聯合 (主控台) 建立角色。

為聯合存取建立角色 (AWS CLI)

從 AWS CLI 為支援的身分提供者 (OIDC 或 SAML) 建立角色的步驟是相同的。區別在於，您在先決條件步驟中建立的信任政策的內容不同。首先，遵循先決條件小節中針對您正在使用之提供者類型的步驟操作：

- 關於 &OIDC; 提供者的詳細資訊，請參閱為 OIDC 建立角色的先決條件。
- 關於 &SAML; 提供者的詳細資訊，請參閱建立適用於 SAML 的角色的先決條件。

從中建立角色 AWS CLI 需要多個步驟。當您使用主控台建立角色時，許多步驟都是為您完成的，但是 AWS CLI 您必須明確地自行執行每個步驟。您必須建立角色，然後為該角色指派許可政策。或者，您也可以設定角色的許可界限。

建立聯合身分的角色 (AWS CLI)

1. 建立角色：aws iam create-role
2. 將許可政策附加到角色：aws iam attach-role-policy

   或

   為角色建立內嵌許可政策：aws iam put-role-policy
3. (選用) 透過連接標籤來將自訂屬性新增至該角色：aws iam tag-role

   如需詳細資訊，請參閱 管理 IAM 角色 (AWS CLI 或 AWS API) 上的標籤。
4. （可選）設置角色的許可邊界：aws iam put-role-permissions-boundary

   許可界限控制角色可以擁有的許可上限。權限界限是一 AWS 項進階功能。

以下範例顯示前兩個最常見步驟，可在簡單的環境中建立一個身分提供者角色。此範例允許 123456789012 帳戶中的任何使用者擔任角色，並檢視 example_bucket Amazon S3 儲存貯體。此範例也假設您在執行 Windows 的電腦 AWS CLI 上執行，並且已 AWS CLI 使用您的認證設定。如需詳細資訊，請參閱設定 AWS Command Line Interface。

下列範例信任政策是針對使用者使用 Amazon Cognito 登入時的行動應用程式而設計。在此範例中，*###:12345678-ffff-ffff-ffff 123456* 代表 Amazon Cognito 可指派的身分集區識別碼。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "RoleForCognito",
        "Effect": "Allow",
        "Principal": {"Federated": "cognito-identity.amazonaws.com"},
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
    }
}
```

下列許可政策允許任何擔任角色的人只對 example_bucket Amazon S3 儲存貯體執行 ListBucket 動作。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

若要建立此 Test-Cognito-Role 角色，您必須先將之前的信任政策以名稱 trustpolicyforcognitofederation.json 和之前名為 permspolicyforcognitofederation.json 的許可政策儲存到您本機 policies 磁碟機的 C: 資料夾。然後，您可以使用以下命令來建立角色，並連接內嵌政策。

```
# Create the role and attach the trust policy that enables users in an account to
 assume the role.
$ aws iam create-role --role-name Test-Cognito-Role --assume-role-policy-document
 file://C:\policies\trustpolicyforcognitofederation.json

# Attach the permissions policy to the role to specify what it is allowed to do.
aws iam put-role-policy --role-name Test-Cognito-Role --policy-name
 Perms-Policy-For-CognitoFederation --policy-document file://C:\policies
\permspolicyforcognitofederation.json
```

建立聯合存取 (AWS API) 的角色

從 AWS CLI 為支援的身分提供者 (OIDC 或 SAML) 建立角色的步驟是相同的。區別在於，您在先決條件步驟中建立的信任政策的內容不同。首先，遵循先決條件小節中針對您正在使用之提供者類型的步驟操作：

- 關於 &OIDC; 提供者的詳細資訊，請參閱為 OIDC 建立角色的先決條件。
- 關於 &SAML; 提供者的詳細資訊，請參閱建立適用於 SAML 的角色的先決條件。

若要建立身分識別聯盟 (AWS API) 的角色

1. 建立角色：CreateRole
2. 將權限原則附加至角色：AttachRolePolicy

   或

   建立角色的內嵌權限原則：PutRolePolicy

3. (選擇性) 透過附加標籤將自訂屬性新增至使用者：TagRole

   如需詳細資訊，請參閱 管理 IAM 使用者 (AWS CLI 或 AWS API) 的標籤。

4. (選擇性) 設定角色的權限範圍：PutRolePermissionsBoundary

   許可界限控制角色可以擁有的許可上限。權限界限是一 AWS 項進階功能。

為 OpenID Connect（控制台）創建角色

您可以使用 OpenID Connect（OIDC）聯合身份提供程序，而不是在. AWS Identity and Access Management AWS 帳戶透過身分識別提供者 (IdP)，您可以管理以外的使用者身分識別， AWS 並授與這些外部使用者身分識別權限，以存取您帳戶中的 AWS 資源。如需關於同盟的更多資訊 IdPs，請參閱身分提供者與聯合。

為 OIDC 建立角色的先決條件

您必須先完成下列先決條件步驟，才能建立 OIDC 聯盟的角色。

若要準備建立 OIDC 聯盟的角色

1. 使用一或多個提供聯合 OIDC 身分的服務進行註冊。如果您正在建立需要存取 AWS 資源的應用程式，您也可以使用提供者資訊來設定應用程式。當您這麼做時，提供者會將應用程式唯一的 ID 提供給您的應用程式或對象。(不同的提供者可能使用不同的術語來表達此程序。本指南則使用術語設定來表示向提供者識別您應用程式的程序)。您可以在每個提供者設定多個應用程式，或在單一應用程式設定多個提供者。檢視有關使用身分提供者的相關資訊，如下所示：

   - Login with Amazon 開發人員中心
   - Facebook 開發人員網站上的新增 Facebook 登入到您的應用程式或網站。
   - Google 開發人員網站上的登入時使用 OAuth 2.0 (OpenID Connect)。

2. 從 IdP 收到必要資訊後，請在 IAM 中建立 IdP。如需詳細資訊，請參閱 在 IAM 中建立 OpenID Connect (OIDC) 身分識別提供者。

   > **Important**
   >
   > 如果您正在使用來自 Google、Facebook 或 Amazon Cognito 的 OIDC IdP，請勿在 AWS Management Console中建立單獨的 IAM IdP。這些 OIDC 身分識別提供者已內建於其中 AWS ，可供您使用。略過此步驟，並在接下來的步驟中使用您的 IdP 建立新角色。

3. 為已進行 IdP 身分驗證的使用者要擔任的角色準備政策。正如任何角色一樣，手機應用程式的角色含有兩項政策。其中一項是信任政策，其指定擔任該角色的對象。另一項政策是許可政策，其指定行動應用程式被允許或拒絕存取的 AWS 動作和資源。

   對於網路版 IdPs，我們建議您使用 Amazon Cognito 來管理身分。在這種情況下，請使用類似於這個範例的信任政策。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"Federated": "cognito-identity.amazonaws.com"},
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {
            "StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east-2:12345678-abcd-abcd-abcd-123456"},
            "ForAnyValue:StringLike": {"cognito-identity.amazonaws.com:amr":
 "unauthenticated"}
        }
    }
}
```

   以 Amazon Cognito 指派給您的身分集區 ID 取代 `us-east-2:12345678-abcd-abcd-abcd-123456`。

   如果您手動設定 OIDC IdP，則在建立信任原則時，必須使用三個值來確保只有您的應用程式可以擔任該角色：

   - 對於 `Action` 元素，可使用 `sts:AssumeRoleWithWebIdentity` 動作。
   - 如需 `Principal` 元素，請使用字串 `{"Federated":`*`providerUrl/providerArn`*`}`。
     - 對於一些常見的 OIDC IdPs，*`providerUrl`*這是一個 URL。下列範例包含指定某些常用主體的方法 IdPs：

       `"Principal":{"Federated":"cognito-identity.amazonaws.com"}`

       `"Principal":{"Federated":"www.amazon.com"}`

       `"Principal":{"Federated":"graph.facebook.com"}`

       `"Principal":{"Federated":"accounts.google.com"}`

- 對於其他的 OIDC 提供者，請使用您在 Step 2 中建立的 OIDC 身分提供者的 Amazon Resource Name (ARN)，如以下範例所示：

  ```
  "Principal":{"Federated":"arn:aws:iam::123456789012:oidc-provider/
  server.example.com"}
  ```

- 對於 Condition 元素，可使用 StringEquals 條件來限制許可。測試身分集區 ID (對於 Amazon Cognito) 或應用程式 ID (對於其他提供者)。身分集區 ID 應與您透過 IdP 配置應用程式時所收到的應用程式 ID 一致。ID 之間的比對可確保請求來自您的應用程式。

> **Note**
>
> Amazon Cognito 身分識別集區的 IAM 角色會信任服務主體cognito-identity.amazonaws.com擔任該角色。此類型的角色必須至少包含一個條件索引鍵，才能限制可擔任該角色的主參與者。
> 其他考量適用於採用跨帳戶 IAM 角色的 Amazon Cognito 身分識別集區。這些角色的信任原則必須接受cognito-identity.amazonaws.com服務主體，且必須包含aud條件索引鍵，才能將角色假設限制為來自預定識別集區的使用者。信任沒有此條件的 Amazon Cognito 身分集區的政策會造成非預期身分集區的使用者承擔該角色的風險。如需詳細資訊，請參閱 Amazon Cognito 開發人員指南中基本 (傳統) 身份驗證中 IAM 角色的信任政策。

建立類似以下其中一個範例的條件元素，其取決於您使用的 IdP：

```
"Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud":
"us-east:12345678-ffff-ffff-ffff-123456"}}
```

```
"Condition": {"StringEquals": {"www.amazon.com:app_id":
"amzn1.application-oa2-123456"}}
```

```
"Condition": {"StringEquals": {"graph.facebook.com:app_id":
"111222333444555"}}
```

```
"Condition": {"StringEquals": {"accounts.google.com:aud":
"66677788899900pro0"}}
```

對於 OIDC 提供者，請將 OIDC IdP 的完全合格 URL 與 aud 內容索引鍵一起使用，如以下範例所示：

```
"Condition": {"StringEquals": {"server.example.com:aud":
"appid_from_oidc_idp"}}
```

> **Note**
>
> 角色的信任政策中的主體的值是 IdP 特有的。OIDC 的角色只能指定一個主體。因此，如果行動應用程式允許使用者從多個 IdP 登入，則為您要支援的每個 IdP 建立不同的角色。分別為每個 IdP 建立信任政策。

如果使用者使用行動應用程式從 Login with Amazon 登入，則以下範例信任政策適用。在範例中，*amzn1.application-oa2-123456* 代表使用 Login with Amazon 設定應用程式時 Amazon 指派的應用程式 ID。

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForLoginWithAmazon",
          "Effect": "Allow",
          "Principal": {"Federated": "www.amazon.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"www.amazon.com:app_id":
 "amzn1.application-oa2-123456"}}
      }]
  }
```

如果使用者使用行動應用程式從 Facebook 登入，則以下範例信任政策適用。在本範例中，*111222333444555* 代表 Facebook 指派的應用程式 ID。

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForFacebook",
          "Effect": "Allow",
          "Principal": {"Federated": "graph.facebook.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"graph.facebook.com:app_id":
 "111222333444555"}}
```

```
      }]
  }
```

如果使用者使用行動應用程式從 Google 登入，則以下範例信任政策適用。在本範例中，*666777888999000* 代表由 Google 指派的應用程式 ID。

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForGoogle",
          "Effect": "Allow",
          "Principal": {"Federated": "accounts.google.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"accounts.google.com:aud":
 "666777888999000"}}
      }]
  }
```

如果使用者使用行動應用程式從 Amazon Cognito 登入，則以下範例信任政策適用。在此範例中，*####:12345678-ffff-ffff-ffff 123456* 代表 Amazon Cognito 可指派的身分集區識別碼。

```
{
      "Version": "2012-10-17",
      "Statement": [{
          "Sid": "RoleForCognito",
          "Effect": "Allow",
          "Principal": {"Federated": "cognito-identity.amazonaws.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
      }]
  }
```

建立 OIDC 的角色

完成先決條件後，您可在 IAM 建立角色。下列程序說明如何在中建立 OIDC 聯盟的角色。 AWS Management Console若要從 AWS CLI 或 AWS API 建立角色，請參閱中的程序針對第三方身分提供者建立角色 (聯合身分)。

> **Important**
>
> 如果您使用 Amazon Cognito，請使用 Amazon Cognito 主控台來設定角色。否則，請使用 IAM 主控台為 OIDC 聯盟建立角色。

若要為 OIDC 聯盟建立 IAM 角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 OIDC 角色類型。
4. 針對 Identity provider (身分提供者)，選擇您角色的身分提供者：
   - 如果要為個別 Web 身分提供者建立角色，請選擇 Login with Amazon、Facebook 或 Google。

     > **Note**
     >
     > 您必須為想要支援的每個身分提供者建立單獨的角色。

   - 如果想要為 Amazon Cognito 建立進階案例角色，請選擇 Amazon Cognito。

     > **Note**
     >
     > 只有在處理進階案例時，必須手動建立與 Amazon Cognito 一起使用的角色。否則，Amazon Cognito 可以為您建立角色。如需 Amazon Cognito 的詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的身分集區 (聯合身分) 外部身分提供者。

   - 如果您想要為 GitHub 動作建立角色，必須先將 GitHub OIDC 提供者新增至 IAM。將 GitHub OIDC 提供者新增至 IAM 之後，請選擇權杖。

> **Note**
>
> 有關如何將信任的 OIDC 設定 AWS 為聯合身分 GitHub的詳細資訊，請參閱GitHub 文件-在 Amazon Web Services 中設定 OpenID Connect。如需限制與 IAM IdP 相關聯之角色存取權的最佳做法的資訊 GitHub，請參閱本頁設定 GitHub OIDC 身分識別提供者的角色上的。

5. 輸入您的應用程式的識別碼。識別碼的標籤會因所選擇的提供者而改變：
   - 如果您要為 Login with Amazon 建立角色，請將應用程式 ID 輸入 Application ID (應用程式 ID) 方塊中。
   - 如果您要為 Facebook 建立角色，請將應用程式 ID 輸入 Application ID (應用程式 ID) 方塊中。
   - 如果您要為 Google 建立角色，請在 Audience (對象) 方塊中輸入對象名稱。
   - 如果您要為 Amazon Cognito 建立角色，請在 Identity Pool ID (身分集區 ID) 方塊中輸入您為 Amazon Cognito 應用程式建立的身分集區 ID。
   - 如果您要建立 GitHub 「動作」的角色，請輸入下列詳細資訊：
     - 針對 Audience (對象)，選擇 `sts.amazonaws.com`。
     - 若為GitHub 組織，請輸入您的 GitHub 組織名稱。 GitHub 組織名稱是必要的，且必須是包含破折號 (-) 的英數字元。您不能使用萬用字元 (* 和?) 在 GitHub 組織名稱中。
     - (選擇性) 對於GitHub 存放庫，請輸入 GitHub存放庫名稱。如果您不指定值，則會預設為萬用字元 (*)。
     - (選擇性) 對於GitHub 分支，請輸入 GitHub 分支名稱。如果您不指定值，則會預設為萬用字元 (*)。
6. (選用) 針對條件 (選用)，選擇新增條件，以建立應用程式使用者在能夠使用角色所授予的許可之前所必須滿足的其他條件。例如，您可以新增條件，僅授與特定 IAM 使用者 ID 的 AWS 資源存取權。您也可以在建立角色之後，將條件新增至信任政策。如需詳細資訊，請參閱 修改角色信任政策 (主控台)。
7. 檢閱您的 OIDC 資訊，然後選擇 [下一步]。
8. IAM 在您的帳戶中包含受 AWS 管政策和客戶受管政策的清單。選取用於許可政策的政策，或者選擇 Create policy (建立政策) 以開啟新的瀏覽器標籤，並從頭建立新的政策。如需詳細資訊，請參閱 建立 IAM 政策。在您建立政策後，關閉該標籤並返回您的原始標籤。選取您希望 OIDC 使用者擁有的權限原則旁邊的核取方塊。如果您希望，您目前可以不選取政策，稍後再將政策連接到角色。角色預設沒有任何許可。
9. (選用) 設定許可界限。這是進階功能。

開啟 Permissions boundary (許可界限) 區段，並選擇 Use a permissions boundary to control the maximum role permissions (使用許可界限來控制角色許可上限)。選取用於許可界限的政策。

10. 選擇下一步。
11. 在 Role name (角色名稱) 中，輸入角色名稱。角色名稱在您的 AWS 帳戶. 它們不區分大小寫。例如，您無法建立名為 **PRODROLE** 和 **prodrole** 的角色。由於其他 AWS 資源可能會參照該角色，因此您無法在建立角色之後編輯該角色的名稱。
12. (選用) 在 Description (說明) 中，輸入新角色的說明。
13. 如要編輯使用案例和角色許可，請在 Step 1: Select trusted entities (步驟 1：選取受信任的實體) 或者 Step 2: Add permissions (步驟 2：新增許可) 區段中選擇 Edit (編輯)。
14. (選用) 若要將中繼資料新增至角色，請附加標籤做為鍵/值對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
15. 檢閱角色，然後選擇 Create role (建立角色)。

設定 GitHub OIDC 身分識別提供者的角色

如果您用 GitHub 作 OpenID Connect (OIDC) 身分識別提供者 (IdP)，最佳做法是限制可擔任與 IAM IdP 相關聯之角色的實體。當您在信任原則中包含條件陳述式時，可以將角色限制為特定 GitHub組織、存放庫或分支。您可以使用具有字符條件運算子的條件索引鍵 `token.actions.githubusercontent.com:sub` 來限制存取權。我們建議您將條件限制在 GitHub 組織內的一組特定存放庫或分支。有關如何將信任的 OIDC 設定 AWS 為聯合身分 GitHub的詳細資訊，請參閱GitHub 文件-在 Amazon Web Services 中設定 OpenID Connect。

如果您在動作工作流程或 OIDC 原則中使用環 GitHub 境，我們強烈建議您在環境中新增保護規則以提高安全性。使用部署分支和標籤來限制哪些分支和標籤可以部署到環境中。如需有關使用保護規則設定環境的詳細資訊，請參閱 < 使用部署環境 > 一文中 GitHub的部署分支和標籤。

當 GitHub OIDC IdP 是您角色的受信任主體時，IAM 會檢查角色信任政策條件以驗證條件`token.actions.githubusercontent.com:sub`是否存在條件金鑰，且其值不僅是萬用字元 (* 和?) 或空。IAM 會在建立或更新信任政策時執行此檢查。如果條件索引鍵 `token.actions.githubusercontent.com:sub` 不存在，或者鍵值不滿足上述值標準，請求將失敗且會傳回錯誤。

> **Important**
>
> 如果您未`token.actions.githubusercontent.com:sub`將條件金鑰限制在特定組織或儲存庫，則來自您控制範圍以外的組織或儲存庫的 GitHub 動作可以在您的 AWS 帳戶中扮演與 GitHub IAM IdP 相關聯的角色。

下列範例信任原則會限制對已定義 GitHub 組織、存放庫和分支的存取。下列範例中的條件索引鍵`token.actions.githubusercontent.com:sub`值是由記錄的預設主旨值格式 GitHub。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::012345678910:oidc-provider/
token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub":
 "repo:GitHubOrg/GitHubRepo:ref:refs/heads/GitHubBranch"
        }
      }
    }
  ]
}
```

下列範例條件會限制對已定義 GitHub 組織和存放庫的存取，但會授予存取存放庫內任何分支的存取權。

```
"Condition": {
  "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/GitHubRepo:*"
  }
```

```
}
```

下列範例條件會限制對已定義 GitHub 組織內任何存放庫或分支的存取。建議您將條件索引鍵限制`token.actions.githubusercontent.com:sub`為限制 GitHub 組織內部 GitHub 動作存取的特定值。

```
"Condition": {
  "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/*"
  }
}
```

如需有關原則中可用於條件檢查的 OIDC 聯合金鑰的詳細資訊，請參閱。AWS OIDC 聯盟的可用金鑰

為 SAML 2.0 聯合 (主控台) 建立角色

您可以使用 SAML 2.0 聯盟，而不是在您 AWS 帳戶的. 透過身分識別提供者 (IdP)，您可以管理以外的使用者身分識別， AWS 並授與這些外部使用者身分識別權限，以存取您帳戶中的 AWS 資源。如需有關聯合身分與身分提供者的詳細資訊，請參閱 身分提供者與聯合。

> **Note**
>
> 若要改善聯合彈性，建議您將 IdP 和 AWS 聯合設定為支援多個 SAML 登入端點。如需詳細資訊，請參閱 AWS 安全性部落格文章如何使用地區性 SAML 端點進行容錯移轉。

建立適用於 SAML 的角色的先決條件

您必須先完成以下先決條件步驟，然後才能建立用於 SAML 2.0 聯合身分的角色。

準備建立用於 SAML 2.0 聯合身分的角色

1. 在為以 SAML 為基礎的聯合身分建立角色之前，必須在 IAM 中建立 SAML 提供者。如需詳細資訊，請參閱 在 IAM 中建立 SAML 身分識別提供者。
2. 為已進行 SAML 2.0–身分驗證的使用者要擔任的角色準備政策。正如任何角色一樣，SAML 聯合身分的角色含有兩項政策。其中一項是角色信任政策，指定擔任該角色的對象。另一個是 IAM 許可政策，用於指定允許或拒絕聯合身分使用者存取的 AWS 動作和資源。

當您為您的角色建立信任政策時，必須使用三個值，以確保只有您的應用程式可擔任該角色：

- 對於 Action 元素，可使用 sts:AssumeRoleWithSAML 動作。
- 如需 Principal 元素，請使用字串 {"Federated":*ARNofIdentityProvider*}。將 *ARNofIdentityProvider* 取代為您在 Step 1 中建立的 SAML 身分提供者 ARN。
- 對於 Condition 元素，使用 StringEquals 條件來測試 SAML 回應中的 saml:aud 屬性是否與 AWS的 SAML 聯合身分端點相符。

以下範例信任政策是專為 SAML 聯合身分使用者設計的政策：

```
{
    "Version": "2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": "sts:AssumeRoleWithSAML",
      "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/PROVIDER-
NAME"},
      "Condition": {"StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/
saml"}}
    }
  }
```

將主體 ARN 取代為您在 IAM 中所建立的 SAML 提供者的實際 ARN。它會有自己的帳戶 ID 和提供者名稱。

建立 SAML 的角色

完成必要步驟後，您可以建立以 SAML 為基礎的聯合身分角色。

若要為以 SAML 為基礎的聯合身分建立角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 SAML 2.0 federation (SAML 2.0 聯合身分) 角色類型。
4. 針對 Select a SAML provider (選擇 SAML 提供者)，選擇您角色的提供者。
5. 選擇 SAML 2.0 存取層級的方法。

- 選擇「僅允許程式設計方式存取」，以建立可從 AWS API 或 AWS CLI以程式設計方式假設的角色。
- 選擇 [允許程式設計和 AWS Management Console 存取]，以建立可透過程式設計方式或從 AWS Management Console.

同時建立類似的角色，但也可以從主控台所擔任的角色包括具有特定條件的信任政策。該條件明確確保可將 SAML 觀眾 (SAML:aud 屬性) 設定為 SAML 的 AWS 登入端點 (https://signin.aws.amazon.com/saml)。

6. 如果要為程式設計存取建立角色，請從 Attribute (屬性) 清單選取屬性。然後，在 Value (值) 方塊中，輸入包含在角色中的值。這會限制從身分提供者存取使用者的角色，其身分提供者的 SAML 身分驗證回應 (聲明) 包括您指定的屬性。您必須至少指定一個屬性，以確保您的角色僅限於組織中的一部分使用者。

   如果要為程式設計和主控台存取建立角色，則會自動新增 SAML:aud 屬性並將其設定為 AWS SAML 端點的 URL (https://signin.aws.amazon.com/saml)。

7. 若要向信任政策新增更多與屬性相關的條件，請選擇 Condition (optional) (條件 (選用))，並選取其他條件，然後指定值。

   > **Note**
   >
   > 該清單包含最常用的 SAML 屬性。IAM 支援可用於建立條件的其他屬性。如需所支援屬性的清單，請參閱 SAML 聯合身分的可用金鑰。如果您需要不在清單中的受支援 SAML 屬性的條件，您可以手動新增該條件。若要這麼做，請在建立角色後編輯信任政策。

8. 檢閱您的 SAML 2.0 信任資訊，然後選擇 Next (下一步)。
9. IAM 在您的帳戶中包含受 AWS 管政策和客戶受管政策的清單。選取用於許可政策的政策，或者選擇 Create policy (建立政策) 以開啟新的瀏覽器標籤，並從頭建立新的政策。如需詳細資訊，請參閱 建立 IAM 政策。在您建立政策後，關閉該標籤並返回您的原始標籤。選取您希望 OIDC 同盟使用者擁有的權限原則旁邊的核取方塊。如果您希望，您目前可以不選取政策，稍後再將政策連接到角色。角色預設沒有任何許可。
10. (選用) 設定許可界限。這是進階功能。

    開啟 Permissions boundary (許可界限) 區段，並選擇 Use a permissions boundary to control the maximum role permissions (使用許可界限來控制角色許可上限)。選取用於許可界限的政策。

11. 選擇下一步。

12. 選擇下一步：檢閱。
13. 在 Role name (角色名稱) 中，輸入角色名稱。角色名稱在您的 AWS 帳戶. 它們無法透過大小寫進行區分。例如，您無法建立名為 **PRODROLE** 和 **prodrole** 的角色。由於其他 AWS 資源可能會參照該角色，因此您無法在建立角色之後編輯該角色的名稱。
14. (選用) 在 Description (說明) 中，輸入新角色的說明。
15. 在 Step 1: Select trusted entities (步驟 1：選取受信任的實體) 或者 Step 2: Add permissions (步驟 2：新增許可) 區段中選擇 Edit (編輯)，可編輯使用案例和角色許可。
16. (選用) 藉由連接標籤作為鍵值對，將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
17. 檢閱角色，然後選擇 Create role (建立角色)。

在您建立角色之後，您透過 AWS的相關資訊設定您的身分提供者軟體來完成 SAML 信任。此資訊包您要聯合身分使用者使用的角色。這是指在您的 IdP 和 AWS之間設定依賴方信任。如需更多詳細資訊，請參閱 使用信賴方信任和新增宣告來設定您的 SAML 2.0 IdP。

使用自訂信任政策建立角色 (主控台)

您可以建立自訂信任原則來委派存取權，並允許其他人在您的 AWS 帳戶. 如需詳細資訊，請參閱 建立 IAM 政策。

如需有關如何使用角色委派許可的資訊，請參閱角色術語和概念。

使用自訂信任政策 (主控台) 建立 IAM 角色

您可以使用 AWS Management Console 建立 IAM 使用者可以承擔的角色。例如，假設您的組織有多個可 AWS 帳戶 以將開發環境與生產環境隔離開來。如需有關建立可讓開發帳戶中的使用者存取生產帳戶中資源的角色的高階資訊，請參閱使用不同的開發和生產帳戶的範例方案。

使用自訂信任政策 (主控台) 建立角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 Custom trust policy (自訂信任政策) 角色類型。
4. 在 Custom trust policy (自訂信任政策) 區段中，輸入或貼上角色的自訂信任政策。如需詳細資訊，請參閱 建立 IAM 政策。

5. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。
6. 選取您建立的自訂信任政策旁的核取方塊。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。

   開啟 Permissions boundary (許可界限) 區段，並選擇 Use a permissions boundary to control the maximum role permissions (使用許可界限來控制角色許可上限)。IAM 在您的帳戶中包含受 AWS 管政策和客戶受管政策的清單。選取用於許可界限的政策。
8. 選擇下一步。
9. 針對 Role name (角色名稱)，服務會定義角色名稱自訂程度。如果服務定義角色名稱，則無法編輯此選項。在其他情況下，服務可能會定義角色的字首，並且可允許您輸入選用後綴。有些服務可讓您指定角色的完整名稱。

   如有可能，請輸入角色名稱或角色名稱後綴。角色名稱在您的 AWS 帳戶. 它們無法透過大小寫進行區分。例如，您無法建立名為 **`PRODROLE`** 和 **`prodrole`** 的角色。由於其他 AWS 資源可能會參照該角色，因此您無法在建立角色之後編輯該角色的名稱。
10. (選用) 在 Description (說明) 中，輸入新角色的說明。
11. 在 Step 1: Select trusted entities (步驟 1：選取受信任的實體) 或者 Step 2: Add permissions (步驟 2：新增許可) 區段中選擇 Edit (編輯)，可編輯角色的自訂政策和許可。
12. (選用) 藉由連接標籤作為鍵值對，將中繼資料新增至角色。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

委派存取權限的政策範例

下列範例顯示如何允許或授與其他資源的 AWS 帳戶 存取權 AWS 帳戶。若要了解如何使用這些範例 JSON 政策文件來建立 IAM 政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

主題

- 使用角色委派對其 AWS 帳戶 他資源之資源的存取權
- 使用政策將存取權限委託給服務
- 使用以資源為基礎的政策來指派存取另一個帳戶中的 Amazon S3 儲存貯體的權限。
- 使用以資源為基礎的政策來指派存取另一個帳戶中的 Amazon SQS 佇列的權限。
- 當拒絕存取帳戶時，不得委託存取

使用角色委派對其 AWS 帳戶 他資源之資源的存取權

如需顯示如何使用 IAM 角色對某個帳戶中使用者授予存取另一個帳戶中 AWS 資源的教學，請參閱 IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶。

> **Important**
>
> 您可以在角色信任政策的 Principal 元素中包含特定角色或使用者的 ARN。當您儲存原則時，會將 ARN AWS 轉換為唯一的主參與者識別碼。如果有人希望藉由刪除並重新建立角色或使用者來提升特權，這麼做可有助於減輕此類風險。您通常不會在主控台中看到此 ID，因為在顯示信任政策時還會反向轉換回 ARN。不過，如果您刪除角色或使用者，則關係會中斷。即使重新建立使用者或角色，您的政策都不再適用，因為它不符合信任政策中所儲存的主體 ID。發生這種情況時，主體識別碼會顯示在主控台中，因為 AWS 無法再將其對應回 ARN。結果是，如果您刪除並重新建立了信任政策的 Principal 元素所引用的使用者或角色，您必須編輯角色來替換 ARN。當您儲存政策時，它會轉換為新的主體 ID。

使用政策將存取權限委託給服務

以下範例顯示一個可以連接到角色的政策。該政策可讓 Amazon EMR 和 AWS Data Pipeline兩種服務擔任該角色。然後服務可以執行指派給角色的許可政策所授予的任何任務 (不顯示)。若要指定多個服務主體，不用指定兩個 Service 元素；您可以只使用一個該元素。實際上，您可以將一組多個服務主體做為單一 Service 元素的值。

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

使用以資源為基礎的政策來指派存取另一個帳戶中的 Amazon S3 儲存貯體的權限。

在本範例中，帳戶 A 使用資源類型政策 (一個 Amazon S3 儲存貯體政策) 授予帳戶 B 完全存取帳戶 A 的 S3 儲存貯體。然後，帳戶 B 建立一個 IAM 使用者政策，向帳戶 B 中的一個使用者授予針對帳戶 A 的儲存貯體的存取權限。

帳戶 A 中的 S3 儲存貯體政策可能與以下政策類似。在本範例中，帳戶 A 的 S3 儲存貯體的名稱為 mybucket，而帳戶 B 的帳戶號碼為 111122223333。它在帳戶 B 中未指定任何單一使用者或群組，僅指定帳戶本身。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBAccess1",
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }
}
```

或者，帳戶 A 可使用 Amazon S3 存取控制清單 (ACL) 來授予帳戶 B 存取 S3 儲存貯體或某個儲存貯體內的單一物件。這種情況下，唯一改變的是帳戶 A 授權存取帳戶 B 的方式。如此範例的下一個部分所述，帳戶 B 仍使用一個政策委託針對帳戶 B 中的 IAM 群組的存取許可。如需有關對 S3 儲存貯體和物件實施控制存取的詳細資訊，請前往Amazon Simple Storage Service 使用者指南中的存取控制。

帳戶 B 的管理員可能建立以下政策範例。該政策向帳戶 B 中的群組或使用者授予讀取存取許可。前一政策向帳戶 B 授予存取許可。但是，除非組或使用者政策明確授予資源存取許可，否則帳戶 B 中的單個群組和使用者不能存取資源。此政策中的許可只能是上述跨帳戶政策中的許可的一個子集。相比於帳戶 A 在第一個政策中授予帳戶 B 的許可，帳戶 B 無法向其群組和使用者授予更多許可。在此政策中，將明確定義 `Action` 元素以僅允許 `List` 操作，而且此政策的 `Resource` 元素與由帳戶 A 執行的儲存貯體政策的 `Resource` 配對。

為執行此政策，帳戶 B 使用 IAM 將其連接到帳戶 B 中的適用使用者 (或群組)。

```
{
  "Version": "2012-10-17",
  "Statement": {
```

```
    "Effect": "Allow",
    "Action": "s3:List*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }
}
```

使用以資源為基礎的政策來指派存取另一個帳戶中的 Amazon SQS 佇列的權限。

在下列範例中，帳戶 A 具有 Amazon SQS 佇列，而此佇列使用連接至佇列的資源類型政策向帳戶 B 授予佇列存取權。然後，帳戶 B 使用 IAM 群組原則委派帳戶 B 中群組的存取權。

以下範例佇列政策為帳戶 B 授予許可，以對帳戶 A 中名為 queue1 的佇列執行 SendMessage 和 ReceiveMessage 動作，但只能在 2014 年 11 月 30 日中午 12:00 至下午 3:00 之間執行該動作。Account B 的帳戶編號為 1111-2222-3333。帳戶 A 使用 Amazon SQS 執行此政策。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": [
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": ["arn:aws:sqs:*:123456789012:queue1"],
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2014-11-30T12:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2014-11-30T15:00Z"}
    }
  }
}
```

帳戶 B 向帳戶 B 中的組委託存取許可的政策可能類似於以下範例。帳戶 B 使用 IAM 將此政策連接到群組 (或使用者)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
```

```
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:123456789012:queue1"
  }
}
```

在前述 IAM 使用者政策範例中，帳戶 B 使用萬用字元授權其使用者存取針對帳戶 A 的佇列的所有 Amazon SQS 操作。但是帳戶 B 可委託的範圍僅限於帳戶 B 被授權存取的範圍。擁有第二個政策的帳戶 B 群組只能在 2014 年 11 月 30 日中午至下午 3:00 之間存取該佇列。根據帳戶 A 的 Amazon SQS 佇列政策的定義，使用者只能執行 SendMessage 與 ReceiveMessage 操作。

當拒絕存取帳戶時，不得委託存取

如果其他帳戶明確拒絕存取使用者的父帳戶，則 AWS 帳戶 無法委派對其他帳戶資源的存取權。拒絕將傳播到該帳戶內的所有使用者，無論使用者的現有政策是否授予這些使用者存取許可。

例如，帳戶 A 編寫了一個針對其帳戶中 S3 儲存貯體的儲存貯體政策，其中明確拒絕了帳戶 B 存取帳戶 A 的儲存貯體。但帳戶 B 編寫了一個 IAM 使用者政策，其中對帳戶 B 中的一個使用者授予了對帳戶 A 的儲存貯體的存取權限。應用於帳戶 A 的 S3 儲存貯體的「明確拒絕」將傳播到帳戶 B 中的使用者。它會覆蓋用於對帳戶 B 中使用者授予存取權限的 IAM 使用者政策 (有關如何計算許可的詳細資訊，請參閱 政策評估邏輯)。

帳戶 A 的儲存貯體政策可能與下列政策類似。在本範例中，帳戶 A 的 S3 儲存貯體的名稱為 mybucket，而帳戶 B 的帳戶號碼為 1111-2222-3333。帳戶 A 使用 Amazon S3 執行此政策。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "AccountBDeny",
    "Effect": "Deny",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::mybucket/*"
  }
}
```

此「明確拒絕」將覆蓋帳戶 B 中所有提供帳戶 A 中 S3 儲存貯體存取許可的政策。

使用 IAM 角色

在使用者、應用程式或服務可以使用您建立的角色之前，您必須授與切換到該角色的許可。您可以使用任何附加至群組或使用者的原則來授與必要的權限。本節說明如何授予使用者使用角色的許可。它

也說明使用者如何從、Windows PowerShell 工具 AWS Management Console、 AWS Command Line Interface (AWS CLI) 和 AssumeRoleAPI 切換到角色。

> **Important**
>
> 當您以程式設計方式而不是在 IAM 主控台中建立角色時，則除了 RoleName 之外，您還可以選擇新增高達 512 個字元的 Path，該字元長度最多 64 個字元。但是，如果您打算在中使用具有「切換角色」功能的角色 AWS Management Console，則合併Path且RoleName不能超過 64 個字元。

您可以從中切換角色 AWS Management Console。您可以透過呼叫 AWS CLI 或 API 作業或使用自訂 URL 來擔任角色。您使用的方法決定誰可以擔任該角色以及角色工作階段可以持續多久。使用 AssumeRole* API 操作時，您擔任的 IAM 角色是資源。呼叫 AssumeRole* API 操作的使用者或角色是主體。

比較使用角色的方法

擔任角色的方法	誰可以擔任這個角色	指定憑證生命週期的方法	憑證存留期 (最小 | 最大 | 預設)
AWS Management Console	使用者 (透過切換角色)	Role (角色) 摘要頁面上的 Maximum session duration (最大工作階段持續時間)	15 分鐘 | 最大工作階段持續時間設定[2] | 1 小時
assume-role CLI 或 AssumeRole API 操作	使用者或角色[1]	duration-seconds CLI 或 DurationSeconds API 參數	15 分鐘 | 最大工作階段持續時間設定[2] | 1 小時
assume-role-with-saml CLI 或 AssumeRol	任何使用 SAML 驗證的使用者	duration-seconds CLI 或 DurationSeconds API 參數	15 分鐘 | 最大工作階段持續時間設定[2] | 1 小時

擔任角色的方法	誰可以擔任這個角色	指定憑證生命週期的方法	憑證存留期 (最小 | 最大 | 預設)
eWithSAML API 操作			
assume-role-with-web-identity CLI 或 AssumeRoleWithWebIdentity API 操作	使用 OIDC 提供者驗證的任何使用者	duration-seconds CLI 或 DurationSeconds API 參數	15 分鐘 | 最大工作階段持續時間設定[2] | 1 小時
使用 建構的主控台 URLAssumeRole	使用者或角色	SessionDuration URL 中的 HTML 參數	15 分鐘 | 12 小時 | 1 小時
使用 建構的主控台 URLAssumeRoleWithSAML	任何使用 SAML 驗證的使用者	SessionDuration URL 中的 HTML 參數	15 分鐘 | 12 小時 | 1 小時
使用 建構的主控台 URLAssumeRoleWithWebIdentity	使用 OIDC 提供者驗證的任何使用者	SessionDuration URL 中的 HTML 參數	15 分鐘 | 12 小時 | 1 小時

[1] 使用一個角色的憑證來擔任不同的角色稱為角色鏈結。當您使用角色鏈結時，新憑證的最大持續時間限制為一小時。使用角色授予許可給在 EC2 執行個體上執行的應用程式時，那些應用程式不受此限制。

[2] 此設定的值可介於 1 小時至 12 小時。有關修改最大工作階段持續時間設定的詳細資訊，請參閱 修改角色。此設定決定取得角色憑證時可以請求的最大工作階段持續時間。例如，當您使用 AssumeRole* API 作業擔任角色時，您可以使用DurationSeconds參數指定工作階段長度。使用此參數指定 900

秒 (15 分鐘) 到角色的最大工作階段持續時間設定之間的角色工作階段長度。在主控台中切換角色的 IAM 使用者會被授予最長工作階段持續時間或使用者工作階段中的剩餘時間，以較短者為準。假設您在角色上設定 5 小時的最大持續時間。已登入主控台 10 小時的 IAM 使用者 (超出預設最大值 12 小時) 會切換至角色。可用的角色工作階段持續時間為 2 小時。如要了解如何查看角色的最大值，請參閱本頁後述的 查看角色的最大工作階段持續時間設定。

備註

- 工作階段持續時間設定上限不會限制由 AWS 服務擔任的工作階段。
- Amazon EC2 IAM 角色登入資料不受該角色中設定的最長工作階段持續時間限制。
- 若要允許使用者在角色工作階段中再次擔任目前的角色，請在角色信任原則中指定角色 AWS 帳戶 ARN 或 ARN 作為主參與者。 AWS 服務 提供 Amazon EC2、Amazon ECS、亞馬遜 EKS 和 Lambda 等運算資源提供臨時登入資料並自動更新這些登入資料。此可確保您始終擁有一組有效的憑證。對於這些服務，不需要再次擔任目前的角色即可取得臨時憑證。但是，如果您想要傳遞工作階段標籤或一個工作階段政策，則需要再次擔任目前的角色。若要瞭解如何修改角色信任原則以新增主參與者角色 ARN 或 AWS 帳戶 ARN，請參閱。修改角色信任政策 (主控台)

主題

查看角色的最大工作階段持續時間設定

您可以使用或使用 AWS Management Console 或 AWS API 來指定角色的工作階段持續時間 AWS CLI 上限。當您使用 AWS CLI 或 API 作業擔任角色時，您可以指定`DurationSeconds`參數的值。您可以

使用此參數指定角色工作階段持續時間，範圍從 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定。在指定參數之前，您應該查看您的角色設定。如果您指定 `DurationSeconds` 參數的值高於最大設定值，則操作將失敗。

要查看角色的最大工作階段持續時間 (主控台)

1. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
2. 選擇要查看的角色名稱。
3. 在 Maximum session duration (持續時間上限) 旁，檢視授予角色的工作階段長度上限。這是您可以在或 API 作業中指定的 AWS CLI工作階段持續時間上限。

要查看角色的最大工作階段持續時間設定 (AWS CLI)

1. 如果您不知道要擔任的角色的名稱，請執行以下命令列出您帳戶中的角色：
   - aws iam list-roles
2. 要查看角色的最大工作階段持續時間，請執行以下命令。然後，檢視最大工作階段持續時間參數。
   - aws iam get-role

若要檢視角色的工作階段持續時間上限設定 (AWS API)

1. 如果您不知道要擔任的角色的名稱，請呼叫以下操作列出您帳戶中的角色：
   - ListRoles
2. 要查看角色的最大工作階段持續時間，請執行以下操作。然後，檢視最大工作階段持續時間參數。
   - GetRole

向使用者授予切換角色的許可

當管理員建立用於跨帳戶存取的角色時，他們會在擁有角色和資源的帳戶 (信任帳戶) 和包含使用者的帳戶 (可信帳戶) 之間建立信任。若要執行此操作，信任帳戶的系統管理員要在角色的信任政策中將可信的帳戶編號指定為 `Principal`。這可能會允許可信帳戶中的任何使用者擔任該角色。要完成設定，信任帳戶的管理員必須為該帳戶中的特定群組或使用者提供切換到該角色的許可。

要授予許可以切換到角色

1. 身為可信帳戶的管理員，請為使用者建立新政策，或編輯現有政策來新增必要元素。如需詳細資訊，請參閱 建立或編輯政策。
2. 然後，選擇您想要如何共享角色資訊：

   - 角色連結：向使用者傳送連結，讓他們進入已填寫所有詳細資訊的 Switch Role (切換角色) 頁面。
   - 帳戶 ID 或別名：為每個使用者提供角色名稱以及帳戶 ID 號碼或帳戶別名。使用者接著前往 Switch Role (切換角色) 頁面，然後手動新增詳細資訊。

   如需詳細資訊，請參閱 提供資訊給使用者。

請注意，只有當您做為 IAM 使用者、SAML 聯合角色或 Web 聯合身分角色登入時，才能切換角色。如果您以 AWS 帳戶根使用者登入，則無法切換角色。

> **Important**
>
> 您無法將中的角色切換 AWS Management Console 到需要ExternalId值的角色。您只能透過呼叫支援 `ExternalId` 參數的 `AssumeRole` API 來切換到此類角色。

> **備註**
>
> - 本主題討論使用者的政策，因為您最終會向使用者授予完成任務的許可。不過，我們不建議您直接向個別使用者授予許可。當使用者擔任角色時，他們會被指派得到與該角色關聯的許可。
> - 當您在中切換角色時 AWS Management Console，主控台一律會使用您的原始認證來授權交換器。無論您作為 IAM 使用者、SAML 聯合角色還是 Web 聯合身分角色登入，上述情形均適用。例如，如果您切換到 RoleA，IAM 會使用您的原始使用者或聯合角色憑證確定是否允許您擔任 RoleA。如果您在使用 RoleA 時嘗試切換到 RoleB，仍會使用您的原始使用者或聯合身分角色憑證對您切換到 RoleB 的嘗試進行授權。RoleA 的憑證不會用於此動作。

主題

- 建立或編輯政策

- 提供資訊給使用者

建立或編輯政策

授予使用者擔任角色的許可政策必須包含在下列項目中擁有 `Allow` 效果的陳述式：

- `sts:AssumeRole` 動作
- 在 `Resource` 元素中的角色的 Amazon Resource Name (ARN)。

取得政策的使用者被允許在所列資源上切換角色 (無論是透過群組成員還是直接連接)。

> **Note**
>
> 如果 `Resource` 已設定為 `*`，則使用者可以在信任使用者帳戶的任何帳戶中擔任任何角色。(換句話說，角色的信任政策會將使用者的帳戶指定為 `Principal`)。最佳實務是建議您遵循最低權限原則，並且僅為使用者所需的角色指定完整的 ARN。

以下範例顯示一個政策，該政策僅允許使用者擔任一個帳戶中的角色。此外，政策使用萬用字元 (*) 指定，如果角色名稱以字元 `Test` 開頭並後跟任何其他字元組合，則使用者只能切換到該帳戶中的角色。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::account-id:role/Test*"
  }
}
```

> **Note**
>
> 角色向使用者授予的許可不會新增到使用者已獲得的許可。當使用者切換到某個角色時，使用者可臨時放棄其原始許可以換取由該角色授予的許可。使用者退出該角色時，將自動恢復原始使用者許可。例如，如果使用者的許可允許使用 Amazon EC2 執行個體，但是角色的許可政策未授予這些許可。在這種情況下，在使用角色的同時，使用者不能使用在主控台中的 Amazon EC2 執行個體。此外，透過 `AssumeRole` 取得的暫時憑證不可透過程式設計方式使用 Amazon EC2 執行個體。

提供資訊給使用者

建立一個角色並向使用者授予切換為該角色的許可後，您必須提供使用者下列項目：

- 角色的名稱
- 包含角色的帳戶 ID 或別名

您可以向使用者發送使用帳戶 ID 和角色名稱預先配置的連結，以簡化他們的存取操作。完成建立角色精靈後，您可以選取檢視角色橫幅，或在任何跨帳戶啟用之角色的角色摘要頁面上查看角色連結。

您還可使用以下格式來手動建構連結。請用您的帳戶 ID 或別名及角色名稱來替換下列範例中的兩個參數：

```
https://signin.aws.amazon.com/switchrole?
account=your_account_ID_or_alias&roleName=optional_path/role_name
```

我們建議您將使用者導向到 切換到角色 (主控台)，以便向他們演練該過程。若要排除您在擔任角色時可能遇到的常見問題，請參閱 我無法擔任角色。

考量事項

- 如果您以程式設計方式建立角色，則可使用路徑以及名稱來建立角色。如果您這麼做，則必須為您的使用者提供完整的路徑和角色名稱以便在 AWS Management Console的 Switch Role (切換角色) 頁面上輸入。例如：`division_abc/subdivision_efg/role_XYZ`。
- 如果您以程式設計方式建立角色，則除了 `RoleName` 外，您還可以新增最長 512 個字元的 `Path`。角色名稱長度上限為 64 個字元。不過，若要將角色與中的「切換角色」功能搭配使用 AWS Management Console，則合併`Path`且`RoleName`不能超過 64 個字元。

- 基於安全考量，您可以檢閱 AWS CloudTrail 記錄檔以瞭解中執行動作的人員 AWS。您可以使用角色信任政策中的 `sts:SourceIdentity` 條件金鑰，請求使用者在擔任角色時指定身分。例如，您可以請求 IAM 使用者將自己的使用者名稱指定為其來源身分。這可以協助您判斷哪位使用者在 AWS中執行了特定動作。如需詳細資訊，請參閱 `sts:SourceIdentity`。您亦可以使用 `sts:RoleSessionName`，請求使用者在擔任角色時指定工作階段名稱。當不同的主體使用角色時，這可協助您區分角色工作階段。

授予使用者將角色傳遞至 AWS 服務的許可

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。這樣可讓該服務擔任該角色，並代表您執行動作。對於大多數服務，您只需在設定期間將角色傳遞服務一次，該服務不需要每次都要擔任角色。例如，假設您的應用程式正在 Amazon EC2 執行個體上執行。該應用程式需要暫時性憑證進行身分驗證，需要許可來授權應用程式在 AWS中執行動作。當設定應用程式時，您必須將角色傳遞到 Amazon EC2，與提供使些憑證的執行個體搭配使用。您為在執行個體上執行的應用程式定義許可，做法是將 IAM 政策連接到角色。應用程式在每次需要時擔任該角色，並在角色允許下執行動作。

若要將角色 (及其權限) 傳遞給 AWS 服務，使用者必須擁有將角色傳遞給服務的權限。這可幫助管理員確定只有核准的使用者可以透過授予許可的角色來設定服務。若要允許使用者將角色傳遞給 AWS 服務，您必須將`PassRole`權限授與使用者的 IAM 使用者、角色或群組。

> **Warning**
>
> - 您只能使用該`PassRole`權限將 IAM 角色傳遞給共用相同 AWS 帳戶的服務。若要將帳戶 A 中的角色傳遞給帳戶 B 中的服務，您必須先在帳戶 B 中建立可從帳戶 A 擔任該角色的 IAM 角色，接著帳戶 B 中的角色才可以傳遞至該服務。如需詳細資訊，請參閱 IAM 中的跨帳戶資源存取。
> - 請勿嘗試透過標記角色，然後使用政策中的 `ResourceTag` 條件金鑰搭配 `iam:PassRole` 動作來控制誰可以傳遞角色。這種方法所產生的結果並不可靠。

設定`PassRole`權限時，您應確定使用者未傳遞角色所擁有的權限超過您希望使用者擁有的角色。例如，愛麗絲可能不被允許執行任何 Amazon S3 動作。如果愛麗絲可以將角色傳遞給允許 Amazon S3 動作的服務，則該服務可以在執行任務時代表愛麗絲執行 Amazon S3 動作。

當您指定服務連結角色時，您也必須擁有將該角色傳遞到服務的許可。有些服務會自動在您在該服務中執行動作時，在您的帳戶中建立服務連結角色。例如，Amazon EC2 Auto Scaling 會在您第一次建立 Auto Scaling 群組時，為您建立 `AWSServiceRoleForAutoScaling` 服務連結角色。如果您在建立 Auto Scaling 群組時，嘗試指定服務連結角色並且您沒有 `iam:PassRole` 許可，則會收到錯誤。如果您沒有明確指定角色，則不需要 `iam:PassRole` 許可，預設情況下，對該群組執行的所有操作都使用 `AWSServiceRoleForAutoScaling` 角色。若要了解哪些服務支援服務連結角色，請參閱 AWS 與 IAM 搭配使用的服務。若要了解哪些服務在您於服務中執行動作時會自動建立服務連結角色，請選擇 Yes (是) 連結，並檢視該服務的服務連結角色文件。

使用者可以在任何 API 操作中，以參數方式傳遞角色 ARN，而該操作便是使用角色將許可指派給服務。然後，該服務會檢查該使用者是否擁有 iam:PassRole 許可。若要限制使用者只傳遞核准的角色，您可以使用 IAM 政策陳述式的 Resources 元素來篩選 iam:PassRole 許可。

您可以使用 JSON 策略中的Condition元素來測試所有 AWS 請求的請求內容中包含的鍵值。若要進一步了解如何在政策中使用條件索引鍵，請參閱 IAM JSON 政策元素：Condition。iam:PassedToService 條件索引鍵可用來指定能傳遞角色之服務的服務主體。若要進一步了解如何在政策中使用iam:PassedToService條件金鑰，請參閱 iam: PassedToService。

範例 1

假設您想要授予使用者在啟動執行個體時，能夠將任何已核准的一組角色傳遞到 Amazon EC2 服務。您需要三種元素：

- 連接到角色的 IAM 許可政策，其決定角色可以做什麼。將許可侷限在角色必須執行的動作，以及角色針對那些動作所需的資源。您可以使用受 AWS 管或客戶建立的 IAM 許可政策。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [ "A list of the permissions the role is allowed to use" ],
        "Resource": [ "A list of the resources the role is allowed to access" ]
    }
}
```

- 角色的「信任政策」，其允許服務擔任角色。例如，您可以將以下信任政策連接到角色及 UpdateAssumeRolePolicy 動作。此信任政策可讓 Amazon EC2 使用角色和連接到該角色的許可。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",
        "Effect": "Allow",
        "Principal": { "Service": "ec2.amazonaws.com" },
       "Action": "sts:AssumeRole"
    }
}
```

- 連接到 IAM 使用者的 IAM 許可政策，其允許使用者僅傳遞已核准的角色。通常是將 `iam:GetRole` 新增至 `iam:PassRole`，讓使用者可以取得要傳遞之角色的詳細資訊。在這個範例中，使用者可以僅傳遞存在於指定的帳戶，名稱開頭為 `EC2-roles-for-XYZ-` 的角色：

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::account-id:role/EC2-roles-for-XYZ-*"
    }]
}
```

現在，使用者可以使用已指派的角色來啟動 Amazon EC2 執行個體。在執行個體上執行的應用程式，可透過執行個體設定檔中繼資料來存取角色的暫時性憑證。連接到該角色的許可政策決定了執行個體可以執行哪些操作。

範例 2

Amazon Relational Database Service (Amazon RDS) 支援增強型監控功能。此功能可讓 Amazon RDS 使用代理程式來監控資料庫執行個體。它還允許 Amazon RDS 將指標記錄到 Amazon CloudWatch 日誌。若要啟用此功能，您必須建立服務角色以提供 Amazon RDS 許可來監控和寫入指標到您的日誌。

為 Amazon RDS 增強型監控建立角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 選擇 Roles (角色)，然後選擇 Create role (建立角色)。
3. 選擇 [AWS 服務角色類型]，然後針對 [其他 AWS 服務使用案例] 選擇 RDS 服務。選擇 RDS – Enhanced Monitoring (RDS – 增強型監控)，然後選擇 Next (下一步)。
4. 選擇亞馬遜 RDS EnhancedMonitoringRole 權限策略。
5. 選擇下一步。
6. 針對 Role name (角色名稱)，輸入可協助您識別此角色用途的角色名稱。角色名稱在您的 AWS 帳戶. 角色名稱用在政策中或作為 ARN 的一部分時，角色名稱區分大小寫。當主控台中的客戶顯示

角色名稱時 (例如在登入程序期間)，角色名稱不區分大小寫。因為有各種實體可能會參考此角色，所以建立角色之後，您就無法編輯其名稱。

7. (選用) 在 Description (說明) 中，輸入新角色的說明。
8. (選用) 藉由連接標籤作為鍵值對，將中繼資料新增至使用者。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
9. 檢閱角色，然後選擇 Create role (建立角色)。

角色會自動取得信任政策，授予 `monitoring.rds.amazonaws.com` 服務許可以擔任角色。在那之後，Amazon RDS 可以執行 `AmazonRDSEnhancedMonitoringRole` 政策允許的所有動作。

您想要存取增強型監控的使用者需要的政策是包含一個可讓使用者列出 RDS 角色的陳述式，以及傳遞角色的陳述式，如下所示。使用您的帳號並將角色名稱取代為您在步驟 6 提供的名稱。

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": "*"
},
{
    "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
    "Effect": "Allow",
    "Action": [ "iam:PassRole" ],
    "Resource": "arn:aws:iam::account-id:role/RDS-Monitoring-Role"
}
```

您可以將此陳述式與在另一個政策中的陳述式併在一起，或是將它放在它自己的政策中。若要改為指定使用者可以傳遞任何開頭為 RDS- 的角色，您可以將資源 ARN 中的角色名稱取代為萬用字元，如下所示。

```
    "Resource": "arn:aws:iam::account-id:role/RDS-*"
```

AWS CloudTrail 日誌中的 **iam:PassRole** 動作

`PassRole`不是一個 API 調用。 `PassRole`是一個權限，這意味著不會為 IAM 生成 CloudTrail 日誌`PassRole`。若要檢閱傳遞給哪 AWS 服務 些角色 CloudTrail，您必須檢閱建立或修改接收該角色之 AWS 資源的 CloudTrail 記錄檔。例如，角色在建立 AWS Lambda 函數時會傳遞給函數。`CreateFunction` 動作的日誌顯示有關被傳遞到函數之角色的記錄。

切換到角色 (主控台)

「角色」指定一組許可，您可以使用它來存取所需的 AWS 資源。從這個意義上說，類似於 AWS Identity and Access Management中的使用者 (IAM)。當您以使用者身分登入時，您將取得一組特定的許可。不過，您不登入角色，但一旦登入後就可以切換角色。這會暫時擱置了原始使用者許可，而不是為您提供指派給該角色的許可。該角色可以在您自己的帳戶或任何其他 AWS 帳戶中。如需有關角色、其優勢以及建立方式的詳細資訊，請參閱 IAM 角色 和 建立 IAM 角色。

> **Important**
>
> 您的使用者以及您切換到任何角色的許可都不會累計。每次只有一組許可是作用中。當您切換角色時，您會暫時放棄使用者許可並使用指派給該角色的許可。當您退出角色後，您的使用者許可會自動恢復。

當您在中切換角色時 AWS Management Console，主控台一律會使用您的原始認證來授權交換器。無論您作為 IAM 使用者、IAM Identity Center 中的使用者、SAML 聯合角色還是 Web 聯合身分角色登入，上述情形均適用。例如，如果您切換到 RoleA，IAM 會使用您的原始使用者或聯合角色憑證確定是否允許您擔任 RoleA。如果您接著在使用 RoleA 時切換至 RoleB， AWS 仍會使用原始使用者或同盟角色認證來授權交換器，而不是 RoleA 的認證。

在主控台中切換角色的注意事項

本節提供有關使用 IAM 主控台切換為角色的其他資訊。

> **備註：**
>
> - 如果您以「」的身分登入，則無法切換角色 AWS 帳戶根使用者。當您做為 IAM 使用者、IAM Identity Center 中的使用者、SAML 聯合角色或 Web 聯合身分角色登入時，才能切換角色。
> - 您無法將中的角色切換 AWS Management Console 到需要ExternalId值的角色。您只能透過呼叫支援 `ExternalId` 參數的 `AssumeRole` API 來切換到此類角色。

- 如果管理員為您提供連結，請選擇該連結，然後在以下程序中跳至步驟 Step 5。該連結將您帶到適當的網頁，並填入帳戶 ID (或別名) 和角色名稱。
- 您可以手動建構連結，然後在以下程序中跳到步驟 Step 5。若要建構您的連結，請使用以下格式：

```
https://signin.aws.amazon.com/switchrole?
account=account_id_number&roleName=role_name&displayName=text_to_display
```

在這裡取代以下文字：

- *account_id_number* – 管理員提供給您的 12 位數的帳戶識別符。或者，您的管理員可能會建立帳戶別名，以便該網址包含您的帳戶名稱而不是帳戶 ID。如需詳細資訊，請參閱《AWS 登入 使用者指南》 中的使用者類型。
- *role_name* – 您要擔任的角色名稱。您可以從角色的 ARN 結束時取得這個。例如，從以下角色 ARN 提供 TestRole 角色名稱： `arn:aws:iam::123456789012:role/TestRole`。
- (選用) *text_to_display* – 當此角色處於作用中時，您希望在導覽列上顯示的文字代替您的使用者名稱。

- 您可以使用管理員提供的資訊手動切換角色，方法如下。您可以使用管理員提供的信息通過以下步驟手動切換角色。

根據預設，當您切換角色時， AWS Management Console 工作階段會持續 1 小時。IAM 使用者工作階段預設為 12 小時。在主控台中切換角色的 IAM 使用者會被授予角色最長工作階段持續時間或使用者工作階段中的剩餘時間，以較短者為準。例如，假設為角色設定的最大工作階段持續時間為 10 小時。IAM 使用者決定切換至角色時已登入主控台 8 小時。使用者工作階段還剩餘 4 小時，因此允許的角色工作階段持續時間為 4 小時。下表顯示在主控台中切換角色時如何判斷 IAM 使用者的工作階段持續時間。

IAM 使用者主控台角色工作階段持續時間

IAM 使用者工作階段剩餘時間為...	角色工作階段持續時間為...		
小於角色最大工作階段持續時間	使用者工作階段中的剩餘時間		
大於角色最大工作階段持續時間	最大工作階段持續時間值		
等於角色最大工作階段持續時間	最大工作階段持續時間值 (近似值)		

> **Note**
>
> 有些 AWS 服務主控台可以在您的角色工作階段到期時自動續約，而無需您採取任何動作。有些主控台可能會提示您重新載入瀏覽器頁面以重新驗證您的工作階段。

若要排除您在擔任角色時可能遇到的常見問題，請參閱 我無法擔任角色。

切換到角色 (主控台)

1. 以 IAM 使用者身分登入，然後在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。 AWS Management Console
2. 在 IAM 主控台中，在右上角的導覽列中選擇您的使用者名稱。它通常如下：*username*@*account_ID_number_or_alias*。
3. 選擇 Switch Role (切換角色)。如果這是第一次選擇此選項，則頁面會顯示更多資訊。讀取後，選擇 Switch Role (切換角色)。如果清除瀏覽器 cookie，則此頁面可以再次顯示
4. 在 Switch Role (切換角色) 頁面上，輸入帳戶 ID 號碼或帳戶別名以及管理員提供的角色名稱。

   > **Note**
   >
   > 如果管理員已使用路徑 (例如 `division_abc/subdivision_efg/roleToDoX`) 建立角色，則必須在 Role (角色)) 方塊中輸入該完整路徑和名稱。如果僅輸入角色名稱，或者組合的 `Path` 和 `RoleName` 超過 64 個字元，則角色切換失敗。這是存放角色名稱的瀏覽器 cookie 的限制。如果發生這種情況，請聯絡您的管理員，並要求他們減小路徑和角色名稱的大小。

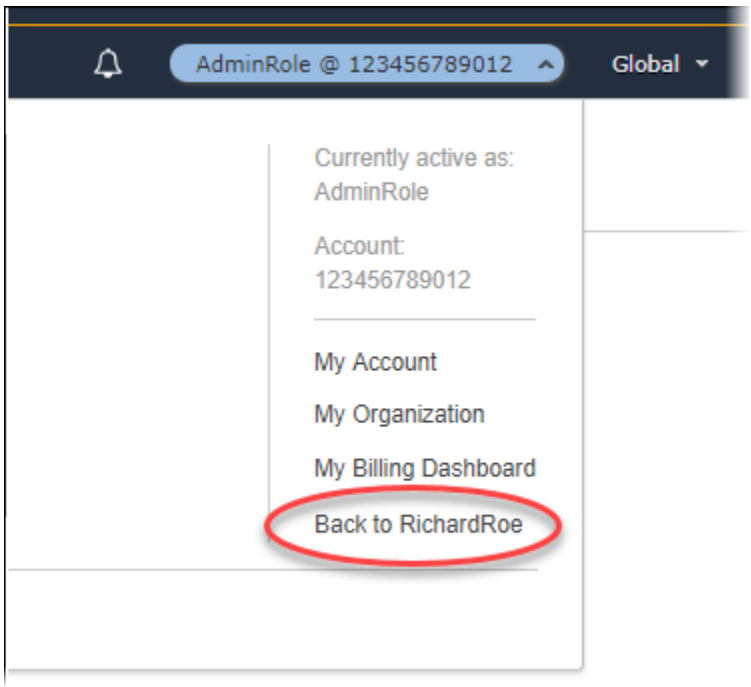
5. (選用) 選擇 Display name (顯示名稱)。當此角色處於作用中時，輸入要在導覽列上顯示的文字代替您的使用者名稱。根據帳戶和角色資訊建議使用名稱，但您可以將其變更為對您有意義的任何名稱。您也可以選擇顏色反白顯示名稱。名稱和顏色有助於在此角色處於作用中時提醒您，這會變更您的許可。例如，對於允許您存取測試環境的角色，您可以指定 **`Test`** 的 Display name (顯示名稱) 並在 Color (顏色)中選擇綠色。對於允許您存取生產的角色，您可以指定 **`Production`** 的 Display name (顯示名稱) 並在 Color (顏色)中選擇紅色。
6. 選擇 Switch Role (切換角色)。顯示名稱和顏色將替換導覽列上的使用者名稱，您可以開始使用該角色授予您的許可。

> **秘訣**
>
> 您使用的最後幾個角色會顯示在選單。下次需要切換到其中一個角色時，您只需選擇所需的角色。如果角色未顯示在選單上，則只需手動輸入帳戶和角色資訊。

要停止使用角色 (主控台)

1. 在 IAM 主控台中，選擇導覽列右上角的角色 Display Name (顯示名稱)。它通常如下：*rolename*@*account_ID_number_or_alias*。
2. 選擇返回*#####*。停用角色及其許可，並自動恢復與 IAM 使用者和群組關聯的許可。

   例如，假設您使用使用者名稱 123456789012 登入到帳戶編號 RichardRoe。使用過 AdminRole 角色後，您想要停止使用該角色並傳回您的原始許可。若要停止使用角色，請選擇 AdminRole @ 123456789012，然後選擇 [返回至]。 RichardRoe



切換到 IAM 角色 (AWS CLI)

「角色」指定一組許可，您可以使用它來存取所需的 AWS 資源。從這個意義上說，類似於 AWS Identity and Access Management中的使用者 (IAM)。當您以使用者身分登入時，您將取得一組特定的許可。不過，您不登入角色，但以使用者身分登入後，就可以切換角色。這會暫時擱置了原始使用者許可，而不是為您提供指派給該角色的許可。該角色可以在您自己的帳戶或任何其他 AWS 帳戶中。如需

有關角色、其優勢以及建立和設定方式的詳細資訊，請參閱 IAM 角色 和 建立 IAM 角色。要了解在擔任角色時使用的各種方法，請參閱 使用 IAM 角色。

> **Important**
>
> 您的 IAM 使用者以及您擔任的任何角色的許可都不會累計。每次只有一組許可是作用中。當您擔任角色時，您會暫時放棄以前的使用者或角色許可，並使用指派給該角色的許可。當您退出角色後，您的使用者許可會自動恢復。

當您以 IAM 使用者身分登入時，您可以使用角色執行 AWS CLI 命令。當您以已使用角色的外部驗證使用者 (SAML 或 OIDC) 身分登入時，您也可以使用角色來執行 AWS CLI 命令。此外，您可以使用角色從 Amazon EC2 執行個體執行 AWS CLI 命令，而該執行個體透過執行個體描述檔連接到角色。當您以 AWS 帳戶根使用者身分登入時，則無法擔任該角色。

角色鏈結 – 您也可以使用角色鏈結，以使用角色的許可來存取第二個角色。

在預設情況下，您的角色工作階段持續一小時。使用 `assume-role*` CLI 操作擔任此角色時，可以為 `duration-seconds` 參數指定值。此值的範圍可以從 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定的值。如果在主控台中切換角色，您工作階段的持續時間會限制為最多一小時。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。

如果使用角色鏈結時，則工作階段持續時間最多限制為一小時。如果您隨後使用 `duration-seconds` 參數提供大於一小時的值，則操作會失敗。

範例案例：切換到生產角色

假設您是在開發環境中工作的 IAM 使用者。在此情況下，您偶爾需要透過 AWS CLI 在命令列中使用生產環境。您已經有存取金鑰憑證組可供您使用。這可以是指派給標準 IAM 使用者的存取金鑰對。或者，如果您以聯合身分使用者的身分登入，則它可以是最初指派給您的角色的存取金鑰對。如果您目前的許可授予您擔任特定 IAM 角色的能力，則可以在設定檔的「設 AWS CLI 定檔」中識別該角色。然後使用指定 IAM 角色的許可來執行該命令，而不是原始身分。請注意，當您在指 AWS CLI 令中指定該設定檔時，您正在使用新角色。在這種情況下，您無法同時在開發帳戶中使用原始許可。原因是一次只有一組許可有效。

> **Note**
>
> 基於安全考量，管理員可以檢閱 AWS CloudTrail 記錄檔，以瞭解中執行動作的人員 AWS。當您擔任角色時，系統管理員可能需要您指定來源身分或角色工作階段名稱 。如需詳細資訊，請參閱 sts:SourceIdentity 及 sts:RoleSessionName。

切換到生產角色 (AWS CLI)

1. 如果您從未使用過 AWS CLI，則必須先設定預設 CLI 設定檔。開啟命令提示字元並將 AWS CLI 安裝設定為使用 IAM 使用者或聯合身分角色的存取金鑰。如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的設定 AWS Command Line Interface。

   執行 aws configure 命令，如下所示：

   ```
   aws configure
   ```

   當出現提示時，請提供下列資訊：

   ```
   AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: us-east-2
   Default output format [None]: json
   ```

2. 在 Unix 或 Linux .aws/config 檔案中，或在 Windows C:\Users\USERNAME\.aws\config 檔案中建立角色的新描述檔。下列範例會建立稱為 prodaccess 的描述檔，以切換到 *ProductionAccessRole* 帳戶中的角色 123456789012。您從建立角色的帳戶管理員處取得角色 ARN。呼叫此設定檔時，會 AWS CLI 使用的認證來source_profile要求角色的認證。因此，做為 source_profile 參考的身分必須具有 sts:AssumeRole 中指定角色的 role_arn 許可。

   ```
   [profile prodaccess]
       role_arn = arn:aws:iam::123456789012:role/ProductionAccessRole
       source_profile = default
   ```

3. 建立新設定檔後，任何指定參數的 AWS CLI 命令都會在附加至 IAM 角色的許可下--profile prodaccess執行，ProductionAccessRole而非預設使用者。

   ```
   aws iam list-users --profile prodaccess
   ```

如果指派給 ProductionAccessRole 的許可啟用列出目前 AWS 帳戶中的使用者，則此命令有效。

4. 若要傳回原始憑證授予的許可，請執行不帶 --profile 參數的命令。 AWS CLI 恢復為使用您在中配置的默認配置文件中的身份Step 1證明。

如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的擔任角色。

範例案例：允許執行個體描述檔角色來切換到另一個帳戶中的角色

想像一下，您正在使用兩個 AWS 帳戶，並且希望允許在 Amazon EC2 實例上運行的應用程序在兩個帳戶中運行AWS CLI命令。假設 EC2 執行個體存在於帳戶 111111111111。該執行個體包含 abcd 執行個體描述檔角色，該角色會允許應用程式在相同 111111111111 帳戶中對 my-bucket-1 儲存貯體執行唯讀 Amazon S3 任務。不過，也必須允許應用程式擔任 efgh 跨帳戶角色來在帳戶 222222222222 中執行任務。若要執行此操作，abcd EC2 執行個體描述檔角色必須有以下許可政策：

帳戶 111111111111 ***abcd*** 角色許可政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::my-bucket-1/*",
                "arn:aws:s3:::my-bucket-1"
            ]
        },
        {
            "Sid": "AllowIPToAssumeCrossAccountRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::222222222222:role/efgh"
        }
    ]
}
```

假設 efgh 跨帳戶角色允許在相同 222222222222 帳戶中對 my-bucket-2 儲存貯體的唯讀 Amazon S3 任務。若要執行此操作，efgh 跨帳戶角色必須有以下許可政策：

帳戶 222222222222 ***efgh*** 角色許可政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-2/*",
                "arn:aws:s3:::my-bucket-2"
            ]
```

```
        }
    ]
}
```

efgh 角色必須允許 abcd 執行個體設定檔角色擔任該角色。若要執行此操作，efgh 角色必須有以下信任政策：

帳戶 222222222222 ***efgh*** 角色信任政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "efghTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
        }
    ]
}
```

然後，若要在帳戶中執行 AWS CLI 命令222222222222，您必須更新 CLI 組態檔。在 AWS CLI 組態檔案中，將 efgh 角色識別為「描述檔」且 abcd EC2 執行個體描述檔角色識別為「憑證來源」。然後，使用 efgh 角色 (而不是原始 abcd 角色) 的許可執行 CLI 命令。

> **Note**
>
> 基於安全性考量，您可以用 AWS CloudTrail 來稽核帳戶中角色的使用情況。若要區分 CloudTrail 記錄中不同主參與者使用角色時的角色工作階段，您可以使用角色工作階段名稱。如果如本主題所述，代表使用者 AWS CLI 擔任角色時，會自動將角色工作階段名稱建立為AWS-CLI-session-*nnnnnnnn*。以下 *nnnnnnnn* 是整數，代表 Unix epoch 時間中的時間 (自 1970 年 1 月 1 日午夜 UTC 的秒數)。如需詳細資訊，請參閱《使用指南》中的AWS CloudTrail 〈CloudTrail 事件參考〉。

讓 EC2 執行個體設定檔角色切換到跨帳戶角色 (AWS CLI)

1. 您不需要設定預設 CLI 描述檔。您反而可以從 EC2 執行個體描述檔中繼資料載入憑證。在 .aws/config 檔案中為角色建立新的設定檔。下列範例會建立 instancecrossaccount 的描述檔，以切換到 *efgh* 帳戶中的角色 222222222222。呼叫此描述檔時， AWS CLI 會使用 EC2 執

行個體描述檔中繼資料的憑證來請求角色的憑證。因此，EC2 執行個體設定檔角色必須具有在 `sts:AssumeRole` 中指定角色的 `role_arn` 許可。

```
[profile instancecrossaccount]
role_arn = arn:aws:iam::222222222222:role/efgh
credential_source = Ec2InstanceMetadata
```

2. 建立新設定檔之後，任何指定參數的 AWS CLI 命令都會在附加至帳戶中`efgh`角色的權限下`--profile instancecrossaccount`執行`222222222222`。

```
aws s3 ls my-bucket-2 --profile instancecrossaccount
```

如果指派給 `efgh` 角色的許可允許列出目前 AWS 帳戶中的使用者，則此命令有效。

3. 若要在帳戶 `111111111111` 中返回原始 EC2 執行個體設定檔許可，在不使用 `--profile` 參數的情形下執行 CLI 命令。

如需詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的擔任角色。

切換至 IAM 角色 (適用於視窗的工具 PowerShell)

「角色」指定一組許可，您可以使用它來存取所需的 AWS 資源。從這個意義上說，類似於 AWS Identity and Access Management中的使用者 (IAM)。當您以使用者身分登入時，您將取得一組特定的許可。不過，您不登入角色，但一旦登入後就可以切換角色。這會暫時擱置了原始使用者許可，而不是為您提供指派給該角色的許可。該角色可以在您自己的帳戶或任何其他 AWS 帳戶中。如需有關角色、其優勢以及建立和設定方式的詳細資訊，請參閱 IAM 角色 和 建立 IAM 角色。

> **Important**
>
> 您的 IAM 使用者以及您切換到任何角色的許可都不會累計。每次只有一組許可是作用中。當您切換角色時，您會暫時放棄使用者許可並使用指派給該角色的許可。當您退出角色後，您的使用者許可會自動恢復。

本節說明在 AWS Tools for Windows PowerShell命令列中如何切換角色。

假設您在開發環境中有一個帳戶，並且偶爾需要在命令列使用 Windows 的 Tools 使用生產環境 PowerShell。您已經有一個存取金鑰憑證組可供您使用。這些可以是指派給標準 IAM 使用者的存取金鑰對。或者，如果您以聯合身分使用者登入，則它們可以是最初指派給您的角色的存取金鑰對。您可以使用這些憑證來執行 `Use-STSRole` cmdlet，該 cmdlet 將新角色的 ARN 做為參數傳送。該命令傳回

所請求角色的臨時安全憑證。然後，您可以在具有角色權限的後續 PowerShell命令中使用這些認證，以存取生產環境中的資源。使用該角色時，您無法在開發帳戶中使用您的使用者許可，因為一次只能有一組許可有效。

> **Note**
>
> 基於安全考量，管理員可以檢閱 AWS CloudTrail 記錄檔，以瞭解中執行動作的人員 AWS。當您擔任角色時，系統管理員可能需要您指定來源身分或角色工作階段名稱 。如需詳細資訊，請參閱 `sts:SourceIdentity` 及 `sts:RoleSessionName`。

請注意，所有存取金鑰和權杖僅為範例，不能如下所示般使用。以您實際環境中的適當值取代。

若要切換到角色 (視窗工具 PowerShell)

1. 開啟 PowerShell 命令提示字元並將預設設定檔設定為使用目前 IAM 使用者或聯合身分角色的存取金鑰。如果您以前使用過 Windows 的工具 PowerShell，那麼這很可能已經完成。請注意，只有在以 IAM 使用者身分 (而非 AWS 帳戶根使用者) 登入時才能切換角色。

   ```
   PS C:\> Set-AWSCredentials -AccessKey AKIAIOSFODNN7EXAMPLE -
   SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY -StoreAs MyMainUserProfile
   PS C:\> Initialize-AWSDefaults -ProfileName MyMainUserProfile -Region us-east-2
   ```

   如需詳細資訊，請參閱《使用指南》中的AWS Tools for Windows PowerShell 〈使用 AWS 認證〉。

2. 若要擷取新角色的憑證，請執行下列命令以切換到 123456789012 帳戶中的 *RoleName* 角色。您從建立角色的帳戶管理員處取得角色 ARN。此命令還需要您提供工作階段名稱。您可以為此選擇任何文字。以下命令請求憑證，然後從傳回的結果物件中擷取 `Credentials` 屬性物件，並將其存放在 `$Creds` 變數中。

   ```
   PS C:\> $Creds = (Use-STSRole -RoleArn "arn:aws:iam::123456789012:role/RoleName" -
   RoleSessionName "MyRoleSessionName").Credentials
   ```

   `$Creds` 是一個物件，現在包含您在下列步驟中所需的 `AccessKeyId`、 `SecretAccessKey` 和 `SessionToken` 元素。以下範例命令說明典型的值：

   ```
   PS C:\> $Creds.AccessKeyId
   AKIAIOSFODNN7EXAMPLE
   ```

```
PS C:\> $Creds.SecretAccessKey
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

PS C:\> $Creds.SessionToken
AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvSRyh0FW7jEXAMPLEW+vE/7s1HRp
XviG7b+qYf4nD00EXAMPLEmj4wxS04L/uZEXAMPLECihzFB5lTYLto9dyBgSDyEXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPy
Oj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1A0SKFEXAMPLEiyw
C
s8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP+4eZScEXAMPLEsnf87eNhyDHq6ikBQ==

PS C:\> $Creds.Expiration
Thursday, June 18, 2018 2:28:31 PM
```

3. 若要將這些憑證用於任何後續命令，請將它們包含在 `-Credential` 參數中。例如，以下命令使用角色中的憑證，僅在角色被授予 `iam:ListRoles` 許可且因此可以執行 `Get-IAMRoles` cmdlet 時才起作用：

   ```
   PS C:\> get-iamroles -Credential $Creds
   ```

4. 若要返回原始認證，只要停止使用`-Credentials $Creds`參數，並 PowerShell 允許還原為預設設定檔中儲存的認證即可。

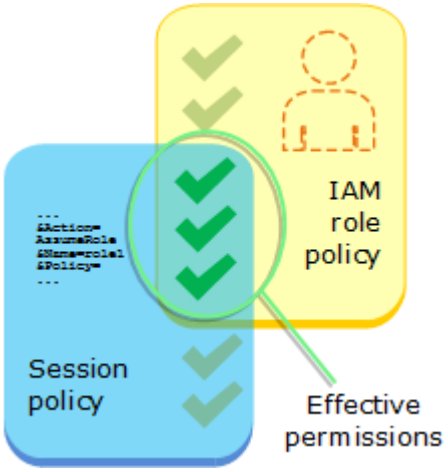
切換到身分與存取權管理角色 (AWS API)

「角色」指定一組許可，您可以使用它來存取 AWS 資源。從這個意義上說，類似於 IAM 使用者。主體 (人員或應用程式) 假設角色可接收執行必要工作並與 AWS 資源互動的臨時權限。該角色可以在您自己的帳戶或任何其他 AWS 帳戶中。如需有關角色、其優勢以及建立和設定方式的詳細資訊，請參閱 IAM 角色 和 建立 IAM 角色。要了解在擔任角色時使用的各種方法，請參閱 使用 IAM 角色。

> **Important**
>
> 您的 IAM 使用者以及您擔任的任何角色的許可都不會累計。每次只有一組許可是作用中。當您擔任角色時，您會暫時放棄以前的使用者或角色許可，並使用指派給該角色的許可。當您退出角色後，原本的許可會自動恢復。

若要擔任角色，應用程式會呼叫 AWS STS AssumeRoleAPI 作業，並傳遞要使用之角色的 ARN。此操作會建立使用臨時憑證的新工作階段。此工作階段的許可，即為以身分為基礎的政策指派給角色的許可。

當您呼叫 AssumeRole 時，您可以選擇性地傳遞內嵌或受管工作階段政策。工作階段政策是一種進階政策，且您會在以程式設計方式建立角色或聯合身分使用者的臨時憑證工作階段時，以參數方式傳遞。您可以使用 Policy 參數傳遞單一 JSON 內嵌工作階段政策文件。您可以使用 PolicyArns 參數，指定多達 10 個受管工作階段政策。所產生工作階段的許可會是實體的身分類型政策和工作階段政策的交集。當您需要將角色的臨時憑證提供給某人時，工作階段政策便可派上用場。他們可以在後續 AWS API 呼叫中，使用角色的臨時憑證來存取擁有該角色的帳戶中的資源。您無法使用工作階段政策來授予超出即將以身分為基礎政策所允許的許可。若要深入瞭解如何 AWS 決定角色的有效權限，請參閱政策評估邏輯。



當您以 IAM 使用者身分登入時，或者作為已使用角色的外部驗證使用者 (SAML 或 OIDC)，您可以呼叫 AssumeRole。您還可以使用角色鏈結，它使用角色來擔任第二個角色。當您以 AWS 帳戶根使用者身分登入時，則無法擔任該角色。

在預設情況下，您的角色工作階段持續一小時。當您使用 AWS STS AssumeRole*API 作業假設此角色時，您可以指定DurationSeconds參數的值。此值的範圍可以從 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定的值。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。

如果使用角色鏈結時，則工作階段最多限制為一小時。如果您隨後使用 DurationSeconds 參數提供大於一小時的值，則操作會失敗。

> **Note**
>
> 基於安全考量，管理員可以檢閱 AWS CloudTrail 記錄檔，以瞭解中執行動作的人員 AWS。當您擔任角色時，系統管理員可能需要您指定來源身分或角色工作階段名稱 。如需詳細資訊，請參閱 `sts:SourceIdentity` 及 `sts:RoleSessionName`。

下列程式碼範例示範如何建立使用者並擔任角色。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立沒有許可的使用者。
- 建立一個可授予許可的角色，以列出帳戶的 Amazon S3 儲存貯體。
- 新增政策，讓使用者擔任該角色。
- 使用暫時憑證，擔任角色並列出 Amazon S3 儲存貯體，然後清理資源。

.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。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
```

```
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
 groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
 AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
```

```
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
```

```
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
 { GroupName = groupName });
        return response.Group;
    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
```

```
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
 { UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
```

```
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
   {
       var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
       {
           AccessKeyId = accessKeyId,
           UserName = userName,
       });

       return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
   }


   /// <summary>
   /// Delete an IAM group.
   /// </summary>
   /// <param name="groupName">The name of the IAM group to delete.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> DeleteGroupAsync(string groupName)
   {
       var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
       return response.HttpStatusCode == HttpStatusCode.OK;
   }


   /// <summary>
   /// Delete an IAM policy associated with an IAM group.
   /// </summary>
   /// <param name="groupName">The name of the IAM group associated with the
   /// policy.</param>
   /// <param name="policyName">The name of the policy to delete.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
   {
       var request = new DeleteGroupPolicyRequest()
       {
           GroupName = groupName,
           PolicyName = policyName,
       };
```

```
        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
 DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
```

```
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
```

```
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
```

```
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
```

```
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
 ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
```

```
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
```

```
        };

        var response = await
 _IAMService.RemoveUserFromGroupAsync(removeUserRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
 policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
```

```
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
```

```
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        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<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
```

```
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
```

```
                $" \"AWS\": \"{userArn}\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();
```

```
        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
```

```
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
complete.");
        Console.WriteLine("Please wait while we clean up the resources we
created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();
```

```
        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
```

```
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }
```

```
    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
```

```
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
(IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
```

```
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

Bash

AWS CLI 與 Bash 腳本

> **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。

```
###############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
```

```
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
```

```
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
```

```
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
  policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created  IAM policy named $policy_name"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
  fi

  local assume_role_policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"$role_arn\"}]}"

  local assume_role_policy_name
  assume_role_policy_name=$(generate_random_name "test-assume-role-")

  # shellcheck disable=SC2181
  local assume_role_policy_arn
  assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
```

```
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
   return 1
 fi

 echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
 sleep 10
 echo_repeat "*" 88
 echo

 echo "Try to list buckets without the new user assuming the role."
 echo_repeat "*" 88
 echo

 # Set the environment variables for the created user.
 # bashsupport disable=BP2001
 export AWS_ACCESS_KEY_ID=$key_name
 # bashsupport disable=BP2001
 export AWS_SECRET_ACCESS_KEY=$key_secret

 local buckets
 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. This should not have
happened."
 else
   errecho "Because the role with permissions has not been assumed, listing
buckets failed."
 fi

 echo
 echo_repeat "*" 88
 echo "Now assume the role $iam_role_name and list the buckets."
 echo_repeat "*" 88
 echo

 local credentials
```

```
 credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
 # bashsupport disable=BP2001
 export AWS_SESSION_TOKEN=${credentials[2]}

 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. Listing buckets
succeeded because of "
   echo "the assumed role."
 else
   errecho "Failed to list buckets. This should not happen."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   export AWS_SESSION_TOKEN=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 local result=0
 export AWS_ACCESS_KEY_ID=""
 export AWS_SECRET_ACCESS_KEY=""
```

```
  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

此案例中使用的 IAM 函數。

```
###############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)
```

```
  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }
```

```
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi
```

```
  echo "$response"

  return 0
}

#############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
#############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
```

```
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
```

```
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
```

```
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
```

```
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
```

```
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

```
##############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
```

```
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an WS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
```

```
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an WS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
```

```
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Username:   $user_name"
  iecho "    Access key:   $access_key"
  iecho ""

  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.
```

```
 # bashsupport disable=BP5008
 function usage() {
   echo "function iam_delete_user"
   echo "Deletes an WS Identity and Access Management (IAM) user. You must
supply a username:"
   echo "  -u user_name    The name of the user."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "u:h" option; do
   case "${option}" in
     u) user_name="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$user_name" ]]; then
   errecho "ERROR: You must provide a username with the -u parameter."
   usage
   return 1
 fi

 iecho "Parameters:\n"
 iecho "    User name:   $user_name"
 iecho ""

 # If the user does not exist, we don't want to try to delete it.
 if (! iam_user_exists "$user_name"); then
   errecho "ERROR: A user with that name does not exist in the account."
   return 1
 fi

 response=$(aws iam delete-user \
   --user-name "$user_name")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
```

```
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
 std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
```

```
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

    request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
```

```
                    outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;
```

```
                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
```

```
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
```

```
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
 s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
 LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
 std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
```

```
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
```

```
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for C++ API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策

- PutUser政策

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。

```
// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper actions.PolicyWrapper
 roleWrapper actions.RoleWrapper
 userWrapper actions.UserWrapper
 questioner demotools.IQuestioner
 helper IScenarioHelper
 isTestRun bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
```

```
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
  helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:   sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser()
 accessKey := scenario.CreateAccessKey(user)
 role := scenario.CreateRoleAndPolicies(user)
```

```
 noPermsConfig := scenario.ListBucketsWithoutPermissions(accessKey)
 scenario.ListBucketsWithAssumedRole(noPermsConfig, role)
 scenario.Cleanup(user, role)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser() *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(userName)
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(user *types.User)
 *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(*user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}
```

```
// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(user *types.User)
 *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err :=
 scenario.roleWrapper.CreateRole(scenario.helper.GetName(), *user.Arn)
 if err != nil {panic(err)}
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
 if err != nil {panic(err)}
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(*listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {panic(err)}
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(*user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {panic(err)}
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
```

```
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(accessKey
 *types.AccessKey) *aws.Config {
  log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
  scenario.questioner.Ask("Press Enter when you're ready.")
  noPermsConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
 ))
  if err != nil {panic(err)}

  // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

  s3Client := s3.NewFromConfig(noPermsConfig)
  _, err = s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
  if err != nil {
   // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
  } else {
   log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
 }
 log.Println(strings.Repeat("-", 88))
  return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
```

```
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//   the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
 list the
//    buckets.
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
 generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(noPermsConfig
 *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(context.TODO(),
 &sts.AssumeRoleInput{
  RoleArn:           role.Arn,
  RoleSessionName:   aws.String("AssumeRoleExampleSession"),
  DurationSeconds:   aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {panic(err)}

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
```

```
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(user *types.User, role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
   policies, err := scenario.roleWrapper.ListAttachedRolePolicies(*role.RoleName)
   if err != nil {panic(err)}
   for _, policy := range policies {
    err = scenario.roleWrapper.DetachRolePolicy(*role.RoleName,
 *policy.PolicyArn)
    if err != nil {panic(err)}
    err = scenario.policyWrapper.DeletePolicy(*policy.PolicyArn)
    if err != nil {panic(err)}
    log.Printf("Detached policy %v from role %v and deleted the policy.\n",
     *policy.PolicyName, *role.RoleName)
   }
   err = scenario.roleWrapper.DeleteRole(*role.RoleName)
   if err != nil {panic(err)}
   log.Printf("Deleted role %v.\n", *role.RoleName)

   userPols, err := scenario.userWrapper.ListUserPolicies(*user.UserName)
   if err != nil {panic(err)}
   for _, userPol := range userPols {
    err = scenario.userWrapper.DeleteUserPolicy(*user.UserName, userPol)
    if err != nil {panic(err)}
    log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
   }
   keys, err := scenario.userWrapper.ListAccessKeys(*user.UserName)
   if err != nil {panic(err)}
   for _, key := range keys {
    err = scenario.userWrapper.DeleteAccessKey(*user.UserName, *key.AccessKeyId)
    if err != nil {panic(err)}
```

```
    log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
   }
   err = scenario.userWrapper.DeleteUser(*user.UserName)
   if err != nil {panic(err)}
   log.Printf("Deleted user %v.\n", *user.UserName)
   log.Println(strings.Repeat("-", 88))
 }

}
```

定義包裝帳號動作的結構。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
 error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

```
// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
 error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

定義包裝政策動作的結構。

```
// 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"`
}



// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
```

```
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
 error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(context.TODO(),
 &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
```

```
 result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
 &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
 _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

定義包裝角色動作的結構。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(context.TODO(),
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
 (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action: []string{"sts:AssumeRole"},
  }},
```

```
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(context.TODO(),
 &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(context.TODO(),
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
 description string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
 &iam.CreateServiceLinkedRoleInput{
```

```
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
 error {
 _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
 &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

```
// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
 ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
 error {
 _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
 &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
 &iam.ListRolePoliciesInput{
```

```
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

定義包裝使用者動作的結構。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
```

```
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(context.TODO(),
 &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
 actions []string,
  roleArn string) error {
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
 &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
```

```
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
 error {
 _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
 &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
 _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
```

```
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
 &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
 &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
```

```
func (wrapper UserWrapper) ListAccessKeys(userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
 &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

建立可包裝 IAM 使用者動作的函數。

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
```

```
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

   public static String userArn;

   public static void main(String[] args) throws Exception {

       final String usage = """

               Usage:
                   <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

               Where:
                   username - The name of the IAM user to create.\s
                   policyName - The name of the policy to create.\s
                   roleName - The name of the role to create.\s
                   roleSessionName - The name of the session required for the
assumeRole operation.\s
                   bucketName - The name of the Amazon S3 bucket from which
objects are read.\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String userName = args[0];
       String policyName = args[1];
       String roleName = args[2];
       String roleSessionName = args[3];
       String bucketName = args[4];

       Region region = Region.AWS_GLOBAL;
       IamClient iam = IamClient.builder()
               .region(region)
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the AWS IAM example scenario.");
       System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

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

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();
```

```
            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

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

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
            return response.role().arn();

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

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
```

```
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

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

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }
```

```
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

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

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();
```

```
            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn)
{

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(polArn)
                    .roleName(roleName)
                    .build();

            iam.detachRolePolicy(rolePolicyRequest);
```

```
            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(polArn)
                    .build();

            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

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

    public static void deleteKey(IamClient iam, String username, String
accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

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

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
```

```
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException 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。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "test_name";
const policyName = "test_policy";
const roleName = "test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
```

```
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    } else {
      throw caught;
    }
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
 (AWS STS).
  // It's not best practice to use access keys. For more information, see
 https://aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
```

```
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;

  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
```

```
              Principal: {
                // Allow the previously created user to assume this role.
                AWS: User.Arn,
              },
              Action: "sts:AssumeRole",
            },
          ],
        }),
        RoleName: roleName,
      }),
    ),
);

if (!Role) {
  throw new Error("Role not created");
}

// Create a policy that allows the user to list S3 buckets.
const { Policy: listBucketPolicy } = await iamClient.send(
  new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: ["s3:ListAllMyBuckets"],
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
```

```
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 60 }, () =>
  listBuckets(s3Client),
);
```

```
  // Clean up.
  await iamClient.send(
    new DetachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeletePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }
```

```
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

建立可包裝 IAM 使用者動作的函數。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>
```

```
    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}
```

```
suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?
): String? {
```

```
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
```

```
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String
) {
    val stsClient =
        StsClient {
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
```

```
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
```

```
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccess關鍵
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUser政策
- DetachRole政策
- PutUser政策

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。

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("---------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("---------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
```

```
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
```

```
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- 如需 API 詳細資訊，請參閱《AWS SDK for PHP API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccess關鍵
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUser政策
- DetachRole政策
- PutUser政策

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。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
```

```
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)
```

```
    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
```

```
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
 why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
 "
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
```

```
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are
 owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
 role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
```

```
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
 credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
```

```
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
```

```
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- 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。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Tried and failed to create demo user.")
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
 user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
```

```
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Principal: {'AWS': user.arn},
                   Action: "sts:AssumeRole"
                 }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
```

```
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Action: "s3:ListAllMyBuckets",
                   Resource: "arn:aws:s3:::*"
                 }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Action: "sts:AssumeRole",
                   Resource: role.arn
                 }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
```

```
    puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated
into a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
  # Because the resource uses credentials with limited access, it may not be able
to
  # list the S3 buckets.
  #
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def list_buckets(s3_resource)
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == "AccessDenied"
      puts("Attempt to list buckets with no permissions: AccessDenied.")
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
```

```
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end

 # Deletes a role. If the role has policies attached, they are detached and
 # deleted before the role is deleted.
 #
 # @param role_name [String] The name of the role to delete.
 def delete_role(role_name)
   @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
     @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
     @iam_client.delete_policy(policy_arn: policy.policy_arn)
     @logger.info("Detached and deleted policy #{policy.policy_name}.")
   end
   @iam_client.delete_role({ role_name: role_name })
   @logger.info("Role deleted: #{role_name}.")
 rescue Aws::Errors::ServiceError => e
```

```
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
 deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
```

```
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Ruby API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
```

```
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;
```

```
    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
```

```
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(&format!("{}{}", "iam_demo_assumerole_session_",
uuid))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
```

```
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
```

```
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的AWS SDK API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可

在 Amazon EC2 執行個體上執行的應用程式必須在 AWS API 請求中包含 AWS 登入資料。您可以讓開發人員將 AWS 登入資料直接存放在 Amazon EC2 執行個體中，並允許該執行個體中的應用程式使用這些登入資料。不過，開發人員將必須管理憑證，並確實將憑證安全地傳遞給每個執行個體，並在需要更新憑證時更新每個 Amazon EC2 執行個體。這是許多額外的工作。

與其這麼做，您可以 (也應該) 使用 IAM 角色來管理 Amazon EC2 執行個體上所執行應用程式的臨時憑證。使用角色時，您不必將長期憑證 (例如登入憑證或存取金鑰) 分發給 Amazon EC2 執行個體。相反

地，該角色會提供應用程式在呼叫其他 AWS 資源時可以使用的暫存權限。啟動 Amazon EC2 執行個體時，指定要與執行個體相關聯的 IAM 角色。然後，在執行個體上執行的應用程式可以使用角色提供的暫時憑證來簽署 API 請求。

如要透過角色為 Amazon EC2 執行個體上執行的應用程式授予許可，需搭配一些額外的組態。在 Amazon EC2 執行個體上執行的應用程式是 AWS 由虛擬化作業系統抽象而來。由於這種額外的分離，您需要執行額外的步驟，將 AWS 角色及其關聯許可指派給 Amazon EC2 執行個體，並使其可用於其應用程式。此額外步驟是建立連接到執行個體的執行個體設定檔。執行個體描述檔包含角色，可以將角色的臨時憑證提供給在執行個體上執行的應用程式。然後，可以在應用程式的 API 呼叫中使用這些臨時憑證來存取資源，並限制僅存取角色指定的那些資源。

> **Note**
>
> 一次只能指派一個角色給 Amazon EC2 執行個體，執行個體上的所有應用程式會共用相同的角色和許可。當您利用 Amazon ECS 來管理 Amazon EC2 執行個體時，您可以向 Amazon ECS 任務指派角色，這些角色和執行 Amazon EC2 執行個體所使用的角色有所區別。為每項任務指派一個角色的做法符合最低權限存取原則，並且允許對動作和資源進行更精細的控制。
> 如需詳細資訊，請參閱《Amazon Elastic Container Service 最佳實務指南》中的對 Amazon ECS 任務使用 IAM 角色。

以這種方式使用角色有幾個好處。由於角色憑證是臨時的並且是自動更新的，因此您不需要管理憑證，也不必擔心長期的安全風險。此外，如果對多個執行個體使用單一角色，則可以對該角色進行變更，並將變更自動傳播到所有執行個體。

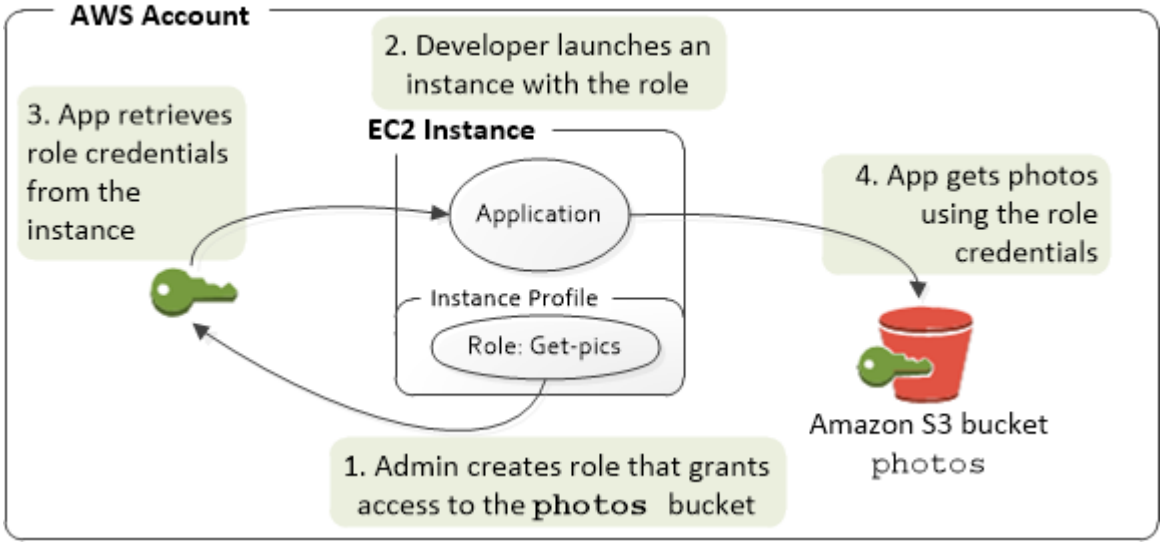
> **Note**
>
> 雖然在啟動角色時通常就會將角色指派給 Amazon EC2 執行個體，但角色也可以與目前執行的 Amazon EC2 執行個體連接。若要了解如何將角色連接到正在執行的執行個體，請參閱 Amazon EC2 的 IAM 角色。

主題

- 使用執行個體設定檔

適用於 Amazon EC2 執行個體的角色如何運作？

下圖中，開發人員在 Amazon EC2 執行個體上執行應用程式，該應用程式需要存取名為 photos 的 S3 儲存貯體。管理員建立 Get-pics 服務角色，並將角色連接到 Amazon EC2 執行個體。該角色包括許可政策，該政策授予對指定的 S3 儲存貯體的唯讀存取許可。角色還包含信任政策，該政策可允許 Amazon EC2 執行個體擔任角色並擷取臨時憑證。當應用程式在執行個體上執行時，它可以使用角色的臨時憑證來存取照片儲存貯體。管理員不必授予開發人員存取照片儲存貯體的許可，開發人員也不必共用或管理憑證。



1. 管理員使用 IAM 建立 **Get-pics** 角色。在角色的信任政策中，管理員指定只有 Amazon EC2 執行個體可以擔任該角色。在角色的許可政策中，管理員為 photos 儲存貯體指定唯讀許可。
2. 開發人員啟動 Amazon EC2 執行個體，並將 Get-pics 角色指派給該執行個體。

> **Note**
>
> 如果您使用 IAM 主控台，則會為您管理執行個體描述檔，並且對您來說幾乎是透明的。但是，如果您使用 AWS CLI 或 API 建立和管理角色和 Amazon EC2 執行個體，則必須建立執行個體設定檔，並以個別步驟將角色指派給該執行個體。然後，當您啟動執行個體時，必須指定執行個體設定檔名稱，而不是角色名稱。

3. 應用程式執行時，會從 Amazon EC2 執行個體中繼資料中取得暫時安全憑證，如從執行個體中繼資料中擷取安全憑證中所述。這些是代表角色的暫時安全憑證，並且在有限的時間段內有效。

透過一些 AWS 開發套件，開發人員可以使用透明地管理暫時安全憑證的提供者。個別 AWS SDK 的文件說明了該 SDK 支援用於管理認證的功能。)

或者，應用程式可以直接從 Amazon EC2 執行個體的執行個體中繼資料中取得臨時憑證。您可以從中繼資料的 `iam/security-credentials/role-name` 類別 (在本例中為 `iam/security-credentials/Get-pics`) 中取得憑證和相關的值。如果應用程式從執行個體中繼資料取得憑證，則可以快取憑證。

4. 使用擷取到的臨時憑證，應用程式存取照片儲存貯體。基於連接至 **Get-pics** 角色的政策，應用程式具有唯讀許可。

   在執行個體上提供的臨時安全憑證在到期之前會自動更新，以便有效設定為永久可用。應用程式只需要確保在目前中繼資料到期之前，它從執行個體中繼資料中取得一組新的憑證。您可以使用 AWS SDK 來管理認證，因此應用程式不需要包含其他邏輯即可重新整理認證。例如，使用執行個體描述檔憑證提供者建立用戶端。不過，如果應用程式從執行個體中繼資料取得臨時安全憑證並快取它們，則應每小時或在目前設定過期之前至少 15 分鐘取得重新整理的憑證集。過期時間包含在 `iam/security-credentials/role-name` 類別中所傳回的資訊。

使用 Amazon EC2 角色所需的許可

若要使用角色啟動執行個體，開發人員必須具有啟動 Amazon EC2 執行個體的許可，以及傳遞 IAM 角色的許可。

下列範例原則可讓使用者使 AWS Management Console 用啟動具有角色的執行個體。此政策包含萬用字元 (*)，以允許使用者傳遞任何角色，並執行列出的 Amazon EC2 動作。`ListInstanceProfiles` 動作讓使用者查看 AWS 帳戶中可用的所有角色。

Example 範例政策，授予使用者使用 Amazon EC2 主控台啟動具有任何角色的執行個體的許可

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "IamPassRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
```

```
                }
            }
        },
        {
            "Sid": "ListEc2AndListInstanceProfiles",
            "Effect": "Allow",
            "Action": [
                "iam:ListInstanceProfiles",
                "ec2:Describe*",
                "ec2:Search*",
                "ec2:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

限制哪些角色可以傳遞給 Amazon EC2 執行個體 (使用 PassRole)

您可以使用 `PassRole` 許可，限制使用者在啟動執行個體時可以將哪個角色傳遞給 Amazon EC2 執行個體。這有助於防止使用者執行具有比授予使用者更多許可的應用程式，也就是說，能夠獲得更高的許可。例如，假設使用者 Alice 只擁有啟動 Amazon EC2 執行個體以及使用 Amazon S3 儲存貯體的許可，但她傳遞給 Amazon EC2 執行個體的角色具有使用 IAM 和 Amazon DynamoDB 的許可。在這種情況下，Alice 或許可以啟動執行個體、登入執行個體、取得暫時安全憑證，然後執行她未授權的 IAM 或 DynamoDB 動作。

若要限制使用者可以將哪些角色傳遞給 Amazon EC2 執行個體，則要建立允許 `PassRole` 動作的政策。然後，您需要將政策連接到預計啟動 Amazon EC2 執行個體的使用者 (或使用者所屬的 IAM 群組)。在政策的 `Resource` 元素中，您需列出允許使用者傳遞給 Amazon EC2 執行個體的角色。當使用者啟動執行個體並將角色與其關聯時，Amazon EC2 檢查是否允許使用者傳送該角色。當然，您也應當確保使用者可以傳遞的角色不包含比使用者應具有的許可更多的許可。

> **Note**
>
> `PassRole` 不是與`RunInstances` 或 `ListInstanceProfiles` 相同的 API 動作。相反，它是一種權限，用於 AWS 檢查角色 ARN 作為參數傳遞給 API（或控制台代表用戶執行此操作）。它可協助管理員控制哪些使用者可以傳遞哪些角色。在這種情況下，它確保允許使用者將特定角色連接到 Amazon EC2 執行個體。

Example 此範例政策授予使用者許可，使其可以啟動具有特定角色的 Amazon EC2 執行個體

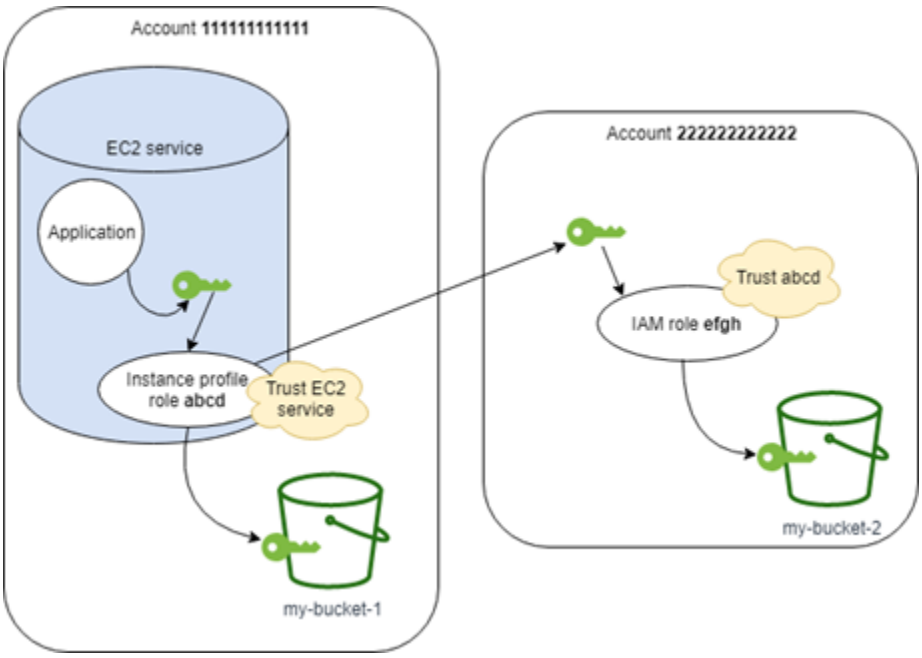
以下範例政策可允許使用者使用 Amazon EC2 API 來啟動具有角色的執行個體。Resource 元素指定角色的 Amazon Resource Name (ARN)。透過指定 ARN，政策授予使用者僅傳遞 Get-pics 角色的許可。如果使用者在啟動執行個體時嘗試指定不同的角色，則動作會失敗。使用者沒有執行任何執行個體的許可，無論其是否有傳遞角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::account-id:role/Get-pics"
    }
  ]
}
```

允許執行個體設定檔角色切換到另一個帳戶中的角色

您可以允許在 Amazon EC2 執行個體上執行的應用程式在另一個帳戶中執行命令。若要執行此操作，您必須允許將第一個帳戶中的 Amazon EC2 執行個體角色切換到第二個帳戶中的角色。

想像一下，您正在使用兩個， AWS 帳戶 並且希望允許在 Amazon EC2 實例上運行的應用程序在兩個帳戶中運行AWS CLI命令。假設 Amazon EC2 執行個體存在於帳戶 111111111111。該執行個體包含 abcd 執行個體描述檔角色，該角色會允許應用程式在相同 111111111111 帳戶中對 my-bucket-1 儲存貯體執行唯讀 Amazon S3 任務。不過，也必須允許應用程式擔任 efgh 跨帳戶角色來在帳戶 222222222222 中存取 my-bucket-2Amazon S3 儲存貯體。



abcd Amazon EC2 執行個體設定檔角色必須具有以下許可政策，以允許應用程式存取 my-bucket-1 Amazon S3 儲存貯體：

帳戶 111111111111 ***abcd*** 角色許可政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::my-bucket-1/*",
                "arn:aws:s3:::my-bucket-1"
            ]
        },
        {
            "Sid": "AllowIPToAssumeCrossAccountRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::222222222222:role/efgh"
        }
    ]
}
```

abcd 角色必須信任 Amazon EC2 服務擔任該角色。若要執行此操作，abcd 角色必須有以下信任政策：

帳戶 111111111111 ***abcd*** 角色信任政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "abcdTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"Service": "ec2.amazonaws.com"}
        }
    ]
}
```

假設 efgh 跨帳戶角色允許在相同 222222222222 帳戶中對 my-bucket-2 儲存貯體的唯讀 Amazon S3 任務。若要執行此操作，efgh 跨帳戶角色必須有以下許可政策：

帳戶 222222222222 ***efgh*** 角色許可政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket-2/*",
                "arn:aws:s3:::my-bucket-2"
            ]
        }
    ]
}
```

efgh 角色必須信任 abcd 執行個體設定檔角色擔任該角色。若要執行此操作，efgh 角色必須有以下信任政策：

帳戶 222222222222 ***efgh*** 角色信任政策

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "efghTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
        }
    ]
}
```

我該如何開始？

若要了解角色如何使用 Amazon EC2 執行個體，您需要使用 IAM 主控台建立角色，啟動使用該角色的 Amazon EC2 執行個體，然後檢查正在執行的執行個體。您可以檢查執行個體中繼資料以查看角色的

暫時憑證如何可用於執行個體。您還可以查看在執行個體上執行的應用程式如何使用該角色。使用以下資源以進一步了解。

-
- SDK 演練。 AWS SDK 文件包含逐步解說，顯示在 Amazon EC2 執行個體上執行的應用程式，該執行個體使用臨時登入資料作為角色讀取 Amazon S3 儲存貯體。以下每個演練都使用不同的程式設計語言提供類似的步驟：
  - 《AWS SDK for Java 開發人員指南》中的使用 SDK for Java 設定 Amazon EC2 的 IAM 角色
  - 《AWS SDK for .NET 開發人員指南》中的使用適用於 .NET 的開發套件啟動 Amazon EC2 執行個體
  - 《AWS SDK for Ruby 開發人員指南》中的使用 SDK for Ruby 建立 Amazon EC2 執行個體

相關資訊

如需建立角色或為 Amazon EC2 執行個體建立角色的詳細資訊，請參閱以下資訊：

- 如需將 IAM 角色與 Amazon EC2 執行個體搭配使用的詳細資訊，請參閱 Amazon EC2 使用者指南。
- 若要建立角色，請參閱 建立 IAM 角色
- 如需有關使用暫時安全憑證的詳細資訊，請參閱 IAM 中的暫時安全憑證。
- 如果使用 IAM API 或 CLI，則必須建立和管理 IAM 執行個體描述檔。如需有關執行個體設定檔的詳細資訊，請參閱 使用執行個體設定檔。
- 如需執行個體中繼資料中繼資料中角色的臨時安全登入資料的詳細資訊，請參閱 Amazon EC2 使用者指南中的從執行個體中繼資料擷取安全

使用執行個體設定檔

使用執行個體描述檔將 IAM 角色傳遞給 EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的適用於 Amazon EC2 的 IAM 角色。

管理執行個體設定檔 (主控台)

如果您使用 AWS Management Console 為 Amazon EC2 建立角色，主控台會自動建立執行個體設定檔，並為其指定與角色相同的名稱。然後，當您使用 Amazon EC2 主控台啟動具有 IAM 角色的執行個體時，可以選擇要與執行個體關聯的角色。在主控台中，顯示的清單實際上是執行個體設定檔名稱的清單。主控台不會為與 Amazon EC2 無關的角色建立執行個體描述檔。

如果角色和執行 AWS Management Console 個體設定檔具有相同的名稱，您可以使用刪除 Amazon EC2 的 IAM 角色和執行個體設定檔。若要進一步了解刪除執行個體描述檔，請參閱 刪除角色或執行個體設定檔。

管理執行個體設定檔 (AWS CLI 或 AWS API)

如果您透過 AWS CLI 或 AWS API 管理角色，您可以將角色和執行個體設定檔建立為單獨的動作。由於角色和執行個體描述檔可以有不同的名稱，因此您必須知道執行個體描述檔的名稱以及它們包含的角色的名稱。這樣，您可以在啟動 EC2 執行個體時選擇正確的執行個體描述檔。

您可以將標籤連接至 IAM 資源 (包括執行個體描述檔)，以識別、整理和控制其存取。您只能在使用 AWS CLI 或 AWS API 時標記執行個體設定檔。

> **Note**
>
> 執行個體描述檔只能包含一個 IAM 角色，但角色可以包含在多個執行個體描述檔中。每個執行個體設定檔的一個角色的限制無法增加。您可以移除現有角色，然後將不同角色新增到執行個體描述檔。然後， AWS 由於最終一致性，您必須等待更改出現在所有內容中。若要強制變更，必須取消關聯執行個體描述檔，然後關聯執行個體描述檔，或者可以停止執行個體，然後重新啟動它。

管理執行個體設定檔 (AWS CLI)

您可以使用下列 AWS CLI 指令來處理 AWS 帳戶中的執行個體設定檔。

- 建立執行個體描述檔: `aws iam create-instance-profile`
- 標記執行個體描述檔：`aws iam tag-instance-profile`
- 列出執行個體描述檔的標籤：`aws iam list-instance-profile-tags`
- 取消標記執行個體描述檔：`aws iam untag-instance-profile`
- 將角色新增至執行個體描述檔: `aws iam add-role-to-instance-profile`
- 列出執行個體描述檔: `aws iam list-instance-profiles`, `aws iam list-instance-profiles-for-role`
- 取得有關執行個體描述檔的資訊: `aws iam get-instance-profile`
- 從執行個體描述檔中移除角色: `aws iam remove-role-from-instance-profile`
- 刪除執行個體描述檔: `aws iam delete-instance-profile`

您還可以使用以下命令將角色連接到已在執行的 EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 的 IAM 角色。

- 將具有角色的執行個體描述檔連接至已停止或正在執行的 EC2 執行個體: aws ec2 associate-iam-instance-profile
- 取得有關連接至 EC2 執行個體的執行個體描述檔的資訊: aws ec2 describe-iam-instance-profile-associations
- 使用已停止或正在執行的 EC2 執行個體中的角色分開執行個體描述檔：aws ec2 disassociate-iam-instance-profile

管理執行個體設定檔 (AWS API)

您可以呼叫下列 AWS API 作業，以使用 AWS 帳戶.

- 建立執行個體描述檔: CreateInstanceProfile
- 標記執行個體描述檔：TagInstanceProfile
- 列出執行個體描述檔的標籤：ListInstanceProfileTags
- 取消標記執行個體描述檔：UntagInstanceProfile
- 將角色新增至執行個體描述檔: AddRoleToInstanceProfile
- 列出執行個體描述檔: ListInstanceProfiles, ListInstanceProfilesForRole
- 取得有關執行個體描述檔的資訊: GetInstanceProfile
- 從執行個體描述檔中移除角色: RemoveRoleFromInstanceProfile
- 刪除執行個體描述檔: DeleteInstanceProfile

您還可以呼叫以下操作將角色連接到已在執行的 EC2 執行個體。如需詳細資訊，請參閱 Amazon EC2 的 IAM 角色。

- 將具有角色的執行個體描述檔連接至已停止或正在執行的 EC2 執行個體: AssociateIamInstanceProfile
- 取得有關連接至 EC2 執行個體的執行個體描述檔的資訊: DescribeIamInstanceProfileAssociations
- 使用已停止或正在執行的 EC2 執行個體中的角色分開執行個體描述檔：DisassociateIamInstanceProfile

撤銷 IAM 角色暫時性安全憑證

> **Warning**
>
> 如果您遵循此頁面上的步驟，則會拒絕透過假設角色建立目前工作階段的所有使用者存取所有 AWS 動作和資源。這會導致使用者遺失未儲存的工作。

當您允許使用者存取 AWS Management Console 具有較長工作階段持續時間 (例如 12 小時) 的存取時，他們的臨時登入資料不會如此快速過期。如果使用者不慎將他們的憑證向未經授權的第三方公開，該方在工作階段的持續時間均有權存取。不過，您可以立即撤銷所有在某個時間點前授予該角色的憑證許可，如果需要的話。該角色在指定時間點前發出的所有暫時性憑證變成無效。這會強迫所有使用者重新驗證和請求新的憑證。

> **Note**
>
> 您無法撤銷服務連結角色的工作階段。

當您使用本主題中的程序撤銷角色的權限時，會將新的內嵌原則 AWS 附加至拒絕所有動作的所有權限的角色。如果使用者在您撤銷許可的時間點「之前」擔任該角色，則它只會包含套用限制條件的情況。如果使用者在撤銷許可「之後」擔任該角色，則拒絕政策不會套用至該使用者。

如需拒絕存取的詳細資訊，請參閱 停用臨時安全性憑證的許可。

> **Important**
>
> 此拒絕政策適用於指定角色的所有使用者，而不只是具主控台工作階段較長持續時間的角色。

從角色撤銷工作階段許可的最低許可

若要從角色成功地撤銷工作階段許可，您必須擁有該角色的 `PutRolePolicy` 許可。這可讓您將 `AWSRevokeOlderSessions` 內嵌政策連接至該角色。

撤銷工作階段許可

您可以撤銷角色的工作階段權限，以拒絕任何擔任該角色之使用者的所有權限。

> **Note**
>
> 您無法編輯從 IAM 身分中心權限集建立的 IAM 中的角色。您必須在 IAM 身分中撤銷使用者的作用中權限集工作階段。如需詳細資訊，請參閱 IAM 身分中心使用者指南中的撤銷由權限集建立的作用中 IAM 角色工作階段。

立即拒絕對任何目前使用者的角色憑證的所有許可

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後選擇您想要撤銷其許可的角色名稱 (不是核取方塊)。
3. 在所選取角色的 Summary (摘要) 頁面上，選擇 Revoke sessions (工作階段) 標籤。
4. 在 Revoke sessions (撤銷工作階段) 標籤上，選擇 Revoke active sessions (撤銷作用中的工作階段)。
5. AWS 會要求您確認動作。選取 I acknowledge that I am revoking all active sessions for this role. (我確認我將撤銷此角色的所有作用中工作階段。) 核取方塊，然後在對話方塊中選擇 Revoke active sessions (撤銷作用中工作階)。

   IAM 接著會將名為的政策附加`AWSRevokeOlderSessions`至該角色。選擇 [撤銷作用中階段作業] 之後，原則會拒絕對過去擔任該角色的使用者以及 future 大約 30 秒的所有存取權。此 future 時間選項會考量原則的傳輸延遲，以處理在指定區域中更新的政策生效之前取得或更新的新階段作業。在您選擇「撤銷作用中工作階段」後，擔任角色超過 30 秒的任何使用者都不會受到影響。若要了解變更不一定會立即顯示的原因，請參閱 我所做的變更不一定都會立刻生效。

> **Note**
>
> 如果您選擇稍後再撤銷作用中階段作業，則會重新整理原則中的日期和時間戳記，並再次拒絕任何在新指定時間之前擔任該角色的使用者的所有權限。

以這種方式呼叫工作階段的有效使用者必須獲得臨時憑證，新工作階段才能繼續運作。在憑證過期前，AWS CLI 會快取憑證。若要強制 CLI 刪除並重新整理已失效的快取憑證，請執行以下命令之一：

Linux、macOS 或 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

在指定時間之前撤銷工作階段許可

您也可以隨時使用 AWS CLI 或 SDK 撤銷工作階段權限，在原則的「條件」元素中指定AWS# TokenIssue##金鑰的值。

在 aws:TokenIssueTime 的值早於指定的日期和時間時，此政策會拒絕所有許可。aws:TokenIssueTime 的值對應於臨時安全性憑證的確切建立時間。此aws:TokenIssueTime值僅存在於使用臨時安全登入資料簽署的 AWS 請求內容中，因此政策中的 Deny 陳述式不會影響使用 IAM 使用者長期登入資料簽署的要求。

此政策也可連接至角色。在這種情況下，該政策只會影響由該角色在指定日期和時間之前建立的臨時安全性憑證。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {"aws:TokenIssueTime": "2014-05-07T23:47:00Z"}
    }
  }
}
```

以這種方式呼叫工作階段的有效使用者必須獲得臨時憑證，新工作階段才能繼續運作。會 AWS CLI 快取認證，直到認證過期為止。若要強制 CLI 刪除並重新整理已失效的快取憑證，請執行以下命令之一：

Linux、macOS 或 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

管理 IAM 角色

有時候您需要修改或刪除已建立的角色。若要變更角色，您可以執行以下任何項目：

- 修改與該角色關聯的政策
- 變更誰可以存取該角色
- 編輯該角色授予使用者的許可
- 針對使用 AWS CLI 或 API 假設的角色變更工作階段持續時間上 AWS Management Console限設定

您還可以刪除不再需要的角色。您可以從 AWS Management Console、和 API 管理您的角色。 AWS CLI

主題

- 修改角色
- 刪除角色或執行個體設定檔

修改角色

您可以使用 AWS Management Console AWS CLI、或 IAM API 對角色進行變更。

主題

- 檢視角色存取
- 根據存取資訊產生政策
- 修改角色 (主控台)
- 修改角色 (AWS CLI)
- 修改角色 (AWS API)

檢視角色存取

變更角色的許可之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

根據存取資訊產生政策

您有時可能會對 IAM 實體 (使用者或角色) 授予超出其要求的許可。為了協助您精簡所授予的許可，您可以根據實體的存取活動產生 IAM 政策。IAM Access Analyzer 會檢閱您的 AWS CloudTrail 記錄檔，並產生政策範本，其中包含實體在指定日期範圍內使用的許可。您可以使用範本建立具有精細許可的受管政策，然後將其連接至 IAM 實體。如此一來，您只會針對特定使用案例授與使用者或角色與 AWS 資源互動所需的權限。如需進一步了解，請參閱根據存取活動產生政策。

修改角色 (主控台)

您可以使用 AWS Management Console 來修改角色。若要變更角色的一組標籤，請參閱 管理 IAM 角色的標籤 (主控台)。

主題

- 修改角色信任政策 (主控台)
- 修改角色許可政策 (主控台)
- 修改角色描述 (主控台)
- 修改角色最大工作階段持續時間 (主控台)
- 修改角色許可界限 (主控台)

修改角色信任政策 (主控台)

若要變更誰可擔任該角色，您必須修改角色的信任政策。您無法修改服務連結角色的信任政策。

> **備註**
>
> - 如果使用者已於角色信任政策中列為主體，卻無法擔任該角色，請檢查該使用者的許可界限。如果已經設定使用者的許可界限，則其必須允許 `sts:AssumeRole` 動作。
> - 若要允許使用者在角色工作階段中再次擔任目前的角色，請在角色信任原則中指定角色 AWS 帳戶 ARN 或 ARN 作為主參與者。 AWS 服務 提供 Amazon EC2、Amazon ECS、亞馬遜 EKS 和 Lambda 等運算資源提供臨時登入資料並自動更新這些登入資料。此可確保您始終擁有一組有效的憑證。對於這些服務，不需要再次擔任目前的角色即可取得臨時憑證。但是，如果您想要傳遞工作階段標籤或一個工作階段政策，則需要再次擔任目前的角色。

修改角色信任政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
3. 在您帳戶的角色清單中，選擇您想要修改的角色名稱。
4. 選擇 Trust Relationships (信任關係) 標籤，然後選擇 Edit Trust Relationship (編輯信任政策)。
5. 視需要編輯信任政策。若要新增其他可擔任該角色的主體，請在 `Principal` 元素中指定它們。例如，下列原則程式碼片段顯示如何參考`Principal`元素 AWS 帳戶 中的兩個：

   ```
   "Principal": {
     "AWS": [
       "arn:aws:iam::111122223333:root",
       "arn:aws:iam::444455556666:root"
     ]
   },
   ```

   如果您指定在另一個帳戶中的主體，這時在角色的信任政策中新增帳戶，只能建立一半的跨帳戶信任關係。根據預設，信任帳戶中沒有任何使用者可擔任該角色。剛信任的帳戶管理員必須授予新的使用者擔任該角色的許可。為了這樣做，管理員必須建立或編輯連接於使用者的政策，以允許使用者存取 `sts:AssumeRole` 動作。如需詳細資訊，請參閱下列程序或 向使用者授予切換角色的許可。

   下列原則程式碼片段顯示如何參考`Principal`元素中的兩個 AWS 服務：

   ```
   "Principal": {
     "Service": [
       "opsworks.amazonaws.com",
       "ec2.amazonaws.com"
     ]
   },
   ```

6. 當您完成編輯您的信任政策，請選擇 Update policy (更新政策) 來儲存您的變更。

   如需有關政策結構和語法的詳細資訊，請參閱 IAM 中的政策和許可 和 IAM JSON 政策元素參考。

允許信任外部帳戶中的使用者使用該角色 (主控台)

如需有關此程序的更多資訊和詳細資訊，請參閱 向使用者授予切換角色的許可。

1. 登入受信任的外部 AWS 帳戶。
2. 決定是否將許可連接到使用者或群組。在 IAM 主控台的導覽窗格中，相應地選擇 Users (使用者) 或 User groups (使用者群組)。
3. 選擇要授予其存取許可的使用者或群組的名稱，然後選擇 Permissions (許可) 標籤。
4. 執行下列任一步驟：

   - 若要編輯客戶受管政策，請選擇政策名稱，並選擇 Edit policy (編輯政策)，然後選擇 JSON 標籤。您無法編輯受 AWS 管理的策略。 AWS 受管理的策略會以 AWS 圖示 () 顯示。如需 AWS 受管政策與客戶管理策略之間差異的詳細資訊，請參閱受管政策與內嵌政策。
   - 若要編輯內嵌政策，請選擇政策名稱旁的箭頭，然後選擇 Edit policy (編輯政策)。
5. 在政策編輯器中，新增一個指定以下內容的新 `Statement` 元素：

   ```
   {
     "Effect": "Allow",
     "Action": "sts:AssumeRole",
     "Resource": "arn:aws:iam::ACCOUNT-ID:role/ROLE-NAME"
   }
   ```

   將陳述式中的 ARN 取代為使用者可擔任的角色的 ARN。
6. 按照螢幕上的提示來完成政策編輯。

修改角色許可政策 (主控台)

若要變更角色允許的許可，請修改角色的許可政策 (或政策)。您無法修改 IAM 中服務連結角色的許可政策。您也許可以修改該服務 (取決於角色) 內的許可政策。若要檢查服務是否支援這項功能，請參閱AWS 與 IAM 搭配使用的服務，並且尋找 Service-linked roles (服務連結的角色) 資料列為 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

變更角色允許的許可 (主控台)

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
3. 選擇您要修改的角色的名稱，然後選擇 Permissions (許可) 標籤。
4. 執行下列任一步驟：

- 若要編輯現有客戶受管政策，請選擇政策名稱，然後選擇 Edit policy (編輯政策)。

> **Note**
>
> 您無法編輯受 AWS 管理的策略。 AWS 受管理的策略會以 AWS 圖示 () 顯示。如需有關 AWS 受管政策與客戶受管政策之間差異的詳細資訊，請參閱 受管政策與內嵌政策。

- 若要將現有受管政策連接至角色，請選擇 Add permissions (新增許可)，然後選擇 Attach policies (連接政策)。
- 若要編輯現有內嵌政策，請展開政策，然後選擇 Edit (編輯)。
- 若要嵌入新的內嵌政策，請選擇 Add permissions (新增許可)，然後選擇 Create inline policy (建立內嵌政策)。
- 若要移除角色中的現有策略，請選取策略名稱旁邊的核取方塊，然後選擇 [移除]。

修改角色描述 (主控台)

若要變更描述的描述，請修改描述文字。

變更角色的說明 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
3. 選擇要修改之角色的名稱。
4. 在 Summary (摘要) 區段中，選擇 Edit (編輯)。
5. 在方塊中輸入新說明，然後選擇 Save changes (儲存變更)。

修改角色最大工作階段持續時間 (主控台)

若要為使用主控台、或 AWS API 假設的角色指定工作階段持續時間上限設定 AWS CLI，請修改工作階段持續時間上限設定值。此設定的值可介於 1 小時至 12 小時。如果未指定值，則套用預設最大值 1 小時。此設定不會限制 AWS 服務擔任的工作階段。

若要變更使用主控台或 AWS API (主控台) 假設角色的工作階段持續時間上限設定 AWS CLI

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)。
3. 選擇要修改之角色的名稱。
4. 在 Summary (摘要) 區段中，選擇 Edit (編輯)。
5. 在 Maximum session duration (最大工作階段持續時間) 中，選擇一個值。或者，選擇 Custom duration (自訂持續時間) 並輸入一個值 (以秒為單位)。
6. 選擇儲存變更。

   在下次有人擔任此角色之前，您的變更不會生效。若要了解如何撤銷此角色的現有工作階段，請參閱撤銷 IAM 角色暫時性安全憑證。

在中 AWS Management Console，IAM 使用者工作階段預設為 12 小時。在主控台中切換角色的 IAM 使用者會被授予角色最長工作階段持續時間或使用者工作階段中的剩餘時間，以較短者為準。

任何擔任 AWS CLI 或 AWS API 角色的人都可以要求更長的工作階段，最多可達此上限。`MaxSessionDuration` 設定決定可以請求之角色工作階段的最大持續時間。

- 若要使 AWS CLI 用`duration-seconds`參數指定工作階段持續時間。如需進一步了解，請參閱切換到 IAM 角色 (AWS CLI)。
- 若要使用 AWS API 指定工作階段持續時間，請使用`DurationSeconds`參數。如需進一步了解，請參閱 切換到身分與存取權管理角色 (AWS API)。

修改角色許可界限 (主控台)

若要變更角色適用的許可上限，請修改角色的許可界限。

變更用於設定角色許可界限的政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色。
3. 選擇您想要為其變更許可界限的角色名稱。
4. 選擇 Permissions (許可) 標籤。如有必要，開啟 許可界限 區段，然後選擇 變更界限。

5. 選擇要用於許可界限的政策。
6. 選擇 Change boundary (變更界限)。

   在下次有人擔任此角色之前，您的變更不會生效。

修改角色 (AWS CLI)

您可以使用 AWS Command Line Interface 來修改角色。若要變更角色的一組標籤，請參閱 管理 IAM 角色 (AWS CLI 或 AWS API) 上的標籤。

主題

- 修改角色信任政策 (AWS CLI)
- 修改角色許可政策 (AWS CLI)
- 修改角色描述 (AWS CLI)
- 修改角色最大工作階段持續時間 (AWS CLI)
- 修改角色許可界限 (AWS CLI)

修改角色信任政策 (AWS CLI)

若要變更誰可擔任該角色，您必須修改角色的信任政策。您無法修改服務連結角色的信任政策。

> **備註**
>
> - 如果使用者已於角色信任政策中列為主體，卻無法擔任該角色，請檢查該使用者的許可界限。如果已經設定使用者的許可界限，則其必須允許 `sts:AssumeRole` 動作。
> - 若要允許使用者在角色工作階段中再次擔任目前的角色，請在角色信任原則中指定角色 AWS 帳戶 ARN 或 ARN 作為主參與者。 AWS 服務 提供 Amazon EC2、Amazon ECS、亞馬遜 EKS 和 Lambda 等運算資源提供臨時登入資料並自動更新這些登入資料。此可確保您始終擁有一組有效的憑證。對於這些服務，不需要再次擔任目前的角色即可取得臨時憑證。但是，如果您想要傳遞工作階段標籤或一個工作階段政策，則需要再次擔任目前的角色。若要瞭解如何修改角色信任原則以新增主參與者角色 ARN 或 AWS 帳戶 ARN，請參閱。修改角色信任政策 (主控台)

修改角色信任政策 (AWS CLI)

1. (選用) 如果您不知道要修改的角色的名稱，請執行以下命令列出您帳戶中的角色：

   - aws iam list-roles

2. (選用) 若要檢視角色的目前信任政策，請執行下列命令：

   - aws iam get-role

3. 若要修改可以存取角色的信任主體，請使用更新的信任政策建立文字檔案。您可以使用任何文字編輯器來建構政策。

   例如，下列信任原則顯示如何參考`Principal`元素 AWS 帳戶 中的兩個。這可讓使用者在兩個不同的 AWS 帳戶 內擔任此角色。

   ```
   {
       "Version": "2012-10-17",
       "Statement": {
           "Effect": "Allow",
           "Principal": {"AWS": [
               "arn:aws:iam::111122223333:root",
               "arn:aws:iam::444455556666:root"
           ]},
           "Action": "sts:AssumeRole"
       }
   }
   ```

   如果您指定在另一個帳戶中的主體，這時在角色的信任政策中新增帳戶，只能建立一半的跨帳戶信任關係。根據預設，信任帳戶中沒有任何使用者可擔任該角色。剛信任的帳戶管理員必須授予新的使用者擔任該角色的許可。為了這樣做，管理員必須建立或編輯連接於使用者的政策，以允許使用者存取 `sts:AssumeRole` 動作。如需詳細資訊，請參閱下列程序或 向使用者授予切換角色的許可。

4. 若要使用剛建立的檔案更新信任政策，請執行下列命令：

   - AWS IAM update-assume-role-policy

若要允許信任外部帳戶中的使用者使用該角色 (AWS CLI)

如需有關此程序的更多資訊和詳細資訊，請參閱 向使用者授予切換角色的許可。

1. 建立 JSON 檔案，其中包含擔任該角色所需的許可的許可政策。例如，以下政策包含最低必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT-ID-THAT-CONTAINS-ROLE:role/ROLE-NAME"
  }
}
```

   將陳述式中的 ARN 取代為使用者可擔任的角色的 ARN。

2. 執行以下命令以上傳 JSON 檔案 (其中包含的信任政策) 至 IAM：

   - aws iam create-policy

   此命令的輸出結果含有該政策的 ARN。記下此 ARN，因為您需要在稍後的步驟中使用。

3. 決定將政策到連接哪位使用者或群組。如果您不知道適用的使用者或群組的名稱，請使用下列其中一個命令來列出帳戶中的使用者或群組：

   - aws iam list-users
   - aws iam list-groups

4. 使用下列命令之一將您在之前的步驟中建立的政策連接到使用者或群組：

   - AWS IAM attach-user-policy
   - AWS IAM attach-group-policy

修改角色許可政策 (AWS CLI)

若要變更角色允許的許可，請修改角色的許可政策 (或政策)。您無法修改 IAM 中服務連結角色的許可政策。您也許可以修改該服務 (取決於角色) 內的許可政策。若要檢查服務是否支援這項功能，請參閱AWS 與 IAM 搭配使用的服務，並且尋找 Service-linked roles (服務連結的角色) 資料列為 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

若要變更角色允許的許可 (AWS CLI)

1. (選用) 若要檢視與角色關聯的目前許可，請執行下列命令：

1. AWS IAM 列 list-role-policies出內聯政策
2. AWS IAM 列 list-attached-role-policies出受管政策

2. 更新角色許可的命令會有所不同，具體取決於您是要更新受管政策還是內嵌政策。

   若要更新受管政策，請執行下列命令來建立新版本的受管政策：

   - AWS IAM create-policy-version

   若要更新內嵌政策，請執行下列命令：

   - AWS IAM put-role-policy

修改角色描述 (AWS CLI)

若要變更描述的描述，請修改描述文字。

變更角色的說明 (AWS CLI)

1. (選用) 若要檢視角色的目前說明，請執行下列命令：

   - aws iam get-role

2. 若要更新角色的說明，請使用說明參數執行下列命令：

   - aws iam update-role

修改角色最大工作階段持續時間 (AWS CLI)

若要為使用 AWS CLI 或 API 擔任角色指定最大工作階段持續時間設定，請修改最大工作階段持續時間設定的值。此設定的值可介於 1 小時至 12 小時。如果未指定值，則套用預設最大值 1 小時。此設定不會限制 AWS 服務假設的工作階段。

> **Note**
>
> 任何擔任 AWS CLI 或 API 角色的人都可以使用 `duration-seconds` CLI 參數或 `DurationSeconds` API 參數來要求較長的工作階段。`MaxSessionDuration` 設定決定可以使用`DurationSeconds` 參數請求的角色工作階段的最大持續時間。如果使用者不為 `DurationSeconds` 參數指定一個值，則使用者的安全憑證有效期為 1 小時。

變更使用 AWS CLI (AWS CLI) 擔任之角色的最大工作階段持續時間設定

1. (選用) 若要檢視角色的目前最大工作階段持續時間設定，請執行下列命令：

   - aws iam get-role
2. 若要更新角色的工作階段持續時間設定，請使用 `max-session-duration` CLI 參數或 `MaxSessionDuration` API 參數執行以下命令：

   - aws iam update-role

   在下次有人擔任此角色之前，您的變更不會生效。若要了解如何撤銷此角色的現有工作階段，請參閱撤銷 IAM 角色暫時性安全憑證。

修改角色許可界限 (AWS CLI)

若要變更角色適用的許可上限，請修改角色的許可界限。

變更用於設定角色許可界限的受管政策 (AWS CLI)

1. (選用) 若要檢視角色的目前許可界限，請執行下列命令：

   - aws iam get-role
2. 若要使用不同的受管政策來更新角色的許可界限，請執行下列命令：

   - AWS IAM put-role-permissions-boundary

   角色只能有一個受管政策設定為許可界限。您若變更許可界限，您會變更角色適用的許可上限。

修改角色 (AWS API)

您可以使用 AWS API 來修改角色。若要變更角色的一組標籤，請參閱 管理 IAM 角色 (AWS CLI 或 AWS API) 上的標籤。

主題

- 修改角色信任原則 (AWS API)
- 修改角色權限原則 (AWS API)
- 修改角色描述 (AWS API)
- 修改角色工作階段持續時間上限 (AWS API)

- 修改角色權限界限 (AWS API)

修改角色信任原則 (AWS API)

若要變更誰可擔任該角色，您必須修改角色的信任政策。您無法修改服務連結角色的信任政策。

> **備註**
>
> - 如果使用者已於角色信任政策中列為主體，卻無法擔任該角色，請檢查該使用者的許可界限。如果已經設定使用者的許可界限，則其必須允許 `sts:AssumeRole` 動作。
> - 若要允許使用者在角色工作階段中再次擔任目前的角色，請在角色信任原則中指定角色 AWS 帳戶 ARN 或 ARN 作為主參與者。 AWS 服務 提供 Amazon EC2、Amazon ECS、亞馬遜 EKS 和 Lambda 等運算資源提供臨時登入資料並自動更新這些登入資料。此可確保您始終擁有一組有效的憑證。對於這些服務，不需要再次擔任目前的角色即可取得臨時憑證。但是，如果您想要傳遞工作階段標籤或一個工作階段政策，則需要再次擔任目前的角色。若要瞭解如何修改角色信任原則以新增主參與者角色 ARN 或 AWS 帳戶 ARN，請參閱。修改角色信任政策 (主控台)

若要修改角色信任原則 (AWS API)

1. (選用) 如果您不知道要修改的角色的名稱，請呼叫以下操作列出您帳戶中的角色：
   - ListRoles
2. (選用) 若要檢視角色的目前信任政策，請呼叫下列操作：
   - GetRole
3. 若要修改可以存取角色的信任主體，請使用更新的信任政策建立文字檔案。您可以使用任何文字編輯器來建構政策。

   例如，下列信任原則顯示如何參考`Principal`元素 AWS 帳戶 中的兩個。這可讓使用者在兩個不同的 AWS 帳戶 內擔任此角色。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"AWS": [
```

```
            "arn:aws:iam::111122223333:root",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "sts:AssumeRole"
    }
}
```

如果您指定在另一個帳戶中的主體，這時在角色的信任政策中新增帳戶，只能建立一半的跨帳戶信任關係。根據預設，信任帳戶中沒有任何使用者可擔任該角色。剛信任的帳戶管理員必須授予新的使用者擔任該角色的許可。為了這樣做，管理員必須建立或編輯連接於使用者的政策，以允許使用者存取 `sts:AssumeRole` 動作。如需詳細資訊，請參閱下列程序或 向使用者授予切換角色的許可。

4. 若要使用剛建立的檔案更新信任政策，請呼叫下列操作：

   - UpdateAssumeRolePolicy

允許受信任外部帳戶中的使用者使用角色 (AWS API)

如需有關此程序的更多資訊和詳細資訊，請參閱 向使用者授予切換角色的許可。

1. 建立 JSON 檔案，其中包含擔任該角色所需的許可的許可政策。例如，以下政策包含最低必要許可：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT-ID-THAT-CONTAINS-ROLE:role/ROLE-NAME"
  }
}
```

   將陳述式中的 ARN 取代為使用者可擔任的角色的 ARN。

2. 呼叫以下操作，以上傳 JSON 檔案 (其中包含的信任政策) 至 IAM：

   - CreatePolicy

   此操作的輸出結果含有該政策的 ARN。記下此 ARN，因為您需要在稍後的步驟中使用。

3. 決定將政策到連接哪位使用者或群組。如果您不知道適用的使用者或群組的名稱，請呼叫下列其中一個操作來列出帳戶中的使用者或群組：

   - ListUsers
   - ListGroups

4. 呼叫下列操作之一將您在之前的步驟中建立的政策連接到使用者或群組：

   - API：AttachUserPolicy
   - AttachGroupPolicy

修改角色權限原則 (AWS API)

若要變更角色允許的許可，請修改角色的許可政策 (或政策)。您無法修改 IAM 中服務連結角色的許可政策。您也許可以修改該服務 (取決於角色) 內的許可政策。若要檢查服務是否支援這項功能，請參閱AWS 與 IAM 搭配使用的服務，並且尋找 Service-linked roles (服務連結的角色) 資料列為 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

若要變更角色 (AWS API) 允許的權限

1. (選用) 若要檢視與角色關聯的目前許可，請呼叫下列操作：

   1. ListRolePolicies以列出內嵌政策
   2. ListAttachedRolePolicies列出受管理的策略

2. 更新角色許可的操作會有所不同，具體取決於您是要更新受管政策還是內嵌政策。

   若要更新受管政策，請呼叫下列操作來建立新版本的受管政策：

   - CreatePolicyVersion

   若要更新內嵌政策，請呼叫下列操作：

   - PutRolePolicy

修改角色描述 (AWS API)

若要變更描述的描述，請修改描述文字。

若要變更角色的描述 (AWS API)

1. (選用) 若要檢視角色的目前說明，請呼叫下列操作：
   - GetRole
2. 若要更新角色的說明，請使用說明參數呼叫下列操作：
   - UpdateRole

修改角色工作階段持續時間上限 (AWS API)

若要為使用 AWS CLI 或 API 擔任角色指定最大工作階段持續時間設定，請修改最大工作階段持續時間設定的值。此設定的值可介於 1 小時至 12 小時。如果未指定值，則套用預設最大值 1 小時。此設定不會限制 AWS 服務假設的工作階段。

> **Note**
>
> 任何擔任 AWS CLI 或 API 角色的人都可以使用 `duration-seconds` CLI 參數或 `DurationSeconds` API 參數來要求較長的工作階段。`MaxSessionDuration` 設定決定可以使用`DurationSeconds` 參數請求的角色工作階段的最大持續時間。如果使用者不為 `DurationSeconds` 參數指定一個值，則使用者的安全憑證有效期為 1 小時。

若要變更使用 API (AWS API) 假設角色的工作階段持續時間上限設定

1. (選用) 若要檢視角色的目前最大工作階段持續時間設定，請呼叫下列操作：
   - GetRole
2. 若要更新角色的工作階段持續時間設定，請使用 `max-sessionduration` CLI 參數或 `MaxSessionDuration` API 參數呼叫以下操作：
   - UpdateRole

   在下次有人擔任此角色之前，您的變更不會生效。若要了解如何撤銷此角色的現有工作階段，請參閱撤銷 IAM 角色暫時性安全憑證。

修改角色權限界限 (AWS API)

若要變更角色適用的許可上限，請修改角色的許可界限。

變更用於設定角色許可界限的受管政策 (AWS API)

1. (選用) 若要檢視角色的目前許可界限，請呼叫下列操作：
   - GetRole
2. 若要使用不同的受管政策來更新角色的許可界限，請呼叫下列操作：
   - PutRolePermissionsBoundary

   角色只能有一個受管政策設定為許可界限。您若變更許可界限，您會變更角色適用的許可上限。

刪除角色或執行個體設定檔

如果您不再需要角色，建議您刪除角色及其關聯的許可。如此一來，您就沒有未主動監控或維護的未使用實體。

如果角色與 EC2 執行個體相關聯，您也可以從執行個體設定檔中移除角色，然後刪除執行個體設定檔。

> **Warning**
>
> 確保您沒有任何 Amazon EC2 執行個體與您即將刪除的角色或執行個體描述檔一起執行。若刪除與執行中的執行個體相關聯的角色或執行個體描述檔，將會中斷執行個體上執行的所有應用程式。

如果您不想永久刪除某個角色，則可以停用該角色。若要執行此操作，請變更角色的政策，然後撤銷所有目前的工作階段。例如，您可以將政策新增至拒絕存取所有資訊的角色 AWS。您也可以編輯信任政策，以拒絕任何嘗試擔任該角色的人存取。如需有關撤銷工作階段的詳細資訊，請參閱 撤銷 IAM 角色暫時性安全憑證。

主題

- 檢視角色存取
- 刪除服務連結角色
- 刪除 IAM 角色 (主控台)
- 刪除 IAM 角色 (AWS CLI)
- 刪除 IAM 角色 (AWS API)

- 相關資訊

檢視角色存取

刪除角色之前，建議您先檢閱上次使用角色的時間。您可以使用 AWS Management Console、或 AWS API 來 AWS CLI執行此操作。您應該檢視此資訊，因為您不想移除正在使用該角色之某個使用者的存取權。

角色上次活動的日期可能與 Access Advisor (存取 Advisor) 索引標籤報告的上次使用日期不符。Access Advisor (存取 Advisor) 索引標籤只報告角色許可政策允許服務的活動。角色上次活動的日期包括存取中任何服務的最後一次嘗試 AWS。

> **Note**
>
> 角色上次活動和 Access Advisor 資料的追蹤期間為過去 400 天。如果您的區域在過去一年內已開始支援這些功能，此期間可能會縮短。該角色的上次使用時間可能已經超過 400 天。如需有關追蹤期間的詳細資訊，請參閱 AWS 追蹤上次存取資訊的位置。

檢視上次使用角色的時間 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)。
3. 尋找具有您要檢視活動之角色的資料列。您可以使用搜尋欄位縮減結果。檢視 Last activity (上次活動) 欄，查看自上次使用該角色後的天數。如果在追蹤期間內未曾使用該角色，資料表會顯示 None (無)。
4. 選擇角色名稱以查看詳細資訊。角色的 Summary (摘要) 頁面也包含 Last activity (上次活動)，會顯示角色上次使用的日期。如果過去 400 天內未使用過該角色，則 Last activity (上次活動) 會顯示 Not accessed in the tracking period (在追蹤期間內未存取)。

檢視上次使用角色的時間 (AWS CLI)

`aws iam get-role` - 執行此命令以傳回角色的資訊，包括 `RoleLastUsed` 物件。此物件包含 `LastUsedDate` 和上次使用角色的 `Region`。如有 `RoleLastUsed` 但不包含值，即表示未在追蹤期間內使用過該角色。

若要檢視上次使用角色的時間 (AWS API)

GetRole - 呼叫此操作以傳回角色的資訊，包括 RoleLastUsed 物件。此物件包含 LastUsedDate 和上次使用角色的 Region。如有 RoleLastUsed 但不包含值，即表示未在追蹤期間內使用過該角色。

刪除服務連結角色

如果角色是服務連結的角色，請檢閱連結服務的文件以了解如何刪除該角色。您可以前往主控台的 IAM Roles (角色) 頁面，檢視您帳戶中的服務連結角色。服務連結角色會在表格的 Trusted entities (受信任實體) 欄中以 (Service-linked role) ((服務連結角色)) 顯示。在 Summary (摘要) 頁面上的橫幅，也會指出角色是一個服務連結角色。

如果服務未包含刪除服務連結角色的文件，您可以使用 IAM 主控 AWS CLI台或 API 刪除角色。如需詳細資訊，請參閱 刪除服務連結角色。

刪除 IAM 角色 (主控台)

當您使用刪除角色時，IAM 會自動卸離與該角色相關聯的受管政策。 AWS Management Console 它還會自動刪除與該角色關聯的內嵌政策，以及包含該角色的任何 Amazon EC2 執行個體設定檔。

> **Important**
>
> 在某些情況下，角色可能會與 Amazon EC2 執行個體描述檔相關聯，而且角色和執行個體描述檔的名稱可能完全相同。在這種情況下，您可以使 AWS Management Console 用刪除角色和執行個體設定檔。此連結就會自動為您在主控台建立的角色和執行個體設定檔。如果您是從 Windows 工具或 AWS API 建立角色 PowerShell，則角色和執行個體設定檔可能具有不同的名稱。 AWS CLI在該情況下，您無法使用主控台將它們刪除。相反地，您必須使 AWS CLI用 Windows 工具或 AWS API PowerShell，先從執行個體設定檔中移除角色。然後，您必須採取不同步驟刪除該角色。

刪除角色 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)，然後勾選您要刪除之角色名稱旁的核取方塊。
3. 在頁面頂端，選擇 Delete (刪除)。
4. 在確認對話方塊中，複查上次存取的資訊，其中顯示每個選取角色上次存取 AWS 服務的時間。這可協助您確認角色目前是否在作用中。如果您要繼續，請在文字輸入欄位中輸入角色的名稱，並選擇 Delete (刪除)。如果您確定，可以繼續刪除，即使上次存取的資訊仍持續載入。

> **Note**
>
> 除非執行個體設定檔和角色同名，否則您不能使用主控台刪除執行個體設定檔。作為刪除角色的過程的一部分，執行個體描述檔會被刪除，如前面的程序中所述。若要刪除執行個體設定檔而不同時刪除角色，您必須使用 AWS CLI 或 AWS API。如需詳細資訊，請參閱下列區段。

刪除 IAM 角色 (AWS CLI)

使用刪除角色時，必須先刪除與該角色相關聯的內嵌政策。 AWS CLI 您還必須分離與該角色關聯的受管政策。如果您要刪除包含角色的相關聯執行個體設定檔，您必須分開刪除。

刪除角色 (AWS CLI)

1. 如果您不知道想要刪除的角色名稱，請輸入以下命令列出帳戶中的角色：

   ```
   aws iam list-roles
   ```

   此清單包含每個角色的 Amazon Resource Name (ARN)。透過 CLI 命令，使用角色名稱 (而非 ARN) 來參照角色。例如，如果角色具有下列 ARN：`arn:aws:iam::123456789012:role/myrole`，請將角色參照為 **`myrole`**。
2. 從角色關聯的所有執行個體設定檔移除該角色。

   a. 若要列出所有與角色相關聯的執行個體設定檔，請輸入下列命令：

      ```
      aws iam list-instance-profiles-for-role --role-name role-name
      ```

   b. 若要從執行個體設定檔中移除該角色，請針對每個執行個體設定檔輸入下列命令：

      ```
      aws iam remove-role-from-instance-profile --instance-profile-name instance-profile-name --role-name role-name
      ```
3. 刪除所有與該角色相關聯的政策。

   a. 若要列出角色中所有的內嵌政策，請輸入下列命令：

      ```
      aws iam list-role-policies --role-name role-name
      ```

   b. 若要刪除角色中的每個內嵌政策，請針對每個政策輸入下列命令：

```
aws iam delete-role-policy --role-name role-name --policy-name policy-name
```

c. 若要列出連接至角色的所有受管政策，請輸入下列命令：

```
aws iam list-attached-role-policies --role-name role-name
```

d. 若要從角色分離每個受管政策，請針對每個政策輸入下列命令：

```
aws iam detach-role-policy --role-name role-name --policy-arn policy-arn
```

4. 輸入下列命令以刪除角色：

```
aws iam delete-role --role-name role-name
```

5. 如果您不打算重複使用曾與角色相關聯的執行個體設定檔，您可以輸入下列命令予以刪除：

```
aws iam delete-instance-profile --instance-profile-name instance-profile-name
```

刪除 IAM 角色 (AWS API)

當您使用 IAM API 刪除該角色，您必須先刪除與該角色關聯的內嵌政策。您還必須分離與該角色關聯的受管政策。如果您要刪除包含角色的相關聯執行個體設定檔，您必須分開刪除。

若要刪除角色 (AWS API)

1. 若要列出與角色相關聯的所有執行個體設定檔，請呼叫ListInstanceProfilesForRole。

   要從實例配置文件中刪除角色，請調用RemoveRoleFromInstanceProfile。您必須傳遞角色名稱和執行個體設定檔名稱。

   如果您不打算重複使用與該角色相關聯的實例配置文件，請調DeleteInstanceProfile用刪除它。

2. 若要列出角色的所有內嵌政策，請呼叫ListRolePolicies。

   若要刪除與角色相關聯的內嵌政策，請呼叫DeleteRolePolicy。您必須傳遞角色名稱和內嵌政策名稱。

3. 若要列出所有附加至角色的受管理原則，請呼叫ListAttachedRolePolicies。

   若要中斷連接至該角色的受管理原則，請呼叫DetachRolePolicy。您必須傳遞角色名稱和受管政策 ARN。

4. 呼叫DeleteRole刪除角色。

相關資訊

如需有關執行個體設定檔的一般資訊，請參閱使用執行個體設定檔。

如需服務連結角色的一般資訊，請參閱 使用服務連結角色。

身分提供者與聯合

如果您已在以外管理使用者身分識別 AWS，則可以使用身分識別提供者，而不是在 AWS 帳戶. 透過身分識別提供者 (IdP)，您可以管理以外的使用者身分識別， AWS 並授與這些外部使用者身分識別權限，以便使用您帳戶中的 AWS 資源。如果您的組織已有自己的身分系統，例如公司使用者目錄，這個方式便很管用。如果您正在建立需要存取 AWS 資源的行動應用程式或 Web 應用程式，這也很管理

外部 IdP 為 AWS 使用 OpenID Connect（OIDC）或 SAML 2.0（安全斷言標記語言 2.0）提供身份信息。OIDC 會將未執行的應用程式 (例如 GitHub 「動作」) 連接 AWS 至 AWS 資源。眾所周知的 SAML 身分識別提供者的範例為 Shibboleth 和作用中目錄聯合服務。

> **Note**
>
> 作為安全最佳實務，建議您使用外部 SAML 身分提供者 (而不是在 IAM 中使用 SAML 聯合)，在 IAM Identity Center 中管理人類使用者。若要了解需要 IAM 使用者的特定情形，請參閱建立 IAM 使用者 (而非角色) 的時機。

當您使用 身分提供者時，您不需要建立自訂登入代碼或管理自己的使用者身分，IdP 會為您處理這些工作。IdP 會為您提供這些功能。您的外部使用者透過 IdP 登入，您可以授與這些外部身分識別權限，以便使用您帳戶中的 AWS 資源。身分識別提供者可協助保 AWS 帳戶 護您的安全，因為您不需要在應用程式中散佈或內嵌長期安全性登入資料，例如存取金鑰。

本指南涵蓋 IAM 聯合。IAM Identity Center 或 Amazon Cognito 可能會為您的使用案例提供更完善的支援。下列摘要與表格提供使用者可用來取得資源聯合存取權的方法概觀。 AWS

	Account type (帳戶類型)	存取管理...	支援的身分來源
使用 IAM Identity Center 的聯合	由多個帳戶管理 AWS Organizations	員工的人類使用者	• SAML 2.0

	Account type (帳戶類型)	存取管理...	支援的身分來源
			• 受管的 Active Directory • Identity Center 目錄
使用 IAM 的聯合	單一獨立帳戶	• 短期、小規模部署中的人類使用者 • 電腦使用者	• SAML 2.0 • OIDC
使用 Amazon Cognito 身分池的聯合	任何	需要 IAM 授權才能存取資源的應用程式使用者	• SAML 2.0 • OIDC • 選取 OAuth 2.0 社交身分提供者

使用 IAM Identity Center 的聯合

如需人類使用者的集中式存取管理，我們建議您使用 IAM Identity Center 管理您帳戶的存取權和這些帳戶內的許可。IAM 身分中心的使用者會獲得資源的短期登入 AWS 資料。您可以使用 Active Directory、外部身分識別提供者 (IdP) 或 IAM 身分中心目錄做為使用者和群組指派 AWS 資源存取權的身分識別來源。

IAM 身分中心支援與 SAML (安全宣告標記語言) 2.0 聯合身分識別，為獲授權可在存取入口網站中使用應用程式的使用者提供聯合單一登入存取權。 AWS 然後，使用者可以單一登入支援 SAML 的服務，包括 AWS Management Console 和第三方應用程式，例如 Microsoft 365、SAP 連接器和 Salesforce。

使用 IAM 的聯合

雖然我們強烈建議在 IAM Identity Center 中管理人類使用者，但您可以在短期、小規模部署中為人類使用者啟用使用 IAM 的聯合身分使用者存取權。IAM 可讓您使用個別的 SAML 2.0 和開放 ID Connect (OIDC)， IdPs 並使用聯合身分使用者屬性進行存取控制。使用 IAM，您可以將使用者屬性 (例如成本中心、標題或地區設定) 從您傳遞 IdPs 到 AWS，並根據這些屬性實作精細的存取權限。

工作負載是可提供商業價值的資源和程式碼的集合，例如應用程式或後端程序。您的工作負載可能需要 IAM 身分，才能對 AWS 服務、應用程式、操作工具和元件提出請求。這些身分包括在您的 AWS 環境中執行的機器，例如 Amazon EC2 執行個體或 AWS Lambda 函數。

您可以管理需要存取權的外部當事人的機器身分。若要將存取權授予機器身分，您可以使用 IAM 角色。IAM 角色具有特定許可，並透 AWS 過仰賴具有角色工作階段的臨時安全登入資料，提供存取方式。此外，您可能有其他電腦需 AWS 要存取您的 AWS 環境。對於在您以外執行的機器， AWS 可以在任何地方使用 IAM 角色。如需角色的詳細資訊，請參閱IAM 角色。如需如何使用角色委派存取權的詳細資訊 AWS 帳戶，請參閱IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶。

若要將 IdP 直接連結至 IAM，您可以建立身分識別提供者實體，以在您 AWS 帳戶 與 IdP 之間建立信任關係。與 OpenID Connect（OIDC）或 SAML 2.0（安全斷言標記語言 2.0）兼容的 IAM 支持 IdPs 。如需搭配使用其中一個 IdPs 搭配使用的詳細資訊 AWS，請參閱下列各節：

- OIDC 聯盟
- SAML 2.0 聯合身分

使用 Amazon Cognito 身分池的聯合

Amazon Cognito 專為想要在行動應用程式和 Web 應用程式中為使用者進行驗證和授權的開發人員而設計。Amazon Cognito 使用者集區會在應用程式中新增登入和註冊功能，而身分池則提供 IAM 憑證，讓使用者可以存取您在 AWS中管理的受保護資源。身分池會透過 `AssumeRoleWithWebIdentity` API 操作取得臨時工作階段的憑證。

Amazon Cognito 可與支援 SAML 和 OpenID Connect 的外部身分提供者以及社交身分提供者 (如 Facebook、Google 和 Amazon) 搭配運作。應用程式可以使用使用者集區或外部 IdP 登入使用者，然後透過 IAM 角色中的自訂臨時工作階段代表使用者擷取資源。

常用案例

> **Note**
>
> 我們建議您要求您的人類使用者在存取時使用臨時登入資料 AWS。你有沒有考慮過使用 AWS IAM Identity Center？ 您可以使用 IAM 身分中心集中管理多個存取權限， AWS 帳戶 並從單一位置為使用者提供所有指派帳戶的 MFA 保護的單一登入存取權。使用 IAM Identity Center，您可以在 IAM Identity Center 中建立和管理使用者身分，或輕鬆連線至您現有的 SAML 2.0 相容身分提供者。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

您可以使用外部身分識別提供者 (IdP) 來管理. 以及外部 IdP 以外的使用者身分識別 AWS。外部 IdP 可以為 AWS 使用 OpenID Connect（OIDC）或安全聲明標記語言（SAML）提供身份信息。當未執行的應用程式 AWS 需要存取資源時，通常會使用 OIDC。 AWS

當您想要設定與外部 IdP 的聯合時，您可以建立 IAM 身分提供者，以通 AWS 知外部 IdP 及其組態。這會在您 AWS 帳戶 與外部 IdP 之間建立信任。下列主題提供使用 IAM 身分提供者的常見案例。

主題

- 針對行動應用程式使用 Amazon Cognito
- 針對行動應用程式使用 OIDC 聯合 API 作業

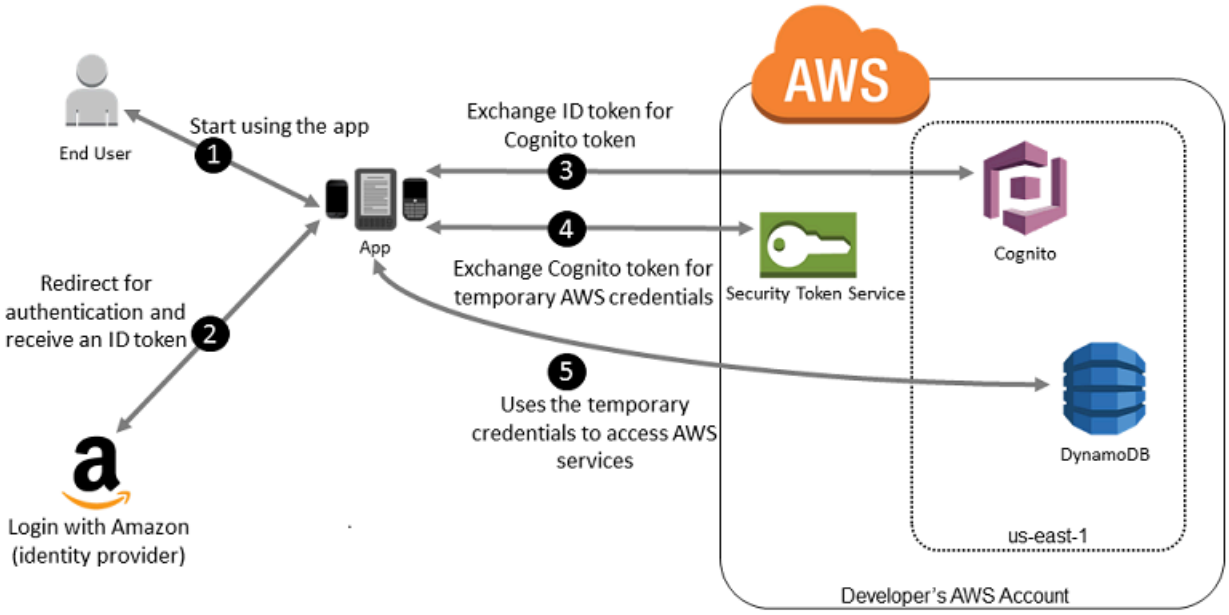
針對行動應用程式使用 Amazon Cognito

使用 OIDC 聯盟的首選方式是使用 Amazon Cognito。例如，開發人員 Adele 正在為行動裝置建置遊戲，其中分數和描述檔這類使用者資料都存放在 Amazon S3 和 Amazon DynamoDB 中。Adele 也會將此資料存放在裝置本機，並使用 Amazon Cognito 跨多個裝置保持同步。她了解為了安全和維護之故，長期的 AWS 安全憑證不應隨遊戲發散。她也了解遊戲可能有大量的使用者。針對所有這些原因，她不想要針對每個玩家在 IAM 建立新的使用者身分。反之，她會設計遊戲，讓使用者可以使用他們透過知名的外部身分提供者 (IdP) 建立的身分登入，例如 Login with Amazon、Facebook、Google 或任何 OpenID Connect (OIDC) 相容的 IdP。她的遊戲可善用其中一個提供者的身分驗證機制，藉以驗證使用者的身分。

為了使移動應用程序能夠訪問她的 AWS 資源，阿黛爾首先使用她選擇 IdPs的開發人員 ID 註冊。她也會使用這些提供者的每一個來設定應用程式。在包 AWS 帳戶 含遊戲的 Amazon S3 儲存貯體和 DynamoDB 表格中，Adele 使用 Amazon Cognito 建立 IAM 角色，以精確定義遊戲所需的許可。如果她使用的是 OIDC IdP，她也會建立 IAM OIDC 身分提供者實體，以便在其中的 Amazon Cognito 身分集區與 IdP 之間建立信任。 AWS 帳戶

在應用程式的程式碼中，Adele 呼叫用於之前設定的 IdP 的登入介面。IdP 會處理讓使用者登入的所有詳細資訊，而應用程式會從提供者取得 OAuth 存取權杖或 OIDC ID 權杖。Adele 的應用程序可以將此身份驗證信息交換為由訪問密鑰 ID，秘密 AWS 訪問密鑰和會話令牌組成的一組臨時安全憑據。然後，應用程序可以使用這些憑據訪問提供的 Web 服務 AWS。該應用程式僅限於許可，其定義於其擔任的角色。

下圖使用 Login with Amazon 做為 IdP，顯示一個有關如何運作的簡化流程。對於步驟 2，應用程式也可使用 Facebook、Google 或任何 OIDC 相容的 IdP，但此處未顯示。



1. 客戶在行動裝置啟動您的應用程式。該應用程式要求使用者登入。
2. 該應用程式會使用 Login with Amazon 資源來接受使用者的憑證。
3. 應用程式使用 Amazon Cognito API 操作 `GetId` 和 `GetCredentialsForIdentity`，用 Login with Amazon 字符交換 Amazon Cognito ID 字符。Amazon Cognito 已設定為信任您的 Login with Amazon 專案，會產生一個權杖，它與 AWS STS交換暫存工作階段憑證。
4. 該應用程序從 Amazon Cognito 接收暫時安全憑證。您的應用程式也可以使用 Amazon Cognito 中的基本 (傳統) 工作流程，從 AWS STS 使`AssumeRoleWithWebIdentity`用中擷取權杖。如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的身分集區 (聯合身分) 驗證流程。
5. 透過應用程式可使用暫時性安全憑證存取操作應用程式所需的任何 AWS 資源。與暫時性安全憑證相關聯的角色及指派政策，決定了可以存取哪些內容。

使用下列程序將您的應用程式設定為使用 Amazon Cognito 驗證使用者，並將 AWS 資源存取權授予應用程式。如需關於完成此案例的特定步驟，請參閱 Amazon Cognito 的說明文件。

1. (選用) 使用 Login with Amazon、Facebook、Google 或其他任何 OpenID Connect (OIDC) 相容的 IdP 註冊為開發人員，並使用提供者設定一個或多個應用程式。此步驟是可選的，因為 Amazon Cognito 也支援未經驗證的使用者存取權 (訪客)。
2. 去 Amazon Cognito 在 AWS Management Console. 使用 Amazon Cognito 精靈來建立身分集區，而該集區是一種容器，可讓 Amazon Cognito 用於為您的應用程式整理最終使用者身分。您可以在應用程式之間分享身分集區。當您設定身分集區時，Amazon Cognito 會建立一或兩個 IAM 角色 (一

個用於已驗證的身分，一個用於未經驗證的「訪客」身分)，其為 Amazon Cognito 使用者定義許可。

3. 整合 AWSAmplify 與您的應用程式，並匯入所需檔案以使用 Amazon Cognito。
4. 建立 Amazon Cognito 憑證提供者的執行個體，並傳遞身分集區 ID、您的 AWS 帳戶 號碼，以及與身分集區建立關聯之角色的 Amazon Resource Name (ARN)。中的 Amazon Cognito 精靈 AWS Management Console 提供可協助您開始使用的範例程式碼。
5. 當您的應用程序訪問 AWS 資源時，將憑據提供程序實例傳遞給客戶端對象，該客戶端對象將臨時安全憑據傳遞給客戶端。憑證的許可是根據您稍早定義的一個或多個角色。

如需詳細資訊，請參閱下列內容：

- 在 AWS Amplify 框架文檔中登錄（安卓）。
- 在 AWS Amplify 框架文檔中登錄（iOS）。

針對行動應用程式使用 OIDC 聯合 API 作業

若要取得最佳結果，請在幾乎所有 OIDC 聯合案例中使用 Amazon Cognito 做為身分識別代理程式。Amazon Cognito 易於使用，並提供額外功能，如匿名 (未經身分驗證) 存取，以及跨裝置和提供者的同步使用者資料。不過，如果您已透過手動呼叫 `AssumeRoleWithWebIdentity` API 建立使用 OIDC 聯盟的應用程式，您可以繼續使用該應用程式，而您的應用程式仍可正常運作。

在沒有 Amazon Cognito 的情況下使用 OIDC 聯盟的程序遵循以下一般概述：

1. 使用 IdP 註冊為開發人員，並使用外部身分提供者 (IdP) 設定您的應用程式，IdP 為您的應用程式提供唯一 ID。（不同的 IdPs 使用不同的術語這個過程。 此大綱使用術語配置，用於使用 IdP 識別您的應用程序的過程。）每個 IdP 都會為您提供該 IdP 唯一的應用程序 ID，因此，如果您將同一個應用程序配置為多個 IdPs，則您的應用程序將具有多個應用程序 ID。可依照每個提供商的要求配置多個應用程式。

   下列外部連結提供有關使用某些常用身分識別提供者 (IdPs) 的資訊：

   - Login with Amazon 開發人員中心
   - Facebook 開發人員網站上的新增 Facebook 登入到您的應用程式或網站。
   - Google 開發人員網站上的登入時使用 OAuth 2.0 (OpenID Connect)。

> **Important**
>
> 如果您使用來自谷歌、臉書或 Amazon Cognito 的 OIDC 身分供應商，請勿在中建立個別的 IAM 身分提供者。 AWS Management Console AWS 內建這些 OIDC 身分識別提供者，可供您使用。略過下列步驟，並直接移至使用您的身分提供者建立新角色。

2. 如果您使用與 OIDC 相容的 Google、Facebook 或 Amazon Cognito 以外的 IdP，請為其建立 IAM 身分提供者實體。
3. 在 IAM 中，建立一或多個角色。對於每個角色，定義可以擔任角色的人員 (信任政策) 以及應用程式使用者擁有的許可 (許可政策)。一般而言，您可以為應用程式所支援的每個 IdP 建立一個角色。例如，您可以建立使用者透過 Login with Amazon 登入之應用程式可擔任的角色，使用者透過 Facebook 登入之相同應用程式可擔任的次要角色，以及使用者透過 Google 登入之應用程式可擔任的第三個角色。對於信任關係，請將 IdP (像是 Amazon.com) 指定為`Principal` (信任實體) 和包含符合 IdP 指派的應用程式 ID 的 `Condition` 。針對第三方身分提供者建立角色 (聯合身分) 中會說明不同提供者的角色範例。
4. 在您的應用程式中，使用 IdP 對您的使用者進行身分驗證。有關如何執行此動作的具體情況會根據您所使用的 IdP (Login with Amazon、Facebook 或 Google) 以及您的應用程式執行的平台而有所不同。例如，Android 應用程序的身份驗證方法可能與 iOS 應用程序或 JavaScript基於 Web 應用程序的方法不同。

   一般而言，如果使用者尚未登入，則 IdP 會負責顯示登入頁面。在 IdP 驗證使用者之後，IdP 會向您的應用程式傳回帶有使用者資訊的身分驗證權杖。所包含的資訊取決於 IdP 公開的內容和使用者願意共用的資訊。您可以在您的應用程式中使用此資訊。
5. 在您的應用程式中，對 `AssumeRoleWithWebIdentity` 動作進行未簽署呼叫以請求臨時安全性憑證。在請求中，您會傳遞 IdP 的身份驗證令牌，並為您為該 IdP 建立的 IAM 角色指定 Amazon 資源名稱 (ARN)。 AWS 驗證令牌是否受信任且有效，如果是，則將臨時安全憑據返回給您的應用程序，該憑據具有您在請求中命名的角色的權限。回應還包含來自 IdP 的有關使用者的中繼資料，例如 IdP 與使用者關聯的唯一使用者 ID。
6. 使用響應中的臨時安全憑據，您的`AssumeRoleWithWebIdentity`應用程序向 AWS API 操作發出簽名請求。來自 IdP 的用戶 ID 信息可以區分應用程序中的用戶。例如，您可以將物件放入包含使用者 ID 做為首碼或尾碼的 Amazon S3 資料夾。這可讓您建立鎖定該資料夾的存取控制政策，以便只有具有該 ID 的使用者才能存取它。如需詳細資訊，請參閱 AWS STS 同盟使用者工作階段主體。
7. 您的應用程式必須快取臨時安全憑證，如此每次應用程式需要向 AWS發出請求時都不必取得新的憑證。在預設情況下，憑證可以使用一小時。當憑證過期 (或在此之前)，您再次呼叫 `AssumeRoleWithWebIdentity` 以取得一組新的臨時安全憑證。根據 IdP 以及如何管理權杖，

您可能需要在對 `AssumeRoleWithWebIdentity` 進行新呼叫之前重新整理 IdP 的權杖，因為 IdP 的權杖通常也會在固定時間後過期。如果您使用適用 AWS SDK for iOS 或適用 AWS SDK for Android，則可以使用 Amazonsts CredentialsProvider 動作來管理 IAM 臨時登入資料，包括視需要重新整理它們。

OIDC 聯盟

假設您正在建立可存取 AWS 資源的應用程式，例如使用工 GitHub作流程存取 Amazon S3 和 DynamoDB 的動作。

當您使用這些工作流程時，您會向必須使用 AWS 存取金鑰簽署的 AWS 服務提出要求。不過，我們強烈建議您不要在外部的應用程式中長期儲存 AWS 認證 AWS。而是使用 OIDC 聯盟設定您的應用程式，在需要時動態要求臨時 AWS 安全登入資料。提供的臨時認證對應至僅具有執行應用程式所需工作所需權限的 AWS 角色。

使用 OIDC 聯盟，您不需要建立自訂登入程式碼或管理自己的使用者身分識別。相反，您可以在應用程序中使用 OIDC，例如 GitHub 操作或任何其他 OpenID Connect（OIDC）兼容的 IdP，以進行身份驗證。 AWS他們收到身份驗證令牌（稱為 JSON Web 令牌（JWT），然後將該令牌交換為 AWS 該映射中的臨時安全憑據，以便將該令牌交換為具有在. AWS 帳戶使用 IdP 可協助您保護安 AWS 帳戶全，因為您不需要在應用程式中內嵌和散佈長期安全登入資料。

在大多數情況下，建議您使用 Amazon Cognito，因為它可以充當身分經紀人，並為您執行許多聯合工作。如需詳細資訊，請參閱下列章節：針對行動應用程式使用 Amazon Cognito。

> **Note**
>
> 由 OpenID Connect（OIDC）身份提供程序發布的 JSON 網絡令牌（JWT）在`exp`聲明中包含指定令牌何時過期的到期時間。如 OpenID Connect（OIDC）核心 1.0 標準允許的那樣，IAM 提供了超出 JWT 中指定的到期時間的五分鐘時段，以解決時鐘偏差。這意味著 IAM 在到期時間之後，但在這五分鐘內收到的 OIDC JWT 可以進行進一步的評估和處理。

主題

- 在 IAM 中建立 OpenID Connect (OIDC) 身分識別提供者
- 取得 OpenID Connect 身分識別提供者的指紋
- OIDC 聯盟的其他資源

在 IAM 中建立 OpenID Connect (OIDC) 身分識別提供者

IAM OIDC 身分提供者是 IAM 中的實體，負責描述可支援 OpenID Connect (OIDC) 標準的外部身分提供者 (IdP) 服務，例如，Google 或 Salesforce。如果想要在 OIDC 相容的 IdP 和您的 AWS 帳戶之間建立信任，請使用 IAM OIDC 身分提供者。這在建立需要 AWS 資源存取權的行動應用程式或 Web 應用程式時非常有用，但您不想建立自訂登入程式碼或管理您自己的使用者身分識別。如需有關此案例的詳細資訊，請參閱 the section called “OIDC 聯盟”。

您可以使用 AWS Management Console、、適用於 Windows PowerShell 的工具或 IAM API 來建立和管理 IAM OIDC 身分識別提供者。 AWS Command Line Interface

在您建立 IAM OIDC 身分提供者之後，您必須建立一個或多個 IAM 角色。角色是一種身份 AWS ，其中沒有自己的憑據（就像用戶一樣）。但在此情況中，角色是以動態指派給聯合身分使用者，而該使用者由您組織的身分提供者 (IdP) 進行驗證。角色可允許您組織的 IdP 請求暫時性安全憑證以存取 AWS。指派給角色的原則會決定允許同盟使用者在中 AWS執行的動作。若要為第三方身分提供者建立角色，請參閱 針對第三方身分提供者建立角色 (聯合身分)。

> **Important**
>
> 當您針對支援 `oidc-provider` 資源的動作設定以身分識別型原則時，IAM 會評估完整的 OIDC 身分識別提供者 URL，包含任何指定的路徑。如果您的 OIDC 身分識別提供者 URL 具有路徑，請務必在 `oidc-provider` ARN 中包含該路徑作為 `Resource` 要素值。您也可以選擇在 URL 路徑中的任何位置附加正斜線和萬用字元 (`/*`) 至 URL 網域，或使用萬用字元 (`*` 和 `?`)。如果要求中的 OIDC 身分識別提供者 URL 與政策的 `Resource` 要素中設定的值不相符，則請求會失敗。

若要疑難排解 IAM OIDC 聯盟的常見問題，請參閱在 Re: POST 上解決與 OIDC 相關的錯誤。 AWS

主題

- 先決條件：驗證身分識別提供者的組態
- 建立及管理 OIDC 提供者 (主控台)
- 建立及管理 IAM OIDC 身分提供者 (AWS CLI)
- 建立和管理 OIDC 身分識別提供者 (API)AWS

先決條件：驗證身分識別提供者的組態

建立 IAM OIDC 身分識別提供者之前，您必須先從 IdP 取得下列資訊。如需取得 OIDC 提供者組態資訊的詳細資訊，請參閱 IdP 的說明文件。

1. 判斷您的 OIDC 身分提供者的公開可用 URL。URL 必須以 https:// 開頭。根據 OIDC 標準，允許路徑組件，但查詢參數不允許。一般而言，網址只包含一個主機名稱，例如 https://server.example.org 或 https://example.com。URL 不應包含連接埠號碼。
2. 在 OIDC 身分識別提供者 URL 的末尾加上 /.well-已知/openid 組態，以查看提供者的公開可用設定文件和中繼資料。您必須具有 JSON 格式的探索文件，其中包含可從 OpenID Connect 提供者探索端點 URL 擷取的提供者組態文件和中繼資料。
3. 確認提供者的組態資訊中包含下列值。如果您的 openid 配置缺少這些字段中的任何一個，則必須更新您的發現文檔。此程序可能會因您的身分提供者而有所不同，因此請遵循 IdP 的說明文件來完成此任務。

   - 發行者：您網域的網址。
   - jwks_uri：IAM 取得您的公開金鑰的 JSON 網頁金鑰集 (JWKS) 端點。您的身分識別提供者必須在開放設定中包含 JSON 網頁金鑰集 (JWKS) 端點。此 URI 定義在何處獲取用於驗證身份提供者的簽名令牌的公鑰。
   - 宣告支援：有關使用者的資訊，可協助您確保來自 IdP 的 OIDC 驗證回應包含 IAM 政策中用來檢查聯合身分 AWS 使用者許可的必要屬性。如需可用於宣告的 IAM 條件金鑰清單，請參閱AWS OIDC 聯盟的可用金鑰。
     - AUD：您必須在 JSON 網絡令牌（JWT）中確定 IdP 問題的受眾聲明值。受眾（aud）聲明是特定於應用程序的，並標識令牌的預期接收者。當您向 OpenID Connect 提供商註冊移動或 Web 應用程序時，他們會建立用於識別該應用程序的客戶端 ID。客戶端 ID 是您的應用程序的唯一標識符，在 aud 身份驗證聲明中傳遞。在建立 IAM OIDC 身分提供者時，aud 聲明必須與對象值相符。
     - iat：聲明必須包含一個表`iat`示 ID 令牌發行時間的值。
     - iss：身分識別提供者的網址。網址必須以 https://開頭，且應該對應於提供給 IAM 的提供者網址。根據 OIDC 標準，允許路徑組件，但查詢參數不允許。一般而言，網址只包含一個主機名稱，例如 https://server.example.org 或 https://example.com。URL 不應包含連接埠號碼。
   - 支持響應類型：id 令牌
   - 支持的主題類型：公共
   - 支援的 ID 標記 _ 簽署值:RS256

Note

您可以在下面的示例中包含諸如 custom 之類的其他聲明；但是， AWS STS 將忽略該聲明。

```
{
  "issuer": "https://example-domain.com",
  "jwks_uri": "https://example-domain.com/jwks/keys",
  "claims_supported": [
    "aud",
    "iat",
    "iss",
    "name",
    "sub",
    "custom"
  ],
  "response_types_supported": [
    "id_token"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "subject_types_supported": [
    "public"
  ]
}
```

建立及管理 OIDC 提供者 (主控台)

按照這些指示，在 AWS Management Console中建立和管理 IAM OIDC 身分提供者。

Important

如果您正在使用來自 Google、Facebook 或 Amazon Cognito 的 OIDC 身分提供者，請勿使用此程序建立個別的 IAM 身分提供者。這些 OIDC 身分識別提供者已內建於中， AWS 並可供您使用。請依照下列步驟為您的身分提供者建立新角色，請參閱 為 OpenID Connect 聯盟（控制台）創建角色。

建立 IAM OIDC 身分提供者 (主控台)

1. 在建立 IAM OIDC 身分提供者之前，您必須使用 IdP 來註冊您的應用程式，才能接收用戶端 ID。用戶端 ID (也稱為觀眾) 是您在向 IdP 註冊應用程式時向您發佈的應用程式的唯一識別符。如需有關取得用戶端 ID 的詳細資訊，請參閱文件以取得您的 IdP。

   > **Note**
   >
   > AWS 透過我們受信任的根憑證授權單位 (CAIdPs) 程式庫，保護與某些 OIDC 身分識別提供者 () 的通訊，而不是使用憑證指紋來驗證您的 IdP 伺服器憑證。在這些情況下，舊式指紋會保留在您的組態中，但不再用於驗證。這些 OIDC IdPs 包括 Auth0、、 GitHub GitLab、谷歌和那些使用 Amazon S3 儲存貯體託管 JSON 網頁金鑰集 (JWKS) 端點的使用者。

2. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
3. 在導覽窗格中，請選擇 Identity providers (身分提供者)，然後選擇 Add provider (新增提供者)。
4. 對於 Configure provider (設定提供者)，選擇 OpenID Connect (OpenID 連線)。
5. 針對 Provider URL (提供者 URL)，輸入 IdP 的 URL。URL 必須符合這些限制：
   - URL 區分大小寫。
   - URL 必須以 **`https://`** 開頭。
   - URL 不應包含連接埠號碼。
   - 在您的 AWS 帳戶每個 IAM OIDC 身分識別提供者中都必須使用唯一的 URL。如果您嘗試提交已在中用於 OpenID Connect 提供程序的 URL AWS 帳戶，則會收到錯誤信息。
6. 對於「對象」，請輸入您在 IdP 中註冊並接收到的應用程式的用戶端 IDStep 1，以及向其發出要AWS求的應用程式。如果此 IdP 有其他用戶端 ID (也稱觀眾)，則稍後可以在提供者詳細資訊頁面上新增它們。

   > **Note**
   >
   > 如果您的 IdP JWT 權杖包含`azp`宣告，請輸入此值作為對象值。

7. (選擇性) 對於「新增標籤」，您可以新增金鑰與值配對，以協助您識別和組織您的. IdPs 您也可以使用標籤來控制對 AWS 資源的存取。若要進一步了解如何標記 IAM OIDC 身分提供者，請參閱 標記 OpenID Connect (OIDC) 身分提供者。選擇 Add tag (新增標籤)。輸入每個標籤鍵值組的值。

8. 請確認您提供的資訊。完成後，請選擇 Add provider (新增提供者)。IAM 將嘗試擷取並使用 OIDC IdP 伺服器憑證的最上層中繼 CA 指紋，以建立 IAM OIDC 身分識別提供者。

   > **Note**
   >
   > OIDC 身分識別提供者的憑證鏈結必須以網域或簽發者 URL 開頭，然後是中繼憑證，並以根憑證結束。如果憑證鏈結順序不同或包含重複或其他憑證，則您會收到簽章不符錯誤，且 STS 無法驗證 JSON Web Token (JWT)。更正伺服器傳回之鏈結中憑證的順序，以解決錯誤。如需有關憑證鏈標準的詳細資訊，請參閱 RFC 系列網站上 RFC 5246 中的憑證清單。

9. 將 IAM 角色指派給您的身分提供者，以授與身分識別提供者所管理的外部使用者身分識別，以存取帳戶中的 AWS 資源。若要深入了解如何建立聯合身分角色，請參閱 針對第三方身分提供者建立角色 (聯合身分)。

   > **Note**
   >
   > 角色信任原則中 IdPs 使用的 OIDC 必須與信任它的角色位於相同的帳戶中。

新增或移除 IAM OIDC 身分提供者的指紋 (主控台)

> **Note**
>
> AWS 透過我們受信任的根憑證授權單位 (CAIdPs) 程式庫，保護與某些 OIDC 身分識別提供者 () 的通訊，而不是使用憑證指紋來驗證您的 IdP 伺服器憑證。在這些情況下，舊式指紋會保留在您的組態中，但不再用於驗證。這些 OIDC IdPs 包括 Auth0、、 GitHub GitLab、谷歌和那些使用 Amazon S3 儲存貯體託管 JSON 網頁金鑰集 (JWKS) 端點的使用者。

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，請選擇 Identity providers (身分提供者)。然後選擇您要更新的 IAM 身分提供者名稱。
3. 在 Thumbprints (指紋) 區段中，請選擇 Manage (管理)。若要輸入新的指紋值，請選擇 Add thumbprint 新增指紋。若要移除指紋，請選擇要移除的指紋旁的 Remove (移除)。

> **Note**
>
> IAM OIDC 身分提供者必須至少有 1 個和最多可有 5 個指紋。

完成後，請選擇 Save changes (儲存變更)。

新增 IAM OIDC 身分提供者的對象 (主控台)

1. 在導覽窗格中，選擇 Identity providers (身分提供者)，然後選擇您所要更新 IAM 身分提供者的名稱。
2. 在 Audiences (對象) 區段中，選擇 Actions (動作) 並選取 Add audience (新增對象)。
3. 輸入您向 IdP 註冊並接收到的應用程式的用戶端 IDStep 1，該用戶端 ID 將向其發出要 AWS求。然後選擇 Add audiences (新增對象)。

> **Note**
>
> IAM OIDC 身分提供者必須至少有 1 個和最多可有 100 個對象。

移除 IAM OIDC 身分提供者的對象 (主控台)

1. 在導覽窗格中，選擇 Identity providers (身分提供者)，然後選擇您所要更新 IAM 身分提供者的名稱。
2. 在 Audiences (對象) 區段中，選取您要移除之對象旁邊的選項按鈕，然後選取 Actions (動作)。
3. 選擇 Remove audience (移除對象)。新的視窗將開啟。
4. 如果您移除對象，與對象聯合的身分無法擔任與對象相關聯的角色。在視窗中，閱讀警告，並在欄位中輸入單字 `remove`，確認您要移除對象。
5. 選擇 Remove (移除) 以移除對象。

刪除 IAM OIDC 身分提供者 (主控台)

1. 在以下網址開啟 IAM 主控台：https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，請選擇 Identity providers (身分提供者)。
3. 在您要刪除的 IAM 身分提供者旁邊，勾選核取方塊。新的視窗將開啟。

4. 在欄位中輸入單字 delete，確認您要刪除提供者。再選擇 Delete (刪除)。

建立及管理 IAM OIDC 身分提供者 (AWS CLI)

您可以使用下列 AWS CLI 命令來建立和管理 IAM OIDC 身分識別提供者。

建立 IAM OIDC 身分提供者 (AWS CLI)

1. (選用) 若要取得 AWS 帳戶中所有 IAM OIDC 身分提供者的清單，請執行下列命令：
   - aws iam list-open-id-connect-providers
2. 若要建立新的 IAM OIDC 身分提供者，請執行下列命令：
   - aws iam create-open-id-connect-provider

更新現有 IAM OIDC 身分提供者的伺服器憑證指紋清單 (AWS CLI)

- 若要更新現有 IAM OIDC 身分提供者的伺服器憑證指紋清單，請執行下列命令：
  - aws iam update-open-id-connect-provider-thumbprint

標記現有的 IAM OIDC 身分提供者 (AWS CLI)

- 若要標記現有 IAM OIDC 身分提供者，請執行下列命令：
  - aws iam tag-open-id-connect-provider

列出現有 IAM OIDC 身分提供者 (AWS CLI) 的標籤

- 若要列出現有 IAM OIDC 身分提供者的標籤，請執行下列命令：
  - aws iam list-open-id-connect-provider-tags

移除 IAM OIDC 身分提供者 (AWS CLI) 的標籤

- 若要移除 IAM OIDC 身分提供者的標籤，請執行下列命令：
  - aws iam untag-open-id-connect-provider

從現有 IAM OIDC 身分提供者新增或移除用戶端 ID (AWS CLI API)

1. (選用) 若要取得 AWS 帳戶中所有 IAM OIDC 身分提供者的清單，請執行下列命令：
   - aws iam list-open-id-connect-providers
2. (選用) 若要取得有關 IAM OIDC 身分提供者的詳細資訊，請執行下列命令：
   - aws iam get-open-id-connect-provider
3. 若要將新的用戶端 ID 新增到現有 IAM OIDC 身分提供者，請執行下列命令：
   - aws iam add-client-id-to-open-id-connect-provider
4. 若要從現有 IAM OIDC 身分提供者中移除用戶端，請執行下列命令：
   - aws iam remove-client-id-from-open-id-connect-provider

若要刪除 IAM OIDC 身分提供者 (AWS CLI)

1. (選用) 若要取得 AWS 帳戶中所有 IAM OIDC 身分提供者的清單，請執行下列命令：
   - aws iam list-open-id-connect-providers
2. (選用) 若要取得有關 IAM OIDC 身分提供者的詳細資訊，請執行下列命令：
   - aws iam get-open-id-connect-provider
3. 若要刪除 IAM OIDC 身分提供者，請執行下列命令：
   - aws iam delete-open-id-connect-provider

建立和管理 OIDC 身分識別提供者 (API)AWS

您可以使用以下 IAM API 命令來建立及管理 OIDC 提供者。

若要建立 IAM OIDC 身分識別提供者 (API)AWS

1. (選用) 若要取得 AWS 帳戶中所有 IAM OIDC 身分提供者的清單，請呼叫下列操作：
   - ListOpenIDConnectProviders
2. 若要建立新的 IAM OIDC 身分提供者，請呼叫下列操作：
   - CreateOpenIDConnectProvider

更新現有 IAM OIDC 身分識別提供者 (API) 的伺服器憑證指紋清單AWS

- 若要更新現有 IAM OIDC 身分提供者的伺服器憑證指紋清單，請呼叫下列操作：
  - UpdateOpenIDConnectProviderThumbprint

標記現有的 IAM OIDC 身分識別提供者 (API)AWS

- 若要標記現有 IAM OIDC 身分提供者，請叫用下列操作：
  - TagOpenIDConnectProvider

列出現有 IAM OIDC 身分識別提供者 (AWS API) 的標籤

- 若要列出現有 IAM OIDC 身分提供者的標籤，請叫用下列操作：
  - ListOpenIDConnectProviderTags

若要移除現有 IAM OIDC 身分識別提供者 (AWS API) 上的標籤

- 若要移除現有 IAM OIDC 身分提供者的標籤，請叫用下列操作：
  - UntagOpenIDConnectProvider

從現有 IAM OIDC 身分提供者新增或移除用戶端 ID (AWS API)

1. (選用) 若要取得 AWS 帳戶中所有 IAM OIDC 身分提供者的清單，請呼叫下列操作：
   - ListOpenIDConnectProviders
2. (選用) 若要取得有關 IAM OIDC 身分提供者的詳細資訊，請呼叫下列操作：
   - GetOpenIDConnectProvider
3. 若要將新的用戶端 ID 新增到現有 IAM OIDC 身分提供者，請呼叫下列操作：
   - AddClientIDToOpenIDConnectProvider
4. 若要從現有 IAM OIDC 身分提供者中移除用戶端 ID，請呼叫下列操作：
   - RemoveClientIDFromOpenIDConnectProvider

若要刪除 IAM OIDC 身分識別提供者 (API)AWS

1. (選用) 若要取得 AWS 帳戶中所有 IAM OIDC 身分提供者的清單，請呼叫下列操作：
   - ListOpenIDConnectProviders
2. (選用) 若要取得有關 IAM OIDC 身分提供者的詳細資訊，請呼叫下列操作：
   - GetOpenIDConnectProvider
3. 若要刪除 IAM OIDC 身分提供者，請呼叫下列操作：
   - DeleteOpenIDConnectProvider

取得 OpenID Connect 身分識別提供者的指紋

當您在 IAM 中建立 OpenID Connect (OIDC) 身分識別提供者時，IAM 需要針對簽署外部身分識別提供者 (IdP) 所使用之憑證的頂級中繼憑證授權單位 (CA) 提供指紋。指紋是 CA 憑證的簽章，該簽章會用來發出與 OIDC 相容的 IdP 憑證。當您建立 IAM OIDC 身分識別提供者時，您信任該 IdP 驗證的身分可存取您的. AWS 帳戶藉由使用 CA 的憑證指紋，您可以信任該 CA 所發行的任何憑證，其 DNS 名稱與已註冊的憑證相同。這讓您在續約 IdP 的簽署憑證時，無需在每個帳戶中更新信任。

> **Important**
>
> 在大多數情況下，聯合伺服器會使用兩個不同的憑證：
>
> - 第一個會在 AWS 和您的 IdP 之間建立 HTTPS 連線。這應該由知名的公共根 CA 發行，例如 AWS Certificate Manager。這可讓用戶端檢查憑證的可靠性和狀態。
> - 第二個憑證用於加密令牌，且應由私有或公有根 CA 簽署。

您可以使用AWS Command Line Interface、適用於 Windows PowerShell 的工具或 IAM API 建立 IAM OIDC 身分識別提供者。使用這些方法時，您可以選擇手動提供指紋。如果您選擇不包含指紋，IAM 將擷取 OIDC IdP 伺服器憑證的最上層中繼 CA 指紋。如果您選擇包含指紋，您必須手動取得指紋並將其提供給。 AWS

當您使用 IAM 主控台建立 OIDC 身分識別提供者時，IAM 會嘗試為您擷取 OIDC IdP 伺服器憑證的最上層中繼 CA 指紋。

我們建議您也手動取得 OIDC IdP 的指紋，並驗證 IAM 擷取正確的指紋。如需取得憑證指紋的相關資訊，請參閱下列各節。

> **Note**
>
> AWS 透過我們受信任的根憑證授權單位 (CAIdPs) 程式庫，保護與某些 OIDC 身分識別提供者 () 的通訊，而不是使用憑證指紋來驗證您的 IdP 伺服器憑證。在這些情況下，舊式指紋會保留在您的組態中，但不再用於驗證。這些 OIDC IdPs 包括 Auth0、、 GitHub GitLab、谷歌和那些使用 Amazon S3 儲存貯體託管 JSON 網頁金鑰集 (JWKS) 端點的使用者。

若要疑難排解 IAM OIDC 聯盟的常見問題，請參閱在 Re: POST 上解決與 OIDC 相關的錯誤。 AWS

取得憑證指紋

您可以使用網頁瀏覽器和 OpenSSL 命令列工具來取得 OIDC 提供者的憑證指紋。不過，您不需要手動取得憑證指紋即可建立 IAM OIDC 身分識別提供者。您可以使用下列程序來取得 OIDC 提供者的憑證指紋。

取得 OIDC IdP 的指紋

1. 您需要先取得 OpenSSL 命令列工具，然後才能取得 OIDC IdP 的指紋。您可使用此工具下載 OIDC IdP 憑證鍊並產生憑證連結中最終憑證的指紋。如果您需要安裝和設定 OpenSSL，請根據安裝 OpenSSL 與 設定 OpenSSL 中的說明操作。
2. 從 OIDC IdP URL 開始 (例如，`https://server.example.com`)，然後新增 `/.well-known/openid-configuration` 以組成該 IdP 的組態文件的 URL，如下所示：

   `https://server.example.com/.well-known/openid-configuration`

   在 Web 瀏覽器中開啟此 URL，並將 *`server.example.com`* 取代為 IdP 的伺服器名稱。
3. 在顯示的文件中，使用您的 web 瀏覽器 Find (尋找) 功能來尋找文字 `"jwks_uri"`。文字 `"jwks_uri"` 後面會有一個冒號 (:)，然後是一個 URL。複製 URL 的完整網域名稱。不包括 `https://` 或在頂層網域後的任何路徑。

   ```
   {
    "issuer": "https://accounts.example.com",
    "authorization_endpoint": "https://accounts.example.com/o/oauth2/v2/auth",
    "device_authorization_endpoint": "https://oauth2.exampleapis.com/device/code",
    "token_endpoint": "https://oauth2.exampleapis.com/token",
    "userinfo_endpoint": "https://openidconnect.exampleapis.com/v1/userinfo",
    "revocation_endpoint": "https://oauth2.exampleapis.com/revoke",
    "jwks_uri": "https://www.exampleapis.com/oauth2/v3/certs",
   ...
   ```

4. 使用 OpenSSL 命令列工具來執行以下命令。將 *keys.example.com* 取代為您在Step 3 中取得的網域名稱。

```
openssl s_client -servername keys.example.com -showcerts -
connect keys.example.com:443
```

5. 在命令視窗中向上滾動，直至看到類似於以下範例的憑證。如果您看到多個憑證，請找到顯示的最後一個憑證 (在命令輸出末尾)。這包含憑證授權機構鏈中的頂層中繼 CA 憑證。

```
-----BEGIN CERTIFICATE-----
 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
 21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
 rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
 nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
 NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
 -----END CERTIFICATE-----
```

複製憑證 (包括 `-----BEGIN CERTIFICATE-----` 與 `-----END CERTIFICATE-----` 行) 並將其貼上文字檔中。接著儲存檔案，並將其命名為 **certificate.crt**。

> **Note**
>
> OIDC 身分識別提供者的憑證鏈結必須以網域或簽發者 URL 開頭，然後是中繼憑證，並以根憑證結束。如果憑證鏈結順序不同或包含重複或其他憑證，則您會收到簽章不符錯誤，且 STS 無法驗證 JSON Web Token (JWT)。更正伺服器傳回之鏈結中憑證的順序，以解決錯誤。如需有關憑證鏈標準的詳細資訊，請參閱 RFC 系列網站上 RFC 5246 中的憑證清單。

6. 使用 OpenSSL 命令列工具來執行以下命令。

```
openssl x509 -in certificate.crt -fingerprint -sha1 -noout
```

您的命令視窗將顯示類似於以下範例的憑證指紋：

```
SHA1 Fingerprint=99:0F:41:93:97:2F:2B:EC:F1:2D:DE:DA:52:37:F9:C9:52:F2:0D:9E
```

從此字串中去掉冒號 (:) 字元以產生最終指紋，如：

```
990F4193972F2BECF12DDEDA5237F9C952F20D9E
```

7. 如果您要使用 Windows PowerShell 適用的工具或 IAM API 建立 IAM OIDC 身分識別提供者，則可選擇提供指紋。 AWS CLI如果您在建立期間選擇不包含指紋，IAM 將擷取 OIDC IdP 伺服器憑證的最上層中繼 CA 指紋。建立 IAM OIDC 身分識別提供者之後，您可以將此指紋與 IAM 擷取的指紋進行比較。

   如果您要在 IAM 主控台中建立 IAM OIDC 身分識別提供者，則主控台會嘗試為您擷取 OIDC IdP 伺服器憑證的頂級中繼 CA 指紋。您可以將此指紋與 IAM 擷取的指紋進行比較。建立 IAM OIDC 身分識別提供者之後，您可以在 OIDC 提供者摘要主控台頁面的端點驗證索引標籤中，檢視 IAM OIDC 身分識別提供者的指紋。

   > **Important**
   >
   > 如果您取得的指紋與您在 IAM OIDC 身分識別提供者指紋詳細資料中看到的指紋不符，則不應使用 OIDC 提供者。相反地，您應該刪除建立的 OIDC 提供者，然後在一段時間過後再次嘗試建立 OIDC 提供者。在使用提供者之前，請先確認指紋相符。如果這些指紋在第二次嘗試後仍然不相符，則請使用 IAM 論壇與 AWS聯絡。

安裝 OpenSSL

如果您尚未安裝 OpenSSL，請遵循此節中的說明操作。

在 Linux 或 Unix 上安裝 OpenSSL

1. 移至 OpenSSL: Source, Tarballs (https://openssl.org/source/)。
2. 下載最新來源並建置套件。

在 Windows 安裝 OpenSSL

1. 請前往 OpenSSL: Binary Distributions (https://wiki.openssl.org/index.php/Binaries)，以取得您可以從中安裝 Windows 版本的網站清單。
2. 依照您所選網站上的指示開始安裝。
3. 如果系統要求您安裝 Microsoft Visual C++ 2008 可轉散發套件，並且您的系統上尚未安裝此版本，請選擇適合您環境的下載連結。請遵照 Microsoft Visual C++ 2008 可轉散發套件安裝精靈所提供的指示執行。

   > **Note**
   >
   > 如果您不確定 Microsoft Visual C++ 2008 可轉散發套件是否已經安裝在您的系統上，您可以嘗試先安裝 OpenSSL。如果尚未安裝 Microsoft Visual C++ 2008 可轉散發套件，則 OpenSSL 安裝程式將會顯示提醒。請確定您安裝的架構 (32 位元或 64 位元) 符合您所安裝的 OpenSSL 版本。

4. 安裝 Microsoft Visual C++ 2008 可轉散發套件後，請選取適用於您環境的 OpenSSL 二進位檔版本並將檔案儲存在本機。啟動 OpenSSL 安裝精靈。
5. 遵循 OpenSSL 安裝精靈中所述的指示。

設定 OpenSSL

在使用 OpenSSL 命令之前，您必須先設定作業系統，使其具有 OpenSSL 安裝位置的相關資訊。

在 Linux 或 Unix 上設定 OpenSSL

1. 在命令列，將 `OpenSSL_HOME` 變數設定為 OpenSSL 安裝的位置：

   ```
   $ export OpenSSL_HOME=path_to_your_OpenSSL_installation
   ```

2. 設定要包含 OpenSSL 安裝的路徑：

   ```
   $ export PATH=$PATH:$OpenSSL_HOME/bin
   ```

> **Note**
>
> 您使用 `export` 命令對環境變數做出的任何變更，僅適用於目前的工作階段。您可以在 shell 組態檔案中設定環境變數，對環境變數進行持續變更。如需詳細資訊，請參閱適用於您作業系統的文件。

在 Windows 上設定 OpenSSL

1. 開啟命令提示視窗。
2. 將 `OpenSSL_HOME` 變數設定為 OpenSSL 安裝的位置：

```
C:\> set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

3. 將 `OpenSSL_CONF` 變數設定為 OpenSSL 安裝中組態檔案的位置：

```
C:\> set OpenSSL_CONF=path_to_your_OpenSSL_installation\bin\openssl.cfg
```

4. 設定要包含 OpenSSL 安裝的路徑：

```
C:\> set Path=%Path%;%OpenSSL_HOME%\bin
```

> **Note**
>
> 您在命令提示視窗中對 Windows 環境變數做出的任何變更，僅適用於目前命令列工作階段。您可以將環境變數設定為系統屬性，對環境變數進行持續變更。確切的程序將取決於您索使用的 Windows 版本。(例如，在 Windows 7 中，開啟控制面板、系統與安全、系統。 然後選擇進階系統設定、進階索引標籤、環境變數。) 如需詳細資訊，請參閱 Windows 文件。

OIDC 聯盟的其他資源

下列資源可協助您進一步瞭解 OIDC 聯盟：

- 通過在 Amazon Web Services 中配置 OpenID Connect 在您的 GitHub 工作流程中使用 OpenID

- 適用於 Android 的 Amplify 程式庫指南中的 Amazon Cognito 身分和適用於 Swift 的 Amplify 程式庫指南中的 Amazon Cognito 身分。
- 透過AWS 合作夥伴網路 (APN) 部落格上的 Microsoft Entra ID 自動化 OpenID 型 AWS IAM 網頁身分識別角色，逐步瞭解如何驗證在使用 OIDC 授權以外執行的自動化背景程序或應 AWS 用 machine-to-machine 程式。
- 行動應用程式的 Web 身分聯盟文章討論 OIDC 聯盟，並顯示如何使用 OIDC 聯合來存取 Amazon S3 中內容的範例。

SAML 2.0 聯合身分

AWS 支援與 SAML 2.0 (安全性宣告標記語言 2.0) 聯合身分識別，這是許多身分識別提供者 (IdPs) 使用的開放標準。此功能可啟用聯合單一登入 (SSO)，因此使用者可以登入 AWS Management Console 或呼叫 AWS API 作業，而不必為組織中的每個人建立 IAM 使用者。透過使用 SAML，您可以簡化設定聯合的程序 AWS，因為您可以使用 IdP 的服務，而不需要撰寫自訂身分識別 Proxy 程式碼。

IAM 聯合身分支援這些使用案例：

- 聯合存取可讓組織中的使用者或應用程式呼叫 AWS API 作業。本使用案例將在下一節中討論。您可以使用組織內產生的 SAML 聲明 (身分驗證回應的一部分) 獲得臨時安全性憑證。此方案類似於 IAM 支援的其他聯合身分方案，如 請求暫時性安全憑證 和 OIDC 聯盟 中說明之情況。不過，組織中的 SAML 2.0 架構會 IdPs 在執行階段處理許多詳細資料，以執行驗證和授權檢查。
- AWS Management Console 從您的組織到網頁式單一登入 (SSO)。使用者可以登入組織中由 SAML 2.0 相容 IdP 所代管的入口網站，選取要移至的選項，然後重新導向至主控台 AWS，而不必提供其他登入資訊。您可以使用第三方 SAML IdP 建立對主控台的 SSO 存取，或您還可以建立自訂 IdP 來支援外部使用者的主控台存取。如需有關建構自訂 IdP 的詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

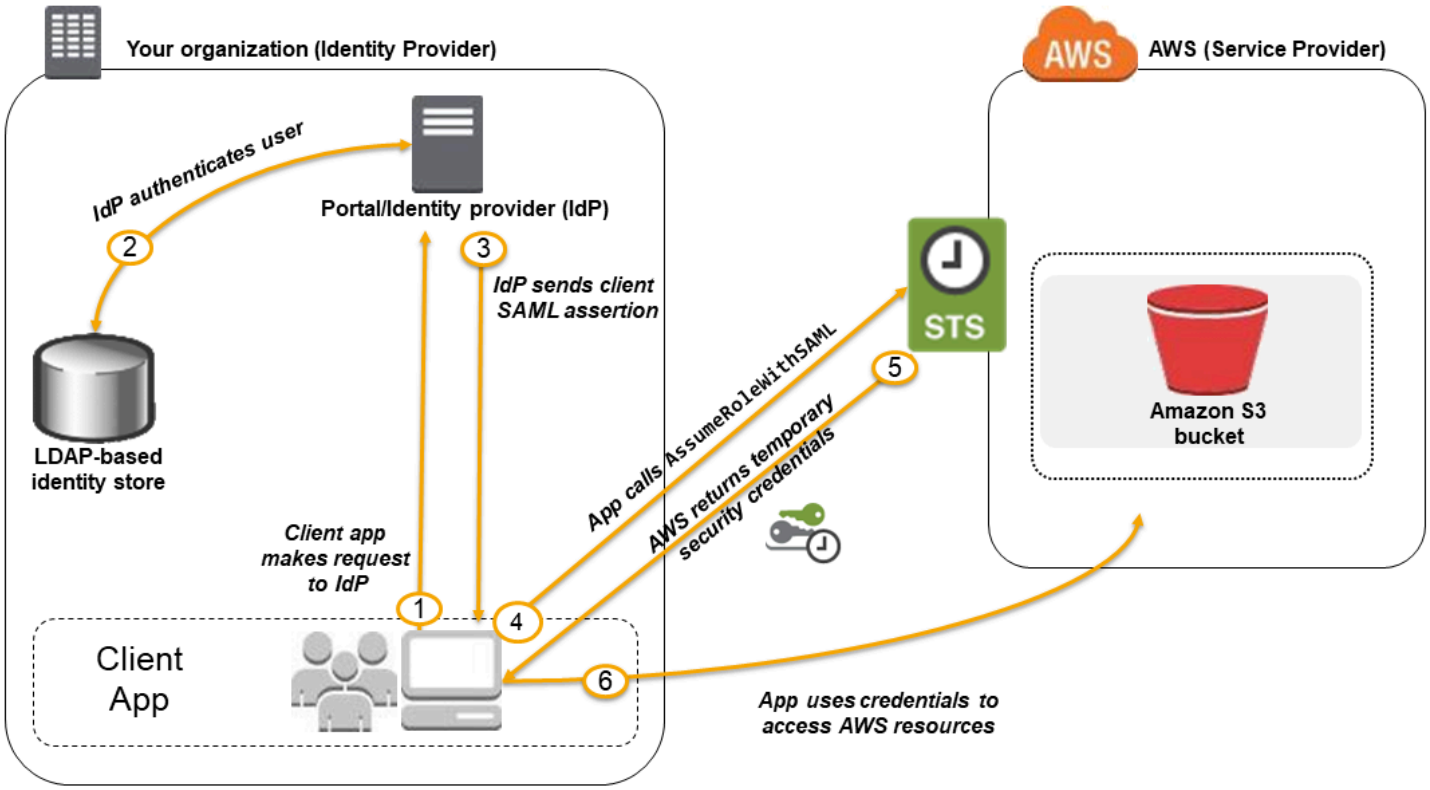
主題

- 使用以 SAML 為基礎的聯合身分來對 AWS進行 API 存取
- 配置以 SAML 2.0 為基礎的聯合身分驗證的概觀
- 允許 SAML 聯合存取資源的角色概觀 AWS
- 單獨辨識以 SAML 為基礎的聯合身分中的使用者
- 在 IAM 中建立 SAML 身分識別提供者
- 使用信賴方信任和新增宣告來設定您的 SAML 2.0 IdP

- 將第三方 SAML 解決方案提供者與 AWS
- 設定驗證回應的 SAML 宣告
- 啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console

使用以 SAML 為基礎的聯合身分來對 AWS進行 API 存取

假設您想要為員工提供一種方式，使其能將資料從他們的電腦中複製到備份資料夾。您可以建構一個可在使用者的電腦上執行的應用程式。在後端，應用程式讀取和寫入 Amazon S3 儲存貯體中的物件。使用者無法直接存取 AWS。而應使用以下程序：



1. 組織中的使用者會使用用戶端應用程式來請求獲得組織 IdP 的身分驗證。
2. IdP 根據組織的身分存放區對使用者進行身分驗證。
3. IdP 會建構一個具有使用者相關資訊的 SAML 聲明，並將此聲明發送到用戶端應用程式。
4. 用戶端應用程式會呼叫 AWS STS `AssumeRoleWithSAML`API、傳遞 SAML 提供者的 ARN、要承擔之角色的 ARN，以及來自 IdP 的 SAML 宣告。
5. API 對用戶端應用程式的回應包括臨時性安全憑證。
6. 用戶端應用程式使用臨時安全性憑證來呼叫 Amazon S3 API 操作。

配置以 SAML 2.0 為基礎的聯合身分驗證的概觀

您必須先設定組織的 IdP 和彼此信任，才能如前述案例和圖表中所述使用 SAML 2.0 型聯合。 AWS 帳戶 以下步驟介紹了用於配置此信任關係的一般過程。組織內部必須有支援 SAML 2.0 的 IdP，例如 Microsoft Active Directory Federation Service (AD FS，Windows Server 的一部分)、Shibboleth 或其他相容的 SAML 2.0 提供者。

> **Note**
>
> 若要改善聯合彈性，建議您將 IdP 和 AWS 聯合設定為支援多個 SAML 登入端點。如需詳細資訊，請參閱 AWS 安全性部落格文章如何使用地區性 SAML 端點進行容錯移轉。

設定組織的 IdP 並互 AWS 相信任

1. 向您組織的 IdP 註冊 AWS 為服務提供者 (SP)。使用 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 中的 SAML 中繼資料文件

   對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。

   您可以選擇性地使用 `https://signin.aws.amazon.com/static/saml-metadata.xml` 中的 SAML 中繼資料文件。
2. 使用您組織的 IdP 時，您會產生一個同等中繼資料 XML 檔，該檔案可將您的 IdP 描述為 AWS中的 IAM 身分提供者。它必須包含發行者名稱、建立日期、到期日，以及 AWS 可用來驗證組織的驗證回應 (宣告) 的金鑰。
3. 在 IAM 主控台中，您可以建立 SAML 身分識別提供者。作為此程序的一部分，您會上傳由中組織中的 IdP 產生的 SAML 中繼資料文件。Step 2如需詳細資訊，請參閱 在 IAM 中建立 SAML 身分識別提供者。
4. 在 IAM 中，建立一或多個 IAM 角色。在角色的信任原則中，您可以將 SAML 提供者設定為主參與者，以便在您的組織與 AWS. 該角色的許可政策將決定允許您組織的使用者在 AWS中執行的操作。如需詳細資訊，請參閱 針對第三方身分提供者建立角色 (聯合身分)。

   > **Note**
   >
   > 角色信任政策中使用的 SAML IDP 必須與角色所在的帳戶相同。

5. 在您組織的 IdP 中，定義可將組織中的使用者或群組映射到 IAM 角色的聲明。請注意，組織中不同的使用者和群組可能映射到不同的 IAM 角色。執行映射的確切步驟取決於您使用的 IdP。在使

用者的 Amazon S3 資料夾的早期方案中，可能出現所有使用者映射到提供 Amazon S3 許可的同一角色的情況。如需詳細資訊，請參閱 設定驗證回應的 SAML 宣告。

如果您的 IdP 對 AWS 主控台啟用 SSO，則您可以設定主控台工作階段的持續時間上限。如需詳細資訊，請參閱 啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console。

6. 在您建立的應用程式中，呼叫 AWS Security Token Service `AssumeRoleWithSAML` API，將您建立的 SAML 提供者的 ARN 傳遞給它Step 3、假設您在其中建立的角色 ARN，以及您從 IdP 取得之目前使用者的 SAML 宣告。Step 4 AWS 確保假定角色的請求來自 SAML 提供者中參考的 IdP。

   如需詳細資訊，請參閱 AWS Security Token Service API 參考資料中的AssumeRole使用 SAML。

7. 如果請求成功，API 會傳回一組臨時安全性憑證，您的應用程式即可用其向 AWS發出已簽署的請求。您的應用程式具有有關目前使用者的資訊並可存取 Amazon S3 中使用者特定的資料夾，如上一方案中所述。

允許 SAML 聯合存取資源的角色概觀 AWS

您在 IAM 中建立的一或多個角色可定義組織中允許在 AWS中執行操作的聯合身分使用者。當您為角色建立信任政策時，您可以將先前建立的 SAML 提供者指定為 `Principal`。此外，您還可以使用 `Condition` 設定信任政策的範圍，以便僅允許與特定 SAML 屬性相符的使用者存取角色。例如，您可以指定僅允許 SAML 從屬關係為 `staff` (在 https://openidp.feide.no 中聲明) 的使用者存取角色，如以下範例政策所示：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/
ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {
      "StringEquals": {
        "saml:aud": "https://signin.aws.amazon.com/saml",
        "saml:iss": "https://openidp.feide.no"
      },
      "ForAllValues:StringLike": {"saml:edupersonaffiliation": ["staff"]}
    }
  }]
}
```

> **Note**
>
> 角色信任政策中使用的 SAML IDP 必須與角色所在的帳戶相同。

如需有關您可以在政策中簽署的 SAML 索引鍵的詳細資訊，請參閱 SAML AWS STS 聯合身分的可用鍵。

您可以包括位於 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 的 `saml:aud` 屬性的區域端點。對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。

對於該角色中的許可政策，您可以像任何角色一樣指定許可。例如，如果允許組織中的使用者管理 Amazon 彈性運算雲端執行個體，您必須在許可政策中明確允許 Amazon EC2 動作，例如 AmazonEC2 FullAccess 受管政策中的動作。

單獨辨識以 SAML 為基礎的聯合身分中的使用者

在 IAM 中建立存取政策時，可根據使用者的身分指定許可，這一點通常很有用。舉例來說，對於已使用 SAML 聯合身分的使用者，應用程式可能希望使用如下的結構保留 Amazon S3 中的資訊：

```
myBucket/app1/user1
myBucket/app1/user2
myBucket/app1/user3
```

您可以透過 Amazon S3 主控台或建立儲存貯體 (`myBucketapp1`) 和資料夾 () AWS CLI，因為這些都是靜態值。不過，使用者特定的資料夾 (*user1*、*user2*、*user3* 等等) 必須在執行階段使用程式碼建立，因為在使用者首次透過聯合身分程序登入時，用來識別使用者的值未知。

若要編寫在資源名稱中引用特定於使用者的詳細資訊的政策，必須在可以用於政策條件的 SAML 索引鍵中提供使用者身分。以下索引鍵可供以 SAML 2.0 為基礎的聯合身分在 IAM 政策中使用。您可以使用以下索引鍵傳回的值為資源 (如 Amazon S3 資料夾) 建立唯一的使用者識別碼。

- `saml:namequalifier`. 雜湊值，以 `Issuer` 回應值 (`saml:iss`)、含 AWS 帳戶 ID 的字串與 IAM 中 SAML 提供者的易記名稱 (ARN 的最後一部分) 的串聯為基礎。帳戶 ID 與 SAML 提供者的易記名稱的串聯可作為索引鍵 `saml:doc` 供 IAM 政策使用。帳戶 ID 與提供者名稱必須使用「/」分隔，例如「123456789012/provider_name」。如需詳細資訊，請參閱 `saml:doc` 中的 SAML AWS STS 聯合身分的可用鍵 索引鍵。

NameQualifier 與 Subject 的組合可用於單獨辨識聯合身分使用者。下列虛擬程式碼顯示這個值是如何計算出來的。在此虛擬程式碼中，+ 表示串聯，SHA1 代表使用 SHA-1 產生訊息摘要的功能，Base64 64 代表產生雜湊輸出的 Base-64 編碼版本的功能。

```
Base64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/
MySAMLIdP" ) )
```

如需有關可用於以 SAML 為基礎的聯合身分政策索引鍵的詳細資訊，請參閱 SAML AWS STS 聯合身分的可用鍵。

- saml:sub (string). 這是該陳述的主題，其中包含單獨辨識組織中某個使用者的值 (例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3)。
- saml:sub_type (string). 此索引鍵可以是 persistent、transient 或在您的 SAML 聲明中使用的 Format 與 Subject 元素的完整 NameID URI。persistent 的值表示在所有工作階段中使用者的 saml:sub 值是相同的。如果值為 transient，則使用者在每個工作階段中擁有不同的 saml:sub 值。如需 NameID 元素的 Format 屬性的詳細資訊，請參閱 設定驗證回應的 SAML 宣告。

以下範例顯示了一個許可政策，該政策使用上述索引鍵為 Amazon S3 中的使用者特定資料夾授予許可。該政策假設 Amazon S3 物件使用同時包含 saml:namequalifier 與 saml:sub 的字首識別。請注意，Condition 元素包括一個測試，用於確保 saml:sub_type 設為 persistent。如果已設為 transient，每個工作階段使用者的 saml:sub 值可能不同，因此不應使用值的組合來辨識使用者特定的資料夾。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}",
      "arn:aws:s3:::exampleorgBucket/backup/${saml:namequalifier}/${saml:sub}/*"
    ],
    "Condition": {"StringEquals": {"saml:sub_type": "persistent"}}
  }
```

```
}
```

如需有關從 IdP 映射聲明到政策索引鍵的詳細資訊，請參閱 設定驗證回應的 SAML 宣告。

在 IAM 中建立 SAML 身分識別提供者

IAM SAML 2.0 身分提供者是 IAM 中的實體，其負責描述支援 SAML 2.0 (安全性聲明標記語言 2.0) 標準的外部身分提供者 (IdP) 服務。當您想要在 SAML 相容 IdP (例如 Shibboleth 或 Active Directory 同盟服務) 之間建立信任，並讓組織中的使用者可以存取資源時 AWS，您可以使用 IAM 身分識別提供者。 AWS 在 IAM 信任政策中，IAM SAML 身分提供者將作為主體。

如需有關此案例的詳細資訊，請參閱 SAML 2.0 聯合身分。

您可以在或使用 AWS CLI、Windows PowerShell 專用工具 AWS Management Console 或 AWS API 呼叫中建立和管理 IAM 身分識別提供者。

在您建立 SAML 提供者之後，您必須建立一個或多個 IAM 角色。角色是沒有自己認證的身份（就像用戶一樣）。 AWS 但在此情況中，角色是以動態指派給聯合身分使用者，而該使用者由您組織的身分提供者 (IdP) 進行驗證。角色可允許您組織的 IdP 請求暫時性安全憑證以存取 AWS。指派給角色的原則會決定允許同盟使用者在中 AWS執行的動作。若要為 SAML 聯合身分建立角色，請參閱 針對第三方身分提供者建立角色 (聯合身分)。

最後，建立角色之後，您可以透過將 IdP 設定為您希望同盟使用者使用的相關資訊以 AWS 及角色來完成 SAML 信任。這是指在您的 IdP 和 AWS之間設定依賴方信任。若要設定依賴方信任，請參閱使用信賴方信任和新增宣告來設定您的 SAML 2.0 IdP。

主題

- 必要條件
- 建立和管理 IAM SAML 身分識別提供者 (主控台)
- 建立和管理 IAM SAML 身分識別提供者 ()AWS CLI
- 建立和管理 IAM SAML 身分識別提供者 (AWS API)

必要條件

建立 SAML 身分識別提供者之前，您必須先從 IdP 取得下列資訊。

- 從您的 IdP 取得 SAML 中繼資料文件。此文件包括發行者的名稱、到期資訊以及可用於驗證從 IdP 接收的 SAML 身分驗證回應 (宣告) 的金鑰。若要產生中繼資料文件，請使用外部 IdP 提供的身分識別管理軟體。

> **Important**
>
> 此中繼資料檔案包括發行者名稱、過期資訊以及可用於驗證從 IdP 接收的 SAML 身分驗證回應 (聲明) 的金鑰。中繼資料檔案必須以 UTF-8 格式編碼，並且不含位元組順序記號 (BOM)。若要移除 BOM，您可以使用文字編輯工具，例如 Notepad++，將檔案編碼為 UTF-8。
>
> 包含在 SAML 中繼資料文件中的 x.509 憑證必須使用大小至少有 1024 位元的金鑰。此外，x.509 憑證也必須沒有任何重複的擴充。您可以使用擴充，但這些擴充只能在憑證中出現一次。如果 x.509 憑證不符合任一條件，IdP 建立會失敗，並傳回「無法剖析中繼資料」錯誤。
>
> 如 SAML V2.0 中繼資料互通性設定檔版本 1.0 所定義，IAM 既不會評估中繼資料文件的 X.509 憑證是否過期，也不會對其採取動作。

如需有關如何設定許多可供使用 IdPs 的指示 AWS，包括如何產生所需的 SAML 中繼資料文件，請參閱將第三方 SAML 解決方案提供者與 AWS。

如需有關 SAML 聯盟的說明，請參閱 SAML 聯盟疑難排解。

建立和管理 IAM SAML 身分識別提供者 (主控台)

您可以使用建 AWS Management Console 立、更新和刪除 IAM SAML 身分識別提供者。如需有關 SAML 聯盟的說明，請參閱 SAML 聯盟疑難排解。

建立 IAM SAML 身分提供者 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，請選擇 Identity providers (身分提供者)，然後選擇 Add provider (新增提供者)。
3. 在 Configure provider (設定提供者) 中，選擇 SAML。
4. 輸入身分提供者的名稱。
5. 對於 Metadata document (中繼資料文件)，選擇 Choose file (選擇檔案)，並指定您在 the section called “必要條件” 下載的 SAML 中繼資料文件。
6. (選擇性) 對於「新增標籤」，您可以新增金鑰與值配對，以協助您識別和組織您的. IdPs 您也可以使用標籤來控制對 AWS 資源的存取。若要進一步了解如何標記 SAML 身分提供者，請參閱標記 IAM SAML 身分提供者。

選擇 Add tag (新增標籤)。輸入每個標籤鍵值組的值。

7. 請確認您提供的資訊。完成後，請選擇 Add provider (新增提供者)。
8. 將 IAM 角色指派給您的身分提供者，以授與身分識別提供者所管理的外部使用者身分識別，以存取帳戶中的 AWS 資源。若要深入了解如何建立聯合身分角色，請參閱 針對第三方身分提供者建立角色 (聯合身分)。

> **Note**
>
> 角色信任政策中使用的 SAML IDP 必須與角色所在的帳戶相同。

刪除 SAML 提供者 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，請選擇 Identity providers (身分提供者)。
3. 在您要刪除的身分提供者旁邊，選取選項按鈕。
4. 選擇 Delete (刪除)。新的視窗將開啟。
5. 在欄位中輸入單字 `delete`，確認您要刪除提供者。再選擇 Delete (刪除)。

建立和管理 IAM SAML 身分識別提供者 ()AWS CLI

您可以使用建 AWS CLI 立、更新和刪除 SAML 提供者。如需有關 SAML 聯盟的說明，請參閱 SAML 聯盟疑難排解。

建立 IAM 身分提供者並上傳中繼資料文件 (AWS CLI)

- 執行此命令：`aws iam create-saml-provider`

若要更新 IAM SAML 身分識別提供者 ()AWS CLI

- 執行此命令：`aws iam update-saml-provider`

標記現有 IAM 身分提供者 (AWS CLI)

- 執行此命令：`aws iam tag-saml-provider`

列出現有 IAM 身分提供者的標籤 (AWS CLI)

- 執行此命令：`aws iam list-saml-provider-tags`

移除現有 IAM 身分提供者的標籤 (AWS CLI)

- 執行此命令：`aws iam untag-saml-provider`

刪除 IAM SAML 身分提供者 (AWS CLI)

1. (選用) 若要列出所有提供者的資訊，例如 ARN、建立日期和過期日期，請執行下列命令：
   - `aws iam list-saml-providers`
2. (選擇性) 若要取得特定提供者的相關資訊，例如 ARN、建立日期、到期日、加密設定和私密金鑰資訊，請執行下列命令：
   - `aws iam get-saml-provider`
3. 若要刪除 IAM 身分提供者，請執行下列命令：
   - `aws iam delete-saml-provider`

建立和管理 IAM SAML 身分識別提供者 (AWS API)

您可以使用 AWS API 來建立、更新和刪除 SAML 提供者。如需有關 SAML 聯盟的說明，請參閱 SAML 聯盟疑難排解。

若要建立 IAM 身分提供者並上傳中繼資料文件 (AWS API)

- 呼叫此操作：`CreateSAMLProvider`

若要更新 IAM SAML 身分識別提供者 (AWS API)

- 呼叫此操作：`UpdateSAMLProvider`

標記現有的 IAM 身分提供者 (AWS API)

- 呼叫此操作：`TagSAMLProvider`

列出現有 IAM 身分提供者 (AWS API) 的標籤

- 呼叫此操作：ListSAMLProviderTags

移除現有 IAM 身分提供者 (AWS API) 上的標籤

- 呼叫此操作：UntagSAMLProvider

若要刪除身分識別提供者 (AWS API)

1. (選擇性) 若要列出所有人的資訊 IdPs，例如 ARN、建立日期和到期日，請呼叫下列作業：
   - ListSAMLProviders
2. (選擇性) 若要取得特定提供者的相關資訊，例如 ARN、建立日期、到期日、加密設定和私密金鑰資訊，請呼叫下列作業：
   - GetSAMLProvider
3. 若要刪除 IdP，請呼叫下列操作：
   - DeleteSAMLProvider

使用信賴方信任和新增宣告來設定您的 SAML 2.0 IdP

當您為 SAML 存取權建立 IAM 身分提供者和角色時，基本上是在告知 AWS 有關允許身分提供者 (IdP) 和其使用者執行哪些動作。您的下一步是告訴 IdP AWS 作為服務提供商。這稱為在 IdP 和 之間新增依賴方信任 AWS。新增依賴方信任的過程，取決於您使用哪種 IdP。如需詳細資訊，請參閱身分管理軟體的文件。

許多可 IdPs 讓您指定一個 URL，IdP 可從中讀取包含信賴憑證者資訊和憑證的 XML 文件。對於 AWS，使用`https://region-code.signin.aws.amazon.com/static/saml-metadata.xml`或`https://signin.aws.amazon.com/static/saml-metadata.xml`。對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。

如果您無法直接指定 URL，請從先前的 URL 下載 XML 文件，然後匯入到您的 IdP 軟體。

您還需要在 IdP 中建立指定 AWS 為信賴憑證者的適當宣告規則。IdP 將 SAML 回應傳送至 AWS 端點時，它會包含包含一或多個宣告的 S AML 宣告。宣告是和使用者及其群組相關的資訊。宣告規則將該資訊對應到 SAML 屬性。這可讓您確保來自 IdP 的 SAML 身份驗證回應包含 IAM 政策中 AWS 用來檢查聯合身分使用者許可的必要屬性。如需詳細資訊，請參閱下列主題：

- 允許 SAML 聯合存取資源的角色概觀 AWS. 此主題使用 SAML 特定索引鍵討論 IAM 政策，以及如何使用這些政策來限制 SAML 聯合身分使用者的許可。
- 設定驗證回應的 SAML 宣告. 此主題討論如何設定 SAML 宣告，其包含有關使用者的資訊。聲明已捆綁在 SAML 聲明中，並且包含在傳送到 AWS的 SAML 回應中。您必須確保 AWS 原則所需的資訊 AWS 以可辨識和使用的形式包含在 SAML 宣告中。
- 將第三方 SAML 解決方案提供者與 AWS。 本主題提供協力廠商組織提供的文件連結，說明如何將身分識別解決方案與整合 AWS。

> **Note**
>
> 若要改善聯合彈性，建議您將 IdP 和 AWS 聯合設定為支援多個 SAML 登入端點。如需詳細資訊，請參閱 AWS 安全性部落格文章如何使用地區性 SAML 端點進行容錯移轉。

將第三方 SAML 解決方案提供者與 AWS

> **Note**
>
> 我們建議您要求您的人類使用者在存取時使用臨時登入資料 AWS。你有沒有考慮過使用 AWS IAM Identity Center？ 您可以使用 IAM 身分中心集中管理多個存取權限， AWS 帳戶 並從單一位置為使用者提供所有指派帳戶的 MFA 保護的單一登入存取權。使用 IAM Identity Center，您可以在 IAM Identity Center 中建立和管理使用者身分，或輕鬆連線至您現有的 SAML 2.0 相容身分提供者。如需詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

下列連結可協助您設定協力廠商 SAML 2.0 身分識別提供者 (IdP) 解決方案以與 AWS 同盟搭配使用。

> **Tip**
>
> AWS Support 工程師可協助擁有商業和企業支援計劃的客戶完成某些涉及協力廠商軟體的整合工作。如需所支援平台和應用程式的最新清單，請參閱 AWS Support 常見問答集中的支援哪些第三方軟體？

解決方案	其他資訊
Auth0	與 Amazon Web Services 整合 — Auth0 文件網站上的這個頁面包含說明如何設定單一登入 (SSO) 的資源連結， AWS Management Console 並包含 JavaScript範例。您可以設定 Auth0 來傳遞工作階段標籤。如需詳細資訊，請參閱 Auth0 宣布與 IAM 工作階段標籤 AWS 合作。
Microsoft Entra	教學課程：Microsoft Entra SSO 與 AWS 單一帳戶存取整合 — 此 Microsoft 網站上的教學課程說明如何使用 SAML 聯盟將 Microsoft Entra (先前稱為 Azure AD) 設定為身分識別提供者 (IdP)。
Centrify	設定中心化和使用 SAML 以進行 SSO AWS — 中心化網站上的此頁面說明如何設定中心化以使用 SAML 進行 SSO。 AWS
CyberArk	進行設CyberArk定，為透過 SAML 單一登入 (SSO AWS) 從使用者入口網站登入的使用 CyberArk 者提供 Amazon 網路服務 () 存取權。
ForgeRock	ForgeRock 身分識別平台與 AWS. 您可以設定 ForgeRock 為傳遞工作階段標籤。如需詳細資訊，請參閱 Amazon Web Services 的屬性型存取控制。
Google Workspace	Amazon Web Services 雲端應用程式 — Google 工作區管理員說明網站上的這篇文章說明如何以服務供應商的身分將 Google 工作區設定 AWS 為 SAML 2.0 IdP。
IBM	您可以設定 IBM 傳遞工作階段標籤。如需詳細資訊，請參閱 IBM 雲端身分識別 IDaaS，其中一個支援 AWS 工作階段標記的第一個。
JumpCloud	透過 Amazon 針對單一登入 (SSO) 透過 IAM 角色授予存取權 AWS — JumpCloud 網站上的這篇文章說明如何根據的 IAM 角色設定和啟用 SSO AWS。

解決方案	其他資訊
Matrix42	MyWorkspace 入門指南 — 本指南說明如何將 AWS 身分識別服務與 Matri MyWorkspace x42 整合。
Microsoft Active Directory Federation Services (AD FS)	欄位備註：整合 Active Directory 同盟服務與 AWS IAM Identity Center — AWS 架構部落格上的這篇文章說明 AD FS 和 AWS IAM Identity Center (IAM 身分中心) 之間的驗證流程。IAM Identity Center 使用 SAML 2.0 支援聯合身分，可與 AD FS 解決方案整合。使用者可透過其公司憑證登入 IAM Identity Center 入口網站，減少在 IAM Identity Center 上維護單獨憑證的管理開銷。您也可以設定 AD FS 來傳遞工作階段標籤。如需詳細資訊，請參閱搭配 AD FS 使用屬性型存取控制，以簡化 IAM 許可管理。
miniOrange	SSO for AWS — miniOrange 網站上的此頁面說明如何 AWS 為企業建立安全存取，以及對 AWS 應用程式存取的完全控制權。
Okta	使用 Okta 整合 Amazon Web Services 命令列介面 – 從 Okta 支援網站的這個頁面，您可以了解如何設定 Okta 以與 AWS一起使用。您可以設定 Okta 傳遞工作階段標籤。如需詳細資訊，請參閱 Okta 和 AWS 合作夥伴以簡化透過工作階段標籤存取。
Okta	AWS 聯合帳戶 — Okta 網站上的這個區段說明如何設定和啟用 IAM 身分中心。 AWS
OneLogin	在OneLogin知識庫中，搜**SAML AWS**尋文章清單，說明如何在單一角色 OneLogin 和多角色案 AWS 例之間設定 IAM 身分中心功能。您可以設定 OneLogin 為傳遞工作階段標籤。如需詳細資訊，請參閱OneLogin 工作階段標籤：以屬性為 AWS 基礎的資源存取控制。

解決方案	其他資訊
Ping 身分	PingFederate AWS 連接器 — 檢視有關連 PingFederate AWS 接器的詳細資料，這是一種可輕鬆設定單一登入 (SSO) 和佈建連線的快速連線範本。閱讀文件並下載最新的 PingFederate AWS 連接器以進行整合 AWS。您可以設定 Ping 身分以傳遞工作階段標籤。如需詳細資訊，請參閱宣佈 Ping Identity 支援 AWS中的屬性型存取控制。
RadiantLogic	輻射邏輯技術合作夥伴 — 輻射邏輯的 RadiantOne聯合身份識別服務與整合， AWS 為基於 SAML 的 SSO 提供身分中樞。
RSA	Amazon Web Services-RSA 就緒實作指南提供整合 AWS 和 RSA 的指導。如需有關 SAML 組態的詳細資訊，請參閱 Amazon Web Services-SAML 我的頁面 SSO 組態-RSA 就緒實作指南。
Salesforce.com	如何將 SSO 從 Salesforce 設定為 AWS— Sales force.com 開發人員網站上的這篇操作說明文章說明如何在 Salesforce 中設定身分識別提供者 (IdP)，並設定 AWS 為服務提供者。
SecureAuth	AWS - SecureAuth SAML SSO — SecureAuth 網站上的這篇文章說明如何為設備設定 SAML 整 AWS 合 SecureAuth 。
Shibboleth	如何將 Shibboleth 用於 SSO 至 AWS Management Console— AWS 安全性部落格上的這個項目提供了如何設定 Shibboleth 並將其設定為的身分識別提供者的 step-by-step 教學課程。 AWS您可以設定 Shibboleth 傳遞工作階段標籤。

如需詳細資訊，請參閱 AWS 網站上的 IAM 合作夥伴頁面。

設定驗證回應的 SAML 宣告

驗證組織中的使用者身分後，外部身分識別提供者 (IdP) 會將驗證回應傳送至位於的 AWS SAML 端點。`https://region-code.signin.aws.amazon.com/saml`對於可能的 *region-code* 替換清

單，請參閱 AWS 登入端點中的 Region (區域) 欄位。此回應是一個包含 SAML 權杖的 POST 請求，該權杖遵循適用於 SAML 2.0 的 HTTP POST 繫結標準，其中包含下列元素或宣告。您可在與 SAML 相容的 IdP 中配置這些聲明。請參考 IdP 文件，以了解有關如何輸入這些聲明的說明。

當 IdP 將包含聲明的響應發送到時 AWS，許多傳入的聲明映射到 AWS 上下文鍵。可以在 IAM 政策中使用 Condition 元素來檢查這些內容索引鍵。可用映射的清單將根據 將 SAML 屬性對應至 AWS 信任原則內容索引鍵 中所示。

Subject 與 NameID

下列的摘錄顯示了範例。用您自己的值替代標記值。同時包含 SubjectConfirmation 和 SubjectConfirmationData 屬性的 NotOnOrAfter 元素必須恰好具有一個 Recipient 元素。這些屬性包括必須與 AWS 端點相符的值https://*region-code*.signin.aws.amazon.com/saml。對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。對於 AWS 值，您也可以使用https://signin.aws.amazon.com/static/saml，如下列範例所示。

NameID 元素的值可以是持久的、暫時的，也可以由 IdP 解決方案提供的完整格式 URI 組成。持久的值表示 NameID 中的值對於工作階段之間的使用者是相同的。如果值為暫時的，則使用者在每個工作階段中擁有不同的 NameID 值。單一登入互動支援下列類型的識別碼：

- urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
- urn:oasis:names:tc:SAML:2.0:nameid-format:transient
- urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
- urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified
- urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName
- urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos
- urn:oasis:names:tc:SAML:2.0:nameid-format:entity

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">_cbb88bf52c2510eabe00c1642d4643f41430fe25e3</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData NotOnOrAfter="2013-11-05T02:06:42.876Z"
 Recipient="https://signin.aws.amazon.com/saml"/>
  </SubjectConfirmation>
</Subject>
```

> **Important**
>
> saml:aud 內容索引鍵來自 SAML recipient 屬性，因其為等同於 OIDC 觀眾欄位 (例如 accounts.google.com:aud) 的 SAML。

PrincipalTag SAML 屬性

(選用) 您可以使用 Attribute 元素，並將 Name 屬性設為 https://aws.amazon.com/SAML/Attributes/PrincipalTag:*{TagKey}*。此元素可讓您在 SAML 聲明中將屬性做為工作階段標籤傳遞。如需有關工作階段標籤的詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

若要將屬性做為工作階段標籤傳遞，請包含指定標籤值的 AttributeValue 元素。例如，若要傳遞標籤金鑰/值對 Project = Marketing 和 CostCenter = 12345，請使用以下屬性。為每個標籤包含個別的 Attribute 元素。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Marketing</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
```

如要將以上標籤設為可轉移，請在其中包含另一個 Attribute 元素，並將其 Name 屬性設為 https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys。這是一個選用的多重值屬性，會將您的工作階段標籤設為可轉移。可轉移標籤會在您於 AWS中使用 SAML 聲明擔任另一個角色時保存。這就是所謂的角色鏈接。例如，若要將 Principal 和 CostCenter 標籤設為可轉移，請使用以下屬性來指定金鑰。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>CostCenter</AttributeValue>
</Attribute>
```

Role SAML 屬性

您可以使用 Attribute 元素，將 Name 屬性設為 https://aws.amazon.com/SAML/Attributes/Role。此元素包含一或多個 AttributeValue 元素，這些元素可列出將由您的 IdP 對應之使用者的 IAM 身分提供者和角色。IAM 角色和 IAM 身分識別提供者會以逗號分隔的 ARN 組指定，格式與傳送至 AssumeRoleWith SAML 的RoleArn和PrincipalArn參數相同。此元素必須包含

至少一個角色提供者對 (AttributeValue 元素)，並且可以包含多個角色提供者對。如果該元素包含多個角色提供者組，則使用者將需要選擇其使用 WebSSO 登入 AWS Management Console時要擔任的角色。

> **Important**
>
> Name 標籤中 Attribute 屬性的值區分大小寫。必須精準設定為 https://aws.amazon.com/SAML/Attributes/Role。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/Role">
  <AttributeValue>arn:aws:iam::account-number:role/role-name1,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name2,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name3,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
</Attribute>
```

RoleSessionName SAML 屬性

您可以使用 Attribute 元素，將 Name 屬性設為 https://aws.amazon.com/SAML/Attributes/RoleSessionName。此元素包含一個為暫時憑證提供識別符的 AttributeValue 元素，而這些憑證是在擔任角色時所發出。您可以使用此功能將臨時憑證與正在使用應用程式的使用者建立關聯。此元素用於在中顯示使用者資訊 AWS Management Console。AttributeValue 元素中的值長度必須介於 2 到 64 個字元之間，只能包含字母數位字元、底線和以下字元：. , + = @ - (連字號)。其中不可含有空格。該值通常是使用者 ID (johndoe) 或電子郵件地址 (johndoe@example.com)。該值不應包含空格，如使用者的顯示名稱 (John Doe)。

> **Important**
>
> Name 標籤中 Attribute 屬性的值區分大小寫。必須精準設定為 https://aws.amazon.com/SAML/Attributes/RoleSessionName。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName">
  <AttributeValue>user-id-name</AttributeValue>
</Attribute>
```

`SessionDuration` SAML 屬性

(選用) 您可以使用 `Attribute` 元素，並將 `Name` 屬性設為 `https://aws.amazon.com/SAML/Attributes/SessionDuration"`。此元素包含一個`AttributeValue`元素，指定使用者在必須要求新的臨時認證 AWS Management Console 之前可以存取的時間長度。該值是一個表示工作階段秒數的整數。該值的範圍是 900 秒 (15 分鐘) 到 43200 秒 (12 小時)。如果此屬性不存在，則憑證有效時間為 1 小時 (`DurationSeconds` API 的 `AssumeRoleWithSAML` 參數的預設值)。

若要使用此屬性，您必須將 SAML 提供者設定為 AWS Management Console 透過主控台登入 Web 端點提供單一登入存取權限。`https://`*`region-code`*`.signin.aws.amazon.com/saml`對於可能的 *`region-code`* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。您可以選擇使用以下 URL：`https://signin.aws.amazon.com/static/saml`。請注意，此屬性只會將工作階段擴展到 AWS Management Console。它不能延長其他憑證的有效期。但是，如果其存在於 `AssumeRoleWithSAML` API 呼叫中，則可以用來縮短工作階段的持續時間。呼叫傳回的憑證預設存留期是 60 分鐘。

另請注意，如果同時定義 `SessionNotOnOrAfter` 屬性，則兩個屬性的較小值 (`SessionDuration` 或 `SessionNotOnOrAfter`) 將建立主控台工作階段的最長持續時間。

在啟用具有更長持續時間的主控台工作階段時，可能會導致憑證洩露的風險。為了幫助緩解這種風險，您可以在 IAM 主控台的 Role Summary (角色摘要) 頁面上選擇 Revoke Sessions (撤銷工作階段) 來立即停用所有角色的使用中的主控台工作階段。如需詳細資訊，請參閱 撤銷 IAM 角色暫時性安全憑證。

> **Important**
>
> Name 標籤中 `Attribute` 屬性的值區分大小寫。必須精準設定為 `https://aws.amazon.com/SAML/Attributes/SessionDuration`。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SessionDuration">
  <AttributeValue>1800</AttributeValue>
</Attribute>
```

`SourceIdentity` SAML 屬性

(選用) 您可以使用 `Attribute` 元素，並將 `Name` 屬性設為 `https://aws.amazon.com/SAML/Attributes/SourceIdentity`。此元素包含一個 `AttributeValue` 元素，提供使用 IAM 角色之人員或應用程式的識別碼。當您使用 SAML 工作階段在 AWS 稱為角色鏈結中擔任另一個角色時，來源識別的值仍然存在。針對在角色工作階段期間所採取的每個動作，來源身分的值會出現在請求中。設

定的值無法在角色工作階段期間變更。然後，系統管理員可以使用 AWS CloudTrail 記錄來監視和稽核來源身分識別資訊，以決定誰使用共用角色執行動作。

AttributeValue 元素中的值長度必須介於 2 到 64 個字元之間，只能包含字母數位字元、底線和以下字元：. , + = @ - (連字號)。其中不可含有空格。此值通常是與使用者相關聯的屬性，例如使用者 ID (johndoe) 或電子郵件地址 (johndoe@example.com)。該值不應包含空格，如使用者的顯示名稱 (John Doe)。如需有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

> **Important**
>
> 如果您的 SAML 聲明設定為使用 SourceIdentity 屬性，則您的信任政策也必須包含 sts:SetSourceIdentity 動作，否則擔任角色操作將失敗。如需有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

若要傳遞來源身分屬性，請包含指定來源身分值的 AttributeValue 元素。例如，若要傳遞來源身分 DiegoRamirez，請使用下列屬性。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
  <AttributeValue>DiegoRamirez</AttributeValue>
```

將 SAML 屬性對應至 AWS 信任原則內容索引鍵

本部分中的表列出了常用的 SAML 屬性以及它們在 AWS中映射到信任政策條件內容索引鍵的方式。您可以使用這些金鑰來控制角色的存取。若要執行此作業，請將金鑰與包含在 SAML 存取請求隨附聲明中的值進行比較。

> **Important**
>
> 這些索引鍵僅在 IAM 信任政策 (可決定誰能擔任角色的政策) 中可用，並且不適用於許可政策。

在 eduPerson 和 eduOrg 屬性工作表中，值是以字串或字串清單的形式輸入的。對於字串值，您可以使用 StringEquals 或 StringLike 條件測試 IAM 信任政策中的這些值。對於包含字串清單的值，您可以使用 ForAnyValue 與 ForAllValues 政策集運算子來測試信任政策中的這些值。

> **Note**
>
> 每個內容索引 AWS 鍵應該只包含一個宣告。如果包含多個聲明，將僅映射其中一個。

eduPerson 和 eduOrg 屬性

eduPerson 或 eduOrg 屬性 (**Name** 索引鍵)	映射到此 AWS 上下文鍵（**FriendlyName** 鍵）	類型
urn:oid:1.3.6.1.4.1.5923.1.1.1.1	eduPerson Affiliation	字串清單
urn:oid:1.3.6.1.4.1.5923.1.1.1.2	eduPersonNickname	字串清單
urn:oid:1.3.6.1.4.1.5923.1.1.1.3	eduPersonOrgDN	字串
urn:oid:1.3.6.1.4.1.5923.1.1.1.4	eduPerson OrgUnitDN	字串清單
urn:oid:1.3.6.1.4.1.5923.1.1.1.5	eduPerson PrimaryAf filiation	字串
urn:oid:1.3.6.1.4.1.5923.1.1.1.6	eduPerson PrincipalName	字串
urn:oid:1.3.6.1.4.1.5923.1.1.1.7	eduPerson Entitlement	字串清單
urn:oid:1.3.6.1.4.1.5923.1.1.1.8	eduPerson PrimaryOrgUnitDN	字串
urn:oid:1.3.6.1.4.1.5923.1.1.1.9	eduPerson ScopedAff iliation	字串清單
urn:oid:1.3.6.1.4.1.5923.1. 1.1.10	eduPerson TargetedID	字串清單

eduPerson 或 eduOrg 屬性 (**Name** 索引鍵)	映射到此 AWS 上下文鍵（**FriendlyName** 鍵）	類型
urn:oid:1.3.6.1.4.1.5923.1.1.1.11	eduPersonAssurance	字串清單
urn:oid:1.3.6.1.4.1.5923.1.2.1.2	eduOrgHomePageURI	字串清單
urn:oid:1.3.6.1.4.1.5923.1.2.1.3	eduOrgIdentityAuthNPolicyURI	字串清單
urn:oid:1.3.6.1.4.1.5923.1.2.1.4	eduOrgLegalName	字串清單
urn:oid:1.3.6.1.4.1.5923.1.2.1.5	eduOrgSuperiorURI	字串清單
urn:oid:1.3.6.1.4.1.5923.1.2.1.6	eduOrgWhitePagesURI	字串清單
urn:oid:2.5.4.3	cn	字串清單

Active Directory 屬性

AD 屬性	映射到此 AWS 上下文鍵	Type
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	name	字串
http://schemas.xmlsoap.org/claims/CommonName	commonName	字串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	givenName	字串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	surname	字串

AD 屬性	映射到此 AWS 上下文鍵	Type
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	mail	字串
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	uid	字串

X.500 屬性

X.500 屬性	映射到此 AWS 上下文鍵	Type
2.5.4.3	commonName	字串
2.5.4.4	surname	字串
2.4.5.42	givenName	字串
2.5.4.45	x500UniqueIdentifier	字串
0.9.2342.19200300100.1.1	uid	字串
0.9.2342.19200300100.1.3	mail	字串
0.9.2342.19200300.100.1.45	organizationStatus	字串

啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console

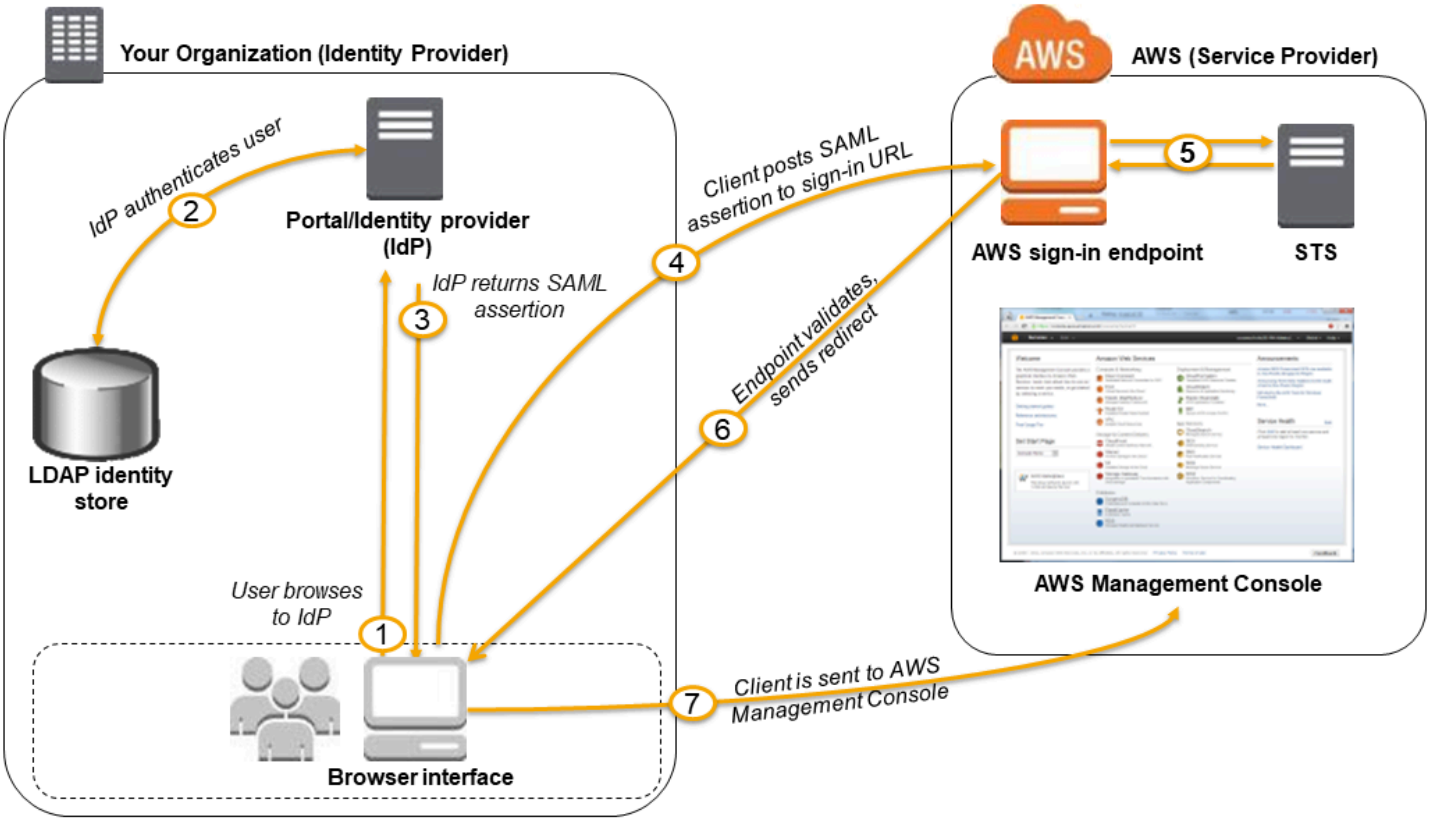
您可以使用角色來設定與 SAML 2.0 相容的身分識別提供者 (IdP)，並允許您的同盟 AWS 使用者存取。 AWS Management Console該角色為使用者授予了在主控台中執行任務的許可。如果您希望為 SAML 聯合身分使用者提供存取 AWS的其他方式，請參閱以下主題之一：

- AWS CLI: 切換到 IAM 角色 (AWS CLI)
- 視窗工具 PowerShell:切換至 IAM 角色 (適用於視窗的工具 PowerShell)
- AWS API：切換到身分與存取權管理角色 (AWS API)

概觀

下圖說明了啟用了 SAML 的單一登入的流程。

> **Note**
>
> SAML 的這種特定用法與中所示的更一般的用法不同，SAML 2.0 聯合身分因為此工作流程會 AWS Management Console 代表使用者開啟。這需要使用 AWS 登入端點，而不是直接呼叫 `AssumeRoleWithSAML` API。該端點將為使用者呼叫 API 並傳回將使用者的瀏覽器自動重新引導到 AWS Management Console的 URL。



此圖說明了下列步驟：

1. 使用者瀏覽到您的組織的入口網站，並選擇前往 AWS Management Console的選項。在您的組織中，入口網站通常是 IdP 的一項功能，可處理組織與 AWS組織之間的信任交換。例如，在 Active Directory Federation Services 中，入口網站的 URL 是：`https://ADFSServiceName/adfs/ls/IdpInitiatedSignOn.aspx`
2. 該入口網站可驗證您的組織使用者的身分。

3. 入口網站會產生一個 SAML 身分驗證回應，其中包括辨識使用者身分的聲明以及使用者的相關屬性。您也可以配置 IdP 以包含一個名為 `SessionDuration` 的 SAML 聲明屬性，該屬性指定主控台工作階段的有效時間長度。您也可以設定 IdP，將屬性做為工作階段標籤傳遞。該入口網站將此回應傳送到用戶端瀏覽器。
4. 用戶端瀏覽器會重新導向至 AWS 單一登入端點，並張貼 SAML 宣告。
5. 端點代表使用者請求臨時安全性憑證，並建立一個使用這些憑證的主控台登入 URL。
6. AWS 將登錄 URL 作為重定向發送回客戶端。
7. 該用戶端瀏覽器將重新引導到 AWS Management Console。如果 SAML 身分驗證回應包含映射到多個 IAM 角色的屬性，則將首先提示使用者選取要用於存取主控台的角色。

從使用者的角度來看，程序會以透明的方式進行：使用者從您組織的內部入口網站開始 AWS Management Console，最後到達，而無需提供任何 AWS 認證。

有關如何配置此行為的概述以及指向詳細步驟的連結，請參閱以下章節。

將您的網路設定為 SAML 提供者 AWS

在您的組織的網路內部，將您的身分存放區 (如 Windows Active Directory) 設定為與 Windows Active Directory 聯合服務、Shibboleth 等以 SAML 為基礎的身分提供者 (IdP) 一起使用。使用 IdP 產生一個中繼資料文件，該文件將您的組織描述為身分提供者 (IdP) 並包含身分驗證索引鍵。您也可以設定組織的入口網站，以便使用 SAML 宣告 AWS Management Console 將對的使用者要求路由到 AWS SAML 端點進行驗證。配置您的 IdP 來產生 metadata.xml 文件之方法將根據您的 IdP 而定。請參閱您的 IdP 文件以獲得說明，或參閱 將第三方 SAML 解決方案提供者與 AWS 以取得包含許多支援的 SAML 提供者之 Web 文件連結。

在 IAM 中建立 SAML 提供者

接下來，您登入 AWS Management Console 並移至 IAM 主控台。您將在此建立一個新的 SAML 提供者，該提供者是 IAM 中包含您組織的身分提供者 (IdP) 的相關資訊的實體。在此過程中，您可以上傳在前一章節中由組織中的 IdP 軟體所產生的中繼資料文件。如需詳細資訊，請參閱 在 IAM 中建立 SAML 身分識別提供者。

在中設定同盟使 AWS 用者的權限

下一個步驟是建立 IAM 角色，以便建立 IAM 和您組織 IdP 之間的信任關係。基於聯合目的，這個角色必須將您的 IdP 識別為主體 (信任的實體)。角色也會定義組織 IdP 驗證的使用者可在中 AWS執行哪些動作。您可使用 IAM 主控台來建立此角色。當您建立信任政策，指出可以擔任角色的人員時，您可

以指定您先前在 IAM 中建立的 SAML 提供者。您也可以指定一或多個 SAML 屬性，讓使用者在符合這些屬性的情況下，才能允許擔任角色。例如，您可以指定只允許其 SAML eduPersonOrgDN 值為 ExampleOrg 的使用者登入。角色精靈會自動新增一個測試 saml:aud 屬性的條件，確保僅以登入 AWS Management Console為目的來擔任該角色。該角色的信任政策可能如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::account-id:saml-provider/
ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {"StringEquals": {
      "saml:edupersonorgdn": "ExampleOrg",
      "saml:aud": "https://signin.aws.amazon.com/saml"
    }}
  }]
}
```

> **Note**
>
> 角色信任政策中使用的 SAML IDP 必須與角色所在的帳戶相同。

您可以包括位於 https://*region-code*.signin.aws.amazon.com/static/saml-metadata.xml 的 saml:aud 屬性的區域端點。對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。

對於該角色中的許可政策，您可以像任何角色、使用者或群組一樣指定許可。例如，如果允許您組織中的使用者管理 Amazon EC2 執行個體，您可以在許可政策中明確允許 Amazon EC2 動作。您可以指派受管政策 (如 Amazon EC2 完整存取受管政策) 來執行此操作。

有關建立用於 SAML IdP 的角色的詳細資訊，請參閱 為 SAML 2.0 聯合 (主控台) 建立角色。

完成組態並建立 SAML 聲明

安裝位https://*region-code*.signin.aws.amazon.com/static/saml-metadata.xml於或的saml-metadata.xml檔案，以通知您的 SAML IdP AWS 是您的服務提供者。https://signin.aws.amazon.com/static/saml-metadata.xml對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。

安裝該檔案的方式取決於您的 IdP。部分提供者為您提供了輸入該 URL 的選項，此時，IdP 將為您取得並安裝該檔案。另一些提供者則要求您從該 URL 處下載檔案，然後將其做為本機檔案提供。請參閱您的 IdP 文件以獲得詳細資訊，或參閱 將第三方 SAML 解決方案提供者與 AWS 以取得包含許多支援的 SAML 提供者之 Web 文件連結。

您也可設定一條資訊，說明您希望 IdP 在身分驗證回應期間將此資訊做為 SAML 屬性傳遞到 AWS 。這些資訊大部分都會顯示在條 AWS 件內容索引鍵中，您可以在原則中評估這些索引鍵。這些條件索引鍵會確保只有經過授權且位於正確內容中的使用者，才能獲得授予存取您 AWS 資源的許可。您可以指定限制可使用主控台時間的時段。您也可以指定使用者在重新整理其憑證前，可存取主控台的時間上限 (最多 12 小時)。如需詳細資訊，請參閱 設定驗證回應的 SAML 宣告。

IAM 中的暫時安全憑證

您可以使用 AWS Security Token Service (AWS STS) 建立並為受信任的使用者提供可控制資源存取權的臨時安全登入 AWS 資料。暫時性安全憑證的作用幾乎與長期存取金鑰憑證完全相同，而其差異如下：

- 暫時安全憑證是「短期」的，如其名稱所暗示。其可設定為在任何地方持續幾分鐘到幾小時不等。憑據過期後， AWS 不再識別它們或允許從它們發出的 API 請求中進行任何類型的訪問。
- 暫時性安全憑證不會與使用者一起儲存，但會動態產生並在請求時提供給使用者。當暫時性安全憑證到期時 (或者即使在此之前)，使用者可以請求新的憑證，只要使用者的請求仍具有可這麼做的許可。

因此，相較於長期憑證，暫時性憑證具有下列優點：

- 您不需要在應用程式中散佈或內嵌長期 AWS 安全性登入資料。
- 您可以為使用者提供對資 AWS 源的存取權，而不必為使用者定義 AWS 身分。臨時認證是角色和身份聯合的基礎。
- 臨時安全憑證的存留期有限，因此當不再需要時，您不需要更新它們或明確予以撤銷。暫時性安全憑證到期之後，就無法重複使用。您可以指定憑證的有效期，達到最長限制。

AWS STS 和 AWS 地區

AWS STS所產生暫時性安全憑證。預設情況下， AWS STS 是具有單一端點位於的全域服務`https://sts.amazonaws.com`。不過，您也可以選擇對任何其他支援區域中的端點進行 AWS STS API 呼叫。這可透過從地理位置較靠近您的區域的伺服器傳送請求，來降低延遲發生機率 (伺服器延遲)。無論您的憑證來自哪個區域，都能全域使用。如需詳細資訊，請參閱 AWS STS 在一個管理 AWS 區域。

暫時性憑證的常見案例

暫時性憑證在涉及聯合身分、委派、跨帳戶存取和 IAM 角色的案例中非常有用。

聯合身分

您可以在外部系統中管理您的使用者身分識別， AWS 並授與從這些系統登入的使用者以執行 AWS 工作和存取您的 AWS 資源的存取權。IAM 支援兩種類型的聯合身分。在這兩種情況下，身份都存儲在 AWS. 差別是外部系統所在地點是您的資料中心或外部第三方的 Web 上。如需有關外部身分提供者的詳細資訊，請參閱 身分提供者與聯合。

- SAML 聯盟 — 您可以驗證組織網路中的使用者，然後為這些使用者提供存取權， AWS 而無需為他們建立新的 AWS 身分，也可以要求他們使用不同的登入認證登入。這稱為暫時存取的單一登入方法。 AWS STS 支持開放標準，如安全斷言標記語言（SAML）2.0，您可以使用 Microsoft AD FS 來利用您的 Microsoft 活動目錄。您也可以使用 SAML 2.0 來管理您自己的聯合使用者身分的解決方案。如需詳細資訊，請參閱 SAML 2.0 聯合身分。
  - 自訂同盟代理人 — 您可以使用組織的驗證系統來授與 AWS 資源的存取權。如需範例案例，請參閱啟用自訂身分識別代理存取主 AWS 控台。
  - 使用 SAML 2.0 的聯合 – 您可以使用您組織的身分驗證系統和 SAML，來授予 AWS 資源的存取。如需詳細資訊和範例案例，請參閱 SAML 2.0 聯合身分。
- OpenID Connect（OIDC）聯盟 — 您可以讓用戶使用知名的第三方身份提供商登錄，例如使用亞馬遜，Facebook，Google 或任何與您的移動或 Web 應用程序的 OIDC 2.0 兼容提供商進行登錄，您無需創建自定義登錄代碼或管理自己的用戶身份。使用 OIDC 聯盟可協助您 AWS 帳戶 確保安全，因為您不需要將長期安全登入資料 (例如 IAM 使用者存取金鑰) 與應用程式散發。如需詳細資訊，請參閱 OIDC 聯盟。

  AWS STS OIDC 聯盟支援使用亞馬遜、臉書、谷歌和任何與 OpenID Connect (OIDC) 相容的身分供應商登入。

  > **Note**
  >
  > 對於行動應用程式，我們建議您使用 Amazon Cognito。您可以將此服務與用於行動開發的 AWS SDK 搭配使用，為使用者建立唯一身分，並對其進行驗證，以便安全存取您的 AWS 資源。Amazon Cognito 支援與相同的身分供應商 AWS STS，並且還支援未驗證 (訪客) 存取，並可讓您在使用者登入時遷移使用者資料。Amazon Cognito 還提供 API 操作以同步使用者資料，如此可在使用者於不同裝置之間移動時進行保留。如需詳細資訊，請參閱 Amplify 文件 中的 Amplify 身分驗證。

跨帳戶存取的角色

許多組織維護多個 AWS 帳戶。您可以使用角色和跨帳戶存取，來定義一個帳戶中的使用者身分，並使用那些身分來存取屬於您組織的其他帳戶中的 AWS 資源。這就是所謂暫時存取的「委派」方法。如需有關建立跨帳戶角色的詳細資訊，請參閱 建立角色以委派許可給 IAM 使用者。若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。

Amazon EC2 的角色

如果您在 Amazon EC2 執行個體上執行應用程式，並且那些應用程式需要存取 AWS 資源，則您可以在您啟動它們時提供暫時安全憑證。這些暫時性安全憑證可供在執行個體上執行的所有應用程式使用，因此您不需要在執行個體上儲存任何長期憑證。如需詳細資訊，請參閱 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可。

其他 AWS 服務

您可以使用臨時安全登入資料來存取大多數 AWS 服務。如需接受暫時性安全性憑證的服務的清單，請參閱AWS 與 IAM 搭配使用的服務。

請求暫時性安全憑證

要請求臨時安全登入資料，您可以在 AWS API 中使用 AWS Security Token Service (AWS STS) 操作。這些作業包括建立並為受信任的使用者提供可控制資源存取權的臨時安全登入 AWS 資料的作業。如需有關的更多資訊 AWS STS，請參閱IAM 中的暫時安全憑證。若要了解可用來在擔任角色時請求暫時性安全憑證的不同方法，請參閱使用 IAM 角色。

若要呼叫 API 操作，您可以使用其中一個 AWS 軟體開發套件。適用於多種程式設計語言和環境的軟體開發套件，包括 Java、.NET、Python、Ruby、Android 和 iOS。開發套件會負責的工作諸如以密碼演算法簽署請求，必要時重試請求，以及處理錯誤回應。您也可以使用 AWS STS 查詢 API，如 AWS Security Token Service API 參考資料中所述。最後，兩個指令行工具支援這些 AWS STS 指令：AWS Command Line Interface、和 AWS Tools for Windows PowerShell.

AWS STS API 操作創建一個包含臨時安全憑據的新會話，其中包括訪問 key pair 和會話令牌。存取金鑰對包含存取金鑰 ID 和秘密金鑰。使用者 (或使用者執行的應用程式) 可以使用這些憑證來存取您的資源。您可以使用 AWS STS API 操作以程式設計方式建立角色工作階段並傳遞工作階段原則和工作階段 所產生工作階段許可會是角色的以身分為基礎的政策和工作階段政策的交集。如需有關工作階段政策的詳細資訊，請參閱 工作階段政策。如需有關工作階段標籤的詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

> **Note**
>
> AWS STS API 操作返回的會話令牌的大小不固定。我們強烈建議您不對大小上限做出任何假設。一般權杖大小小於 4096 個位元組，但有可能會不同。

AWS STS 與 AWS 區域搭配使用

您可以將 AWS STS API 呼叫傳送至全域端點或其中一個區域端點。如果您選擇比較靠近您的端點，您可以減少延遲並改善您的 API 呼叫的效能。如果可以不再與原始端點通訊，您也可以選擇將您的呼叫導向至其他區域性端點。如果您使用的是各種 AWS SDK 之一，請在進行 API 呼叫之前使用該 SDK 方法來指定區域。如果您手動建構 HTTP API 請求，則必須您自己將請求直接導向到正確的端點。如需詳細資訊，請參閱區域與端點的AWS STS 一節和 AWS STS 在一個管理 AWS 區域。

以下是您可用來取得暫時登入資料的 API 作業，以便在您的 AWS 環境和應用程式中使用。

— AssumeRole 跨帳戶委派和透過自訂身分經紀人的聯合

`AssumeRole`API 操作對於允許現有 IAM 使用者存取他們尚未擁有存取權限的 AWS 資源非常有用。例如，使用者可能需要存取另一個 AWS 帳戶中的資源。它也能夠用來暫時獲得特殊許可，例如，提供多重要素驗證 (MFA)。您必須使用作用中的登入資料呼叫此 API。若要了解誰可以呼叫此操作，請參閱比較 AWS STS API 操作。如需更多詳細資訊，請參閱 建立角色以委派許可給 IAM 使用者 及 設定受 MFA 保護的 API 存取。

此呼叫必須使用有效的 AWS 安全憑證進行。當您進行此呼叫時，您傳遞了下列資訊：

- 應用程式應擔任角色的 Amazon Resource Name (ARN)。
- (選用) 持續時間，指定暫時安全憑證的持續時間。使用 `DurationSeconds` 參數來指定 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定之間的角色工作階段持續時間。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。如果您不傳遞此參數，暫時性憑證會在一個小時內到期。此 API 的 `DurationSeconds` 參數不同於您用來指定主控台工作階段持續時間的 `SessionDuration` HTTP 參數。在主控台登入權杖的聯合端點請求中使用 `SessionDuration` HTTP 參數。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。
- 角色工作階段名稱。當不同主體使用角色時，請使用此字串值來識別工作階段。基於安全考量，管理員可以在 AWS CloudTrail 日誌中檢視此欄位，以協助識別在 AWS中執行動作的人員。當您擔任該角色時，系統管理員可能會請求您將 IAM 使用者名稱指定為工作階段名稱。如需詳細資訊，請參閱 `sts:RoleSessionName`。

- (選用) 來源身分。您可以要求使用者在擔任角色時指定來源身分。設定來源身分之後，就無法變更值。針對在角色工作階段期間所採取的所有動作，這會出現在請求中。來源身分值在鏈結的角色工作階段間會持續存在。您可以使用 AWS CloudTrail 記錄檔中的來源身分識別資訊來決定誰對角色執行動作。如需有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。
- (選用) 內嵌或受管的工作階段政策。這些政策會限制角色工作階段獲派自角色的以身分為基礎政策的許可。所產生工作階段的許可會是角色的以身分為基礎的政策和工作階段政策的交集。工作階段政策不能用來授予超出即將擔任角色之以身分為基礎政策所允許的許可。如需有關角色工作階段許可的詳細資訊，請參閱 工作階段政策。
- (選用) 工作階段標籤。您可以承擔角色，然後使用暫時憑證提出請求。當您這麼做時，工作階段的主體標籤會包括角色標籤和傳遞的工作階段標籤。如果您使用暫時憑證發起呼叫，新的工作階段也會繼承來自呼叫發起工作階段的轉移工作階段標籤。如需有關工作階段標籤的詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。
- (選用) MFA 資訊。如果設定為使用多重要素驗證 (MFA)，則包含 MFA 裝置的識別符，以及該裝置所提供的一次性程式碼。
- (選用) `ExternalId` 值，可在將您的帳戶存取權委派給第三方時使用。這個值有助於確保只有指定的第三方可以存取角色。如需詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。

以下範例顯示使用 `AssumeRole` 的範例請求及回應。此範例請求假設指定時間內的 `demo` 角色具有已包含的工作階段政策、工作階段標籤、外部 ID 和來源身分。產生的工作階段名為 `John-session`。

Example 範例請求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=AssumeRole
&RoleSessionName=John-session
&RoleArn=arn:aws::iam::123456789012:role/demo
&Policy=%7B%22Version%22%3A%222012-10-17%22%2C%22Statement%22%3A%5B%7B%22Sid%22%3A
%20%22Stmt1%22%2C%22Effect%22%3A%20%22Allow%22%2C%22Action%22%3A%20%22s3%3A*%22%2C
%22Resource%22%3A%20%22*%22%7D%5D%7D
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&ExternalId=123ABC
&SourceIdentity=DevUser123
```

```
&AUTHPARAMS
```

在上述範例中顯示的政策值是以下政策的 URL 編碼版本：

```
{"Version":"2012-10-17","Statement":
[{"Sid":"Stmt1","Effect":"Allow","Action":"s3:*","Resource":"*"}]}
```

範例中的 AUTHPARAMS 參數是簽章的預留位置。簽章是您必須包含在 AWS HTTP API 要求中的驗證資訊。我們建議使用 AWS 開發套件來建立 API 請求。執行此作業的其中一個好處是開發套件會為您處理請求簽署。如果您必須手動建立和簽署 API 要求，AWS 請參閱中的「使用簽名版本 4 簽署請求」Amazon Web Services 一般參考以瞭解如何簽署請求。

除了暫時性安全憑證外，回應包含聯合身分使用者的 Amazon Resource Name (ARN) 和憑證過期時間。

Example 回應範例

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<AssumeRoleResult>
<SourceIdentity>DevUser123</SourceIdentity>
<Credentials>
  <SessionToken>
   AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQW
   LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
   QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
   9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
   +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCR/oLxBA==
  </SessionToken>
  <SecretAccessKey>
   wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2019-07-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
<AssumedRoleUser>
  <Arn>arn:aws:sts::123456789012:assumed-role/demo/John</Arn>
  <AssumedRoleId>ARO123EXAMPLE123:John</AssumedRoleId>
</AssumedRoleUser>
<PackedPolicySize>8</PackedPolicySize>
</AssumeRoleResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
```

```
</ResponseMetadata>
</AssumeRoleResponse>
```

> **Note**
>
> AWS 轉換會將傳遞的工作階段原則和工作階段標籤壓縮為具有單獨限制的封裝二進位格式。即使您的純文字符合其他需求，您的請求也可能因不符限制而失敗。PackedPolicySize 回應元素會按百分比指出請求的政策和標籤與大小上限的距離。

AssumeRoleWithWebIdentity—透過以 Web 為基礎的身分提供者聯合

這個 AssumeRoleWithWebIdentity API 操作會傳回資料聯合身分使用者的一組臨時安全登入操作，該使用者透過公有身分提供者進行驗證。公有身分提供者的範例包含 Login with Amazon、Facebook、Google 或任何與身分提供者相容的 OpenID Connect (OIDC)。此作業對於建立需要存取的行動應用程式或以用戶端為基礎的 Web 應用程式非常有用 AWS。使用此操作意味著您的使用者不需要自己的身分 AWS 或 IAM 身分。如需詳細資訊，請參閱 OIDC 聯盟。

建議您不要直接撥打電話AssumeRoleWithWebIdentity，而是使用 Amazon Cognito 和 Amazon Cognito 登入資料提供者搭配開發套件進行行動 AWS 開發。如需詳細資訊，請參閱 Amplify 文件 中的 Amplify 身分驗證。

如果您不是使用 Amazon Cognito，則呼叫 AWS STS的 AssumeRoleWithWebIdentity 動作。這是一個未簽署的呼叫，表示應用程式不需要存取任何 AWS 安全憑證，就能進行呼叫。當您進行此呼叫時，您傳遞了下列資訊：

- 應用程式應擔任角色的 Amazon Resource Name (ARN)。如果您的應用程式支援多種方式讓使用者登入，您必須定義多個角色，每個身分提供者一個角色。呼叫 AssumeRoleWithWebIdentity 應該包含角色的 ARN，其特定於使用者登入時所使用的提供者。
- 在應用程式驗證使用者後應用程式從 IdP 取得的權杖。
- 您可以設定 IdP，將屬性傳遞至權杖做為工作階段標籤。
- (選用) 持續時間，指定暫時安全憑證的持續時間。使用 DurationSeconds 參數來指定 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定之間的角色工作階段持續時間。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。如果您不傳遞此參數，暫時性憑證會在一個小時內到期。此 API 的 DurationSeconds 參數不同於您用來指定主控台工作階段持續時間的 SessionDuration HTTP 參數。在主控台登入權杖的聯合端點請求中使用 SessionDuration HTTP 參數。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

- 角色工作階段名稱。當不同主體使用角色時，請使用此字串值來識別工作階段。基於安全考量，管理員可以在 AWS CloudTrail 日誌中檢視此欄位，以了解在 AWS中執行動作的人員。當您擔任角色時，系統管理員可能會請求您提供工作階段名稱的特定值。如需詳細資訊，請參閱 `sts:RoleSessionName`。
- (選用) 來源身分。您可以要求聯合身分使用者在擔任角色時指定來源身分。設定來源身分之後，就無法變更值。針對在角色工作階段期間所採取的所有動作，這會出現在請求中。來源身分值在鏈結的角色工作階段間會持續存在。您可以使用 AWS CloudTrail 記錄檔中的來源身分識別資訊來決定誰對角色執行動作。如需有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。
- (選用) 內嵌或受管的工作階段政策。這些政策會限制角色工作階段獲派自角色的以身分為基礎政策的許可。所產生工作階段的許可會是角色的以身分為基礎的政策和工作階段政策的交集。工作階段政策不能用來授予超出即將擔任角色之以身分為基礎政策所允許的許可。如需有關角色工作階段許可的詳細資訊，請參閱 工作階段政策。

> **Note**
>
> `AssumeRoleWithWebIdentity` 呼叫未簽署 (加密)。因此，如果請求是透過信任的媒介傳輸，您應該只包含選用的工作階段政策。在這種情況下，有人可以更改政策以移除限制。

打電話時`AssumeRoleWithWebIdentity`， AWS 驗證令牌的真實性。例如，視提供者而定， AWS 可能會呼叫提供者，並包含應用程式已傳遞的權杖。假設身份提供程序驗證令牌， AWS 則將以下信息返回給您：

- 一組暫時性安全憑證。這些包含存取金鑰 ID、私密存取金鑰和工作階段權杖。
- 所擔任角色的角色 ID 和 ARN。
- `SubjectFromWebIdentityToken` 值，其中包含唯一的使用者 ID。

當您擁有臨時安全登入資料時，您可以使用它們進行 AWS API 呼叫。這與使用長期安全登入資料進行 AWS API 呼叫的程序相同。差別在於您必須包含工作階段權杖，這個權杖可讓 AWS 驗證臨時安全憑證是否有效。

您的應用程式應該快取憑證。預設情況下，如上所述，憑證在一小時後到期。如果您不在 AWS SDK 中使用 Amazonsts CredentialsProvider 操作，則由您和您的應用程序重新調`AssumeRoleWithWebIdentity`用。呼叫此操作，或取得一組新的暫時性安全憑證，之後舊的憑證才會過期。

AssumeRoleWithSAML — 透過與 SAML 2.0 相容的企業身分識別提供者進行聯合

這個 AssumeRoleWithSAML API 操作會傳回聯合身分使用者的一組臨時安全憑證，該使用者透過您組織現有的身分系統進行驗證。使用者還必須使用 SAML 2.0 (安全聲明標記語言) 來將身分驗證和授權資訊傳遞至 AWS。這個 API 操作適用於整合他們自己的身分系統的組織 (例如，Windows Active Directory 或 OpenLDAP)，而軟體可以產生 SAML 聲明。這種整合會提供有關使用者身分和許可 (例如 Active Directory Federation Services 或 Shibboleth) 的資訊。如需詳細資訊，請參閱 SAML 2.0 聯合身分。

> **Note**
>
> AssumeRoleWithSAML 呼叫未簽署 (加密)。因此，如果請求是透過信任的媒介傳輸，您應該只包含選用的工作階段政策。在這種情況下，有人可以更改政策以移除限制。

這是一個未簽署的呼叫，表示應用程式不需要存取任何 AWS 安全憑證，就能進行呼叫。當您進行此呼叫時，您傳遞了下列資訊：

- 應用程式應擔任角色的 Amazon Resource Name (ARN)。
- 在描述身分提供者的 IAM 中建立的 SAML 提供者的 ARN。
- SAML 身分提供者會在其對您應用程式的登入請求的身分驗證回應中，提供以基數 64 編碼的 SAML 聲明。
- 您可以設定 IdP，將屬性傳遞至 SAML 聲明做為工作階段標籤。
- (選用) 持續時間，指定暫時安全憑證的持續時間。使用 DurationSeconds 參數來指定 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定之間的角色工作階段持續時間。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。如果您不傳遞此參數，暫時性憑證會在一個小時內到期。此 API 的 DurationSeconds 參數不同於您用來指定主控台工作階段持續時間的 SessionDuration HTTP 參數。在主控台登入權杖的聯合端點請求中使用 SessionDuration HTTP 參數。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。
- (選用) 內嵌或受管的工作階段政策。這些政策會限制角色工作階段獲派自角色的以身分為基礎政策的許可。所產生工作階段的許可會是角色的以身分為基礎的政策和工作階段政策的交集。工作階段政策不能用來授予超出即將擔任角色之以身分為基礎政策所允許的許可。如需有關角色工作階段許可的詳細資訊，請參閱 工作階段政策。
- 角色工作階段名稱。當不同主體使用角色時，請使用此字串值來識別工作階段。基於安全考量，管理員可以在 AWS CloudTrail 日誌中檢視此欄位，以了解在 AWS中執行動作的人員。當

您擔任角色時，系統管理員可能會請求您提供工作階段名稱的特定值。如需詳細資訊，請參閱 sts:RoleSessionName。

- (選用) 來源身分。您可以要求聯合身分使用者在擔任角色時指定來源身分。設定來源身分之後，就無法變更值。針對在角色工作階段期間所採取的所有動作，這會出現在請求中。來源身分值在鏈結的角色工作階段間會持續存在。您可以使用 AWS CloudTrail 記錄檔中的來源身分識別資訊來決定誰對角色執行動作。如需有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

呼叫時AssumeRoleWithSAML， AWS 會驗證 SAML 宣告的真實性。假設身分識別提供者驗證宣告，AWS 則會將下列資訊傳回給您：

- 一組暫時性安全憑證。這些包含存取金鑰 ID、私密存取金鑰和工作階段權杖。
- 所擔任角色的角色 ID 和 ARN。
- Audience 值，其包含 SAML 聲明的 Recipient 元素的 SubjectConfirmationData 屬性值。
- Issuer 值，其包含 SAML 聲明的 Issuer 元素的值。
- 包含從 SAML 提供者的Issuer值、 AWS 帳戶 ID 和易記名稱建立的雜湊值的NameQualifier元素。當結合 Subject 元素時，他們可唯一識別聯合身分使用者。
- Subject 元素，其包含 SAML 聲明的 NameID 元素的 Subject 元素值。
- SubjectType 元素指出 Subject 元素的格式。此值可以是 persistent、transient 或在您的 SAML 聲明中使用的 Format 與 Subject 元素的完整 NameID URI。如需 NameID 元素的 Format 屬性的詳細資訊，請參閱 設定驗證回應的 SAML 宣告。

當您擁有臨時安全登入資料時，您可以使用它們進行 AWS API 呼叫。這與使用長期安全登入資料進行 AWS API 呼叫的程序相同。差別在於您必須包含工作階段權杖，這個權杖可讓 AWS 驗證臨時安全憑證是否有效。

您的應用程式應該快取憑證。預設情況下，憑證會在一小時後到期。如果您沒有在 AWS SDK 中使用 Amazonsts CredentialsProvider 操作，則由您和您的應用程序重新調AssumeRoleWithSAML用。呼叫此操作，或取得一組新的暫時性安全憑證，之後舊的憑證才會過期。

GetFederationToken—透過自訂身分經紀人聯合

這個 GetFederationToken API 操作會傳回一組臨時安全憑證供聯合身分使用者使用。此 API 與 AssumeRole 不同，其中的預設過期時段是實質上比較長 (12 小時，而不是 1 小時)。此外，您可以使用 DurationSeconds 參數來指定暫時性安全憑證的持續時間，以保持有效。產生的認證在指定的持續時間內有效，介於 900 秒 (15 分鐘) 到 129,600 秒 (36 小時) 之間。較長的到期時間有助於減少呼叫次數， AWS 因為您不需要經常取得新的認證。

當您提出此請求時，您可以使用特定 IAM 使用者的憑證。臨時安全憑證的許可，取決於您呼叫 GetFederationToken 時傳遞的工作階段政策。所產生工作階段的許可會是 IAM 使用者政策和您傳遞之工作階段政策的交集。工作階段政策不能用來授予超出即將請求聯合的 IAM 使用者之以身分為基礎政策所允許的許可。如需有關角色工作階段許可的詳細資訊，請參閱 工作階段政策。

當您使用 GetFederationToken 操作傳回的暫時憑證時，工作階段的主體標籤會包含使用者的標籤和傳遞的工作階段標籤。如需有關工作階段標籤的詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

GetFederationToken 呼叫會傳回暫時性安全憑證，該憑證由工作階段字符、存取金鑰、私密金鑰和過期時間組成。如果您想要管理您組織內的許可 (例如，使用 Proxy 應用程式以指派許可)，您可以使用 GetFederationToken。

以下範例顯示使用 GetFederationToken 的範例請求及回應。此範例請求會將指定期間的呼叫發起使用者與工作階段政策 ARN 和工作階段標籤聯合在一起。產生的工作階段名為 Jane-session。

Example 範例請求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetFederationToken
&Name=Jane-session
&PolicyArns.member.1.arn==arn%3Aaws%3Aiam%3A%3A123456789012%3Apolicy%2FRole1policy
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&AUTHPARAMS
```

上述範例中顯示的政策 ARN 包含以下 URL 編碼 ARN：

```
arn:aws:iam::123456789012:policy/Role1policy
```

此外，請注意範例中的 &AUTHPARAMS 參數表示身分驗證資訊的預留位置。這是簽名，您必須將其包含在 AWS HTTP API 請求中。我們建議使用 AWS 開發套件來建立 API 請求。執行此作業的其中一個好處是開發套件會為您處理請求簽署。如果您必須手動建立和簽署 API 要求，AWS 請前往中的「Amazon Web Services 一般參考使用簽名版本 4 簽署請求」，瞭解如何簽署請求。

除了暫時性安全憑證外，回應包含聯合身分使用者的 Amazon Resource Name (ARN) 和憑證過期時間。

Example 回應範例

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetFederationTokenResult>
<Credentials>
  <SessionToken>
   AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQW
   LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
   QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
   9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
   +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCEXAMPLE==
  </SessionToken>
  <SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2019-04-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE;</AccessKeyId>
</Credentials>
<FederatedUser>
  <Arn>arn:aws:sts::123456789012:federated-user/Jean</Arn>
  <FederatedUserId>123456789012:Jean</FederatedUserId>
</FederatedUser>
<PackedPolicySize>4</PackedPolicySize>
</GetFederationTokenResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</GetFederationTokenResponse>
```

> **Note**
>
> AWS 轉換會將傳遞的工作階段原則和工作階段標籤壓縮為具有單獨限制的封裝二進位格式。即使您的純文字符合其他需求，您的請求也可能因不符限制而失敗。PackedPolicySize 回應元素會按百分比指出請求的政策和標籤與大小上限的距離。

AWS 建議您在資源層級授與許可 (例如，將以資源為基礎的政策附加到 Amazon S3 儲存貯體)，您可以省略該Policy參數。不過，如果您不包含聯合身分使用者的政策，暫時性安全憑證將不會授予任何許可。在這種情況下，您「必須」使用資源政策，來授予聯合身分使用者存取您 AWS 資源的許可。

例如，假設您的 AWS 帳戶 號碼是 111122223333，而且您有一個想要允許蘇珊訪問的 Amazon S3 存儲桶。Susan 的暫時性安全憑證不包含儲存貯體的政策。在這種情況下，您需要確保儲存貯體具有其 ARN 與 Susan 的 ARN 相符的政策，例如 `arn:aws:sts::111122223333:federated-user/Susan`。

— GetSessionToken 不信任環境中使用者的暫時憑證

這個 `GetSessionToken` API 操作會傳回一組臨時安全憑證供現有 IAM 使用者使用。這對於提供增強的安全性非常有用，例如僅在為 IAM 使用者啟用 MFA 時允許 AWS 請求。因為憑證為暫時性，它們可在您有透過較不安全環境存取您資源的 IAM 使用者時，提供增強的安全性。較不安全環境的範例包含行動裝置或 Web 瀏覽器。如需詳細資訊，請參閱 AWS Security Token Service API 參考GetSessionToken中的請求暫時性安全憑證或。

在預設情況下，IAM 使用者的暫時性安全憑證的有效期最長為 12 小時。但是，您可以使用 `DurationSeconds` 參數請求持續時間最短為 15 分鐘最長為 36 小時。出於安全原因，令牌 AWS 帳戶根使用者 的持續時間限制為一小時。

`GetSessionToken` 會傳回暫時性安全憑證，其由工作階段字符、存取金鑰 ID 和私密存取金鑰組成。以下範例顯示使用 `GetSessionToken` 的範例請求及回應。回應還包括暫時性安全憑證的過期時間。

Example 範例請求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=1800
&AUTHPARAMS
```

範例中的 `AUTHPARAMS` 參數是簽章的預留位置。簽章是您必須包含在 AWS HTTP API 要求中的驗證資訊。我們建議使用 AWS 開發套件來建立 API 請求。執行此作業的其中一個好處是開發套件會為您處理請求簽署。如果您必須手動建立和簽署 API 要求，AWS 請前往中的「Amazon Web Services 一般參考使用簽名版本 4 簽署請求」，瞭解如何簽署請求。

Example 回應範例

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetSessionTokenResult>
<Credentials>
  <SessionToken>
```

```
    AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/L
    To6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgRmpRV3z
    rkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/AXlzBBko7b15fjrBs2+cTQtp
    Z3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE
  </SessionToken>
  <SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2011-07-11T19:55:29.611Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
</GetSessionTokenResult>
<ResponseMetadata>
<RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>
</ResponseMetadata>
</GetSessionTokenResponse>
```

或者，GetSessionToken要求可以包含SerialNumber和 AWS 多重要素驗證 (MFA) 驗證的TokenCode值。如果提供的值有效，請 AWS STS 提供包含 MFA 驗證狀態的臨時安全登入資料。然後，只要 MFA 驗證有效，就可以使用臨時安全登入資料來存取受 MFA 保護的 API 作業或 AWS 網站。

以下範例顯示 GetSessionToken 請求，其中包含 MFA 驗證碼和裝置序號。

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=7200
&SerialNumber=YourMFADeviceSerialNumber
&TokenCode=123456
&AUTHPARAMS
```

> **Note**
>
> 呼叫 AWS STS 可以是全球端點或您啟動的任何區域端點 AWS 帳戶。如需詳細資訊，請參閱區域與端點的AWS STS 一節。
>
> 範例中的 AUTHPARAMS 參數是簽章的預留位置。簽章是您必須包含在 AWS HTTP API 要求中的驗證資訊。我們建議使用 AWS 開發套件來建立 API 請求。執行此作業的其中一個好處是開發套件會為您處理請求簽署。如果您必須手動建立和簽署 API 要求，AWS 請參閱中的「使用簽名版本 4 簽署請求」Amazon Web Services 一般參考以瞭解如何簽署請求。

比較 AWS STS API 操作

下表比較傳回臨時安全登入資料之 AWS STS API 作業的功能。若要了解可用來在擔任角色時請求暫時性安全憑證的不同方法，請參閱使用 IAM 角色。若要瞭解可讓您傳遞工作階段標籤的不同 AWS STS API 作業，請參閱傳遞工作階段標籤 AWS STS。

比較您的 API 選項

AWS STS API	誰可以呼叫	憑證存留期 (最小 | 最大 | 預設)	MFA 支援[1]	工作階段政策支援[2]	產生的暫時憑證限制
AssumeRole	具現有暫時性安全憑證的 IAM 使用者或 IAM 角色	15 分鐘 | 最大工作階段持續時間設定[3] | 1 小時	是	是	無法呼叫 GetFederationToken 或 GetSessionToken 。
AssumeRoleWithSAML	任何使用者；呼叫者必須傳遞 SAML 身分驗證回應，指出通過已知身分提供者的身分驗證	15 分鐘 | 最大工作階段持續時間設定[3] | 1 小時	否	是	無法呼叫 GetFederationToken 或 GetSessionToken 。
AssumeRoleWithWebIdentity	任何用戶; 調用者必須傳遞符合 OIDC 的 JWT 令牌，該令牌指示來自已知身份提供者的身份驗證	15 分鐘 | 最大工作階段持續時間設定[3] | 1 小時	否	是	無法呼叫 GetFederationToken 或 GetSessionToken 。
GetFederationToken	IAM 使用者或 AWS 帳戶根使用者	IAM 使用者：15 分鐘 | 36 小	否	是	無法使用 AWS CLI 或 AWS API 呼叫 IAM 作業。此限制不適用主控台工作階段。

AWS STS API	誰可以呼叫	憑證存留期 (最小 | 最大 | 預設)	MFA 支援[1]	工作階段政策支援[2]	產生的暫時憑證限制
		時 | 12 小時 根使用者：15 分鐘 | 1 小時 | 1 小時			無法調用除 `GetCallerIdentity` .以外的 AWS STS 操作 允許 SSO 到主控台。[5]
GetSessionToken	IAM 使用者或 AWS 帳戶根使用者	IAM 使用者：15 分鐘 | 36 小時 | 12 小時 根使用者：15 分鐘 | 1 小時 | 1 小時	是	否	除非請求包含 MFA 資訊，否則無法呼叫 IAM API 操作。 除了`AssumeRole` 或之外，無法呼叫 AWS STS API 作業`GetCallerIdentity` 。 不允許 SSO 到主控台。[6]

[1] MFA 支援。當您呼叫 AssumeRole 和 GetSessionToken API 作業時，您可以包含多因素驗證 (MFA) 裝置的相關資訊。這可確保 API 呼叫所產生的暫時性安全憑證，僅可由使用 MFA 裝置進行身分驗證的使用者使用。如需詳細資訊，請參閱 設定受 MFA 保護的 API 存取。

[2] 工作階段政策支援。工作階段政策是一種政策，且您會在以程式設計方式建立角色或聯合身分使用者的臨時工作階段時，做為參數進行傳遞。這個政策會限制工作階段獲派自角色或使用者之以身分為基礎政策的許可。所產生工作階段的許可會是實體的身分類型政策和工作階段政策的交集。工作階段政策不能用來授予超出即將擔任角色之以身分為基礎政策所允許的許可。如需有關角色工作階段許可的詳細資訊，請參閱 工作階段政策。

[3] 最大工作階段持續時間設定。使用 `DurationSeconds` 參數來指定 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定之間的角色工作階段持續時間。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。

GetCallerIdentity. 執行此操作不需要任何許可。如果管理員將明確拒絕存取 `sts:GetCallerIdentity` 動作的政策新增到 IAM 使用者或角色，您仍然可以執行此操作。不需要許可，因為當 IAM 使用者或角色被拒絕存取時，會傳回相同的資訊。若要檢視範例回應，請參閱 我沒有授權執行：IAM：DeleteVirtualMFA 設備。

[5] Single sign-on (SSO) 至主控台。若要支援 SSO， AWS 可讓您呼叫同盟端點 (`https://signin.aws.amazon.com/federation`) 並傳遞臨時安全性登入資料。端點會傳回權杖，您可用來建構 URL，直接將使用者登入主控台，而不需要密碼。如需詳細資訊，請參閱 AWS 安全性部落格中的啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console和如何啟用 AWS 管理主控台的跨帳戶存取。

擷取臨時登入資料後，您無法透 AWS Management Console 過將認證傳遞至同盟單一登入端點來存取。如需更多詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

搭配使用暫時憑證與 AWS 資源

您可以使用臨時安全登入資料，使用 AWS CLI 或 AWS API (使用 AWS SDK) 對 AWS 資源進行程式設計要求。暫時性憑證提供的許可與長期安全憑證許可相同，例如 IAM 使用者憑證。不過，有幾個差異：

- 當您使用臨時安全憑據進行呼叫時，調用必須包含會話令牌，該令牌與這些臨時憑據一起返回。AWS 使用會話令牌來驗證臨時安全憑據。
- 暫時性憑證在指定的間隔時間後到期。暫時性憑證到期後，使用這些憑證進行的任何呼叫都將失敗，所以您必須產生一組新的暫時性憑證。暫時性憑證無法延期或重新整理超過原始指定間隔。
- 當您使用暫時憑證發出請求時，您的主體可能會包含一組標籤。這些標籤來自工作階段標籤和連接到您擔任角色的標籤。如需有關工作階段標籤的詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

如果您使用的是 AWS SDK、AWS Command Line Interface(AWS CLI) 或 Windows PowerShell 工具，取得和使用臨時安全性登入資料的方式會隨內容而有所不同。如果您在 EC2 執行個體內執行程式碼或 Windows 專用工具 PowerShell 命令，您可以利用適用於 Amazon EC2 的角色。 AWS CLI否則，您可以呼叫 AWS STS API 以取得暫時憑證，然後明確地使用他們呼叫 AWS 服務。

> **Note**
>
> 您可以使用 AWS Security Token Service (AWS STS) 建立並為受信任的使用者提供可控制資源存取權的臨時安全登入 AWS 資料。如需有關的更多資訊 AWS STS，請參閱IAM 中的暫時安全憑證。 AWS STS 是具有預設端點的全域服務`https://sts.amazonaws.com`。此端點位在美國東部 (維吉尼亞北部) 區域，但您從此端點和其他端點取得的憑證為全域有效。這些憑證可配合任何區域中的服務和資源使用。您也可以選擇對任何支援的區域中的端點進行 AWS STS API 呼叫。從地理位置較靠近您的區域的伺服器提出請求，可以降低延遲發生機率。無論您的憑證來自哪個區域，都能全域使用。如需詳細資訊，請參閱 AWS STS 在一個管理 AWS 區域。

內容

在 Amazon EC2 執行個體中使用暫時憑證

如果您想要在 EC2 執行個體內執行 AWS CLI 命令或程式碼，建議取得登入資料的方式是使用Amazon EC2 的角色。您建立 IAM 角色，指定要授予在 EC2 執行個體上執行之應用程式的許可。當您啟動執行個體時，會建立角色與執行個體的關聯。

然後 AWS CLI，在執行個體上執行的 Windows PowerShell 命令適用的應用程式、和工具可以從執行個體中繼資料取得自動暫時安全登入資料。您不必明確取得暫時安全憑證。適用於 Windows 的 AWS 開發套件和工具 PowerShell 會自動從 EC2 執行個體中繼資料服務 (IMDS) 取得登入資料並加以使用。AWS CLI暫時性憑證擁有您為與執行個體定義關聯的角色所定義的許可。

如需詳細資訊及範例，請參閱下列：

- 使用 IAM 角色授予對 Amazon 彈性運算雲端上 AWS 資源的存取權 — AWS SDK for Java
- 使用 IAM 角色授予存取權限 — AWS SDK for .NET
- 建立角色 — AWS SDK for Ruby

使用暫時性安全憑證與 AWS 開發套件

要在代碼中使用臨時安全憑據，您可以通過編程方式調用類似的 AWS STS API AssumeRole 並提取生成的憑據和會話令牌。然後，您可以使用這些值作為後續呼叫的認證 AWS。下列範例顯示如何在使用 SDK 時使用臨時安全性憑證的虛擬程式碼： AWS

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
   assumeRoleResult.AccessKeyId,
   assumeRoleResult.SecretAccessKey,
   assumeRoleResult.SessionToken);
s3Request = CreateAmazonS3Client(tempCredentials);
```

如需使用 Python 撰寫的範例 (使用 AWS SDK for Python (Boto))，請參閱切換到身分與存取權管理角色 (AWS API)。此範例會示範如何呼叫 AssumeRole 以取得暫時安全憑證，然後使用這些憑證呼叫 Amazon S3。

如需如何呼叫 AssumeRole、GetFederationToken 以及其他 API 操作的詳細資訊，請參閱 AWS Security Token Service API 參考。有關從結果取得臨時安全憑證和工作階段權杖的詳細資訊，請參閱您正在使用的軟體開發套件的說明文件。您可以在主要文件頁面的 AWS SDK 和工具組區段中找到所有 SDK 的AWS 文件。

您必須在舊的憑證到期之前，確實取得一組新的憑證。在某些軟體開發套件中，您可以使用為您管理重新整理憑證程序的提供者；檢查您正在使用的開發套件的文件。

使用暫時性安全憑證與 AWS CLI

您可以使用暫時性安全憑證與 AWS CLI。這很適合用於測試政策。

若使用 AWS CLI，您便可以呼叫 AWS STS API (例如 AssumeRole 或 GetFederationToken)，然後擷取產生的輸出。以下範例顯示對 AssumeRole 的呼叫，其會傳送輸出至檔案。在此範例中，會假設profile參數是組態檔案中的設 AWS CLI 定檔。同時也假設會參考有權擔任角色之 IAM 使用者的憑證。

```
aws sts assume-role --role-arn arn:aws:iam::123456789012:role/role-name --role-session-
name "RoleSession1" --profile IAM-user-name > assume-role-output.txt
```

當命令完成時，您可以從您路由的任何地方擷取存取金鑰 ID、私密存取金鑰和工作階段權杖。您可以手動或使用指令碼執行這項作業。然後，您可以將這些值指派給環境變數。

當您執行 AWS CLI 命令時，會以特定順序 AWS CLI 尋找認證 — 首先在環境變數中，然後在組態檔案中尋找認證。因此，在您將暫存認證放入環境變數之後，依預設 AWS CLI 會使用這些認證。(如果您在指令中指定`profile`參數，則會 AWS CLI 略過環境變數。 而是在配置文件中查 AWS CLI 找，如果需要，它可以讓您覆蓋環境變量中的認證。)

下列範例顯示如何設定暫時安全性登入資料的環境變數，然後呼叫 AWS CLI 命令。由於 AWS CLI 命令中未包含任何`profile`參數，因此會先在環境變數中 AWS CLI 尋找認證，因此會使用暫時認證。

Linux

```
$ export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
$ aws ec2 describe-instances --region us-west-1
```

Windows

```
C:\> SET AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
C:\> SET AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of token>
C:\> aws ec2 describe-instances --region us-west-1
```

使用暫時性安全憑證與 API 操作

如果您直接向 HTTPS API 請求發出 AWS，則可以使用從 AWS Security Token Service (AWS STS) 獲得的臨時安全登入資料來簽署這些請求。若要這麼做，您可以使用從中接收的存取金鑰 ID 和秘密存取金鑰 AWS STS。使用存取金鑰 ID 和私密存取金鑰簽署請求的方式，就和使用長期憑證簽署請求一樣。您還將從中接收的會話令牌添加到 API 請求中 AWS STS。您將工作階段權杖新增到 HTTP 標頭，或查詢名為 `X-Amz-Security-Token` 的字串參數。您將工作階段權杖新增到 HTTP 標頭「或」查詢字串參數，但不是兩者。如需簽署 HTTPS API 要求的詳細資訊，請參閱 AWS 一般參考. AWS

其他資訊

如需 AWS STS 與其他 AWS 服務搭配使用的詳細資訊，請參閱下列連結：

- Amazon S3 請參閱 《Amazon Simple Storage Service 使用者指南》中的使用 IAM 使用者暫時憑證提出請求或使用聯合身分使用者暫時憑證提出請求。
- Amazon SNS。請參閱 Amazon 簡單通知服務開發人員指南中的將身分型政策與 Amazon SNS 搭配使用。

- Amazon SQS 請參閱 Amazon 簡單佇列服務開發人員指南中的 Amazon SQS 中的身分識別和存取管理。
- Amazon SimpleDB。請參閱使用臨時安全憑證中的Amazon SimpleDB 開發人員指南。

控制臨時安全安全憑證的許可

您可以使用 AWS Security Token Service (AWS STS) 建立並為受信任的使用者提供可控制資源存取權的臨時安全登入 AWS 資料。如需有關的更多資訊 AWS STS，請參閱IAM 中的暫時安全憑證。在 AWS STS 發出臨時安全憑證後，它們在過期期間有效，且無法撤銷。但是，每次發出使用憑證的請求時，都會評估指派給臨時安全憑證的許可，因此您可以透過在發出憑證後變更其存取權來達到撤銷憑證的效果。

下列主題假設您具有 AWS 權限和原則的工作知識。如需有關這些主題的詳細資訊，請參閱 AWS 資源存取管理。

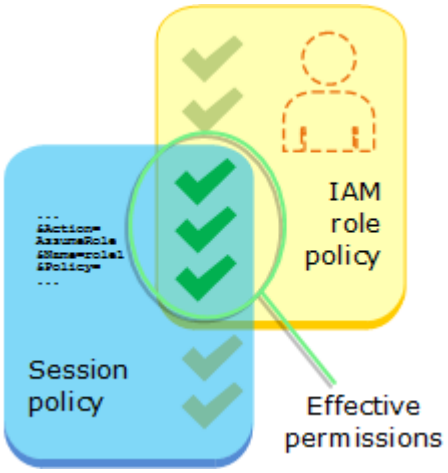
主題

- AssumeRole、 AssumeRoleWith SAML 和的權限 AssumeRoleWithWebIdentity
- 監控並控制使用擔任角色所採取的動作
- GetFederationToken 的許可
- GetSessionToken 的許可
- 停用臨時安全性憑證的許可
- 授予許可以建立暫時性安全憑證
- 授與使用識別感知主控台工作階段的權限

AssumeRole、 AssumeRoleWith SAML 和的權限 AssumeRoleWithWebIdentity

所擔任角色的許可政策，將會決定由 `AssumeRole`、`AssumeRoleWithSAML` 和 `AssumeRoleWithWebIdentity` 傳回的臨時安全憑證的許可。當您建立或更新角色時定義這些許可。

或者，您可以傳遞內嵌或受管的工作階段政策作為 `AssumeRole`、`AssumeRoleWithSAML` 或 `AssumeRoleWithWebIdentity` API 操作的參數。工作階段政策會為該角色的臨時憑證工作階段限制許可。所產生工作階段的許可會是角色的身分類型政策和工作階段政策的交集。您可以在後續 AWS API 呼叫中使用該角色的臨時登入資料，以存取擁有該角色的帳戶中的資源。您無法使用工作階段政策來授予超出即將擔任角色之以身分為基礎政策所允許的許可。若要進一步了解 AWS 如何決定角色的有效許可，請參閱政策評估邏輯。



在進行「允許」或「拒絕」授權決策 AWS 時，不AssumeRole會評估原始呼叫之認證所附加的原則。使用者暫時放棄其原始許可，以支援由擔任角色指派的許可。在AssumeRoleWithSAML和AssumeRoleWithWebIdentity API 操作的情況下，由於 API 的調用者不是 AWS 身份，因此沒有要評估的策略。

範例：使用指派權限 AssumeRole

您可以將 AssumeRole API 操作與不同類型的政策一起使用。以下是幾個範例。

角色許可政策

在此範例中，您在呼叫 AssumeRole API 操作時未使用選用 Policy 參數來指定工作階段。指派給臨時憑證的許可由所擔任角色的許可政策決定。以下範例許可政策授予角色許可，以列出名為 productionapp 的 S3 儲存貯體中包含的所有物件。它還允許角色取得、放置和刪除該儲存貯體中的物件。

Example 角色許可政策範例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

工作階段政策作為參數傳遞

假設您想要讓使用者擔任與上一個範例相同的角色。但是，此時您希望該角色工作階段擁有的許可，只允許其取得 productionapp S3 儲存貯體中的物件，以及在其中放入物件。您不想讓他們刪除物件。實現此目的一種方法是建立新的角色並在該角色的許可政策中指定所需許可。實現此目的之另一種方法是呼叫 AssumeRole API，並將工作階段政策包含在可選的 Policy 參數做為 API 操作的一部分。所產生工作階段的許可會是角色的以身分為基礎的政策和工作階段政策的交集。工作階段政策不能用來授予超出即將擔任角色之以身分為基礎政策所允許的許可。如需有關角色工作階段許可的詳細資訊，請參閱 工作階段政策。

擷取新的工作階段臨時憑證後，您可以將這些資料傳遞給您希望擁有這些許可的使用者。

例如，假設以下政策做為 API 呼叫的參數傳遞。使用這項工作階段的人員所擁有的許可，將僅能執行下列動作：

- 列出 productionapp 儲存貯體中的所有物件。
- 取得並將物件放入 productionapp 儲存貯體中。

在以下工作階段政策中，s3:DeleteObject 許可會遭篩除，而且擔任的工作階段不會獲得 s3:DeleteObject 許可的授予。此政策設定角色工作階段的最大許可，並供以覆寫該角色的任何現有許可政策。

Example 工作階段使用**AssumeRole** API 呼叫傳遞政策範例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

以資源為基礎政策

某些 AWS 資源支援以資源為基礎的原則，而這些原則提供另一種機制來定義會影響臨時安全登入資料的權限 只有少數資源 (例如 Amazon S3 儲存貯體、Amazon SNS 主題和 Amazon SQS 佇列) 支援資源式政策。以下範例使用名為 `productionapp` 的 S3 儲存貯體擴展前面的範例。以下政策連接至儲存貯體。

當您將下面以資源為基礎的政策連接到 `productionapp` 儲存貯體時，將拒絕所有使用者從儲存貯體中刪除物件的許可。(請參閱政策中的 `Principal` 元素。) 這包括所有擔任角色使用者，即使角色許可政策授予 `DeleteObject` 許可。明確 `Deny` 陳述式一律優先於 `Allow` 陳述式。

Example 儲存貯體政策的範例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "*"},
    "Effect": "Deny",
    "Action": "s3:DeleteObject",
    "Resource": "arn:aws:s3:::productionapp/*"
  }
}
```

如需如何結合及評估多個原則類型的詳細資訊 AWS，請參閱政策評估邏輯。

監控並控制使用擔任角色所採取的動作

IAM 角色是 IAM 中受指派許可的物件。當您從外部使用 IAM 身分或身分假設該角色時 AWS，您會收到一個工作階段，其中包含指派給該角色的許可。

當您在中執行動作時 AWS，可以記錄有關工作階段的資訊，以 AWS CloudTrail 便您的帳戶管理員監控。管理員可以將角色設定為要求身分傳遞自訂字串，以識別在 AWS中執行動作的人員或應用程式。

此身分資訊會儲存為 AWS CloudTrail中的來源身分。當管理員檢閱中的活動時 CloudTrail，他們可以檢視來源身分識別資訊，以決定使用假設角色工作階段執行動作的人員或哪些動作。

設定來源識別之後，就會出現在角色工作階段期間所 AWS 採取的任何動作的要求中。當角色透過 AWS CLI 或 AWS API (稱為角色鏈結) 擔任其他角色時，所設定的值仍然存在。設定的值無法在角色工作階段期間變更。系統管理員可以根據來源身分識別的存在狀態或值來設定精細的權限，以進一步控制使用共用角色執行的 AWS 動作。您可以決定是否可以使用來源身分屬性、是否需要，以及可以使用的數值。

您使用來源身分的方式與角色工作階段名稱和工作階段標記有很大的不同。來源身分值在設定之後即無法變更，而且對於對角色工作階段採取的任何其他動作，會持續存在。以下說明如何使用工作階段標籤和角色工作階段名稱：

- 工作階段標籤 – 您也可以在擔任角色或與使用者聯合身分時傳遞工作階段標籤。擔任角色時，工作階段標籤即會出現。您可以定義使用標籤條件金鑰的政策，以根據主體的標籤來授與許可。然後，您可以使用 CloudTrail 來檢視為擔任角色或同盟使用者所做的要求。若要進一步了解工作階段標籤，請參閱 傳遞工作階段標籤 AWS STS。
- 角色工作階段名稱 – 您可以在角色信任政策中使用 `sts:RoleSessionName` 條件金鑰，請求您的使用者在擔任角色時提供特定的工作階段名稱。當不同主體使用角色時，角色工作階段名稱可用來區分角色工作階段。若要深入瞭解角色工作階段名稱，請參閱 sts: RoleSessionName。

當您想要控制擔任角色的身分時，建議您使用來源身分。來源識別對於採礦 CloudTrail 記錄檔來判斷使用此角色執行動作的使用者也很有用。

主題

- 設定以使用來源身分
- 來源身分的須知事項
- 設定來源身分所需的許可
- 擔任角色時指定來源身分
- 使用來源身分 AssumeRole
- 搭配 AssumeRoleWith SAML 使用來源身分識別
- 使用來源身分 AssumeRoleWithWebIdentity
- 使用來源身分資訊控制存取
- 檢視來源身分 CloudTrail

設定以使用來源身分

您設定為使用來源身分的方式將取決於擔任角色時所使用的方法。例如，您的 IAM 使用者可能會直接使用 AssumeRole 操作擔任角色。如果您擁有企業身分識別 (也稱為員工身分識別)，他們可能 AWS 會使用AssumeRoleWithSAML. 如果最終使用者要存取您的行動或 Web 應用程式，他們可能會使用 AssumeRoleWithWebIdentity 來存取。以下是高階工作流程概觀，可協助您瞭解如何設定以利用現有環境中的來源身分資訊。

1. 設定測試使用者和角色 – 使用生產前環境，設定測試使用者和角色，並設定其政策以允許設定來源身分。

   如果您使用身分提供者 (IdP) 作為聯合身分，請將 IdP 設定為將您選擇的使用者屬性傳遞至聲明或權杖中的來源身分。
2. 擔任角色 – 測試擔任角色，並使用您設定要進行測試的使用者和角色傳遞來源身分。
3. 檢閱 CloudTrail — 檢閱記 CloudTrail 錄中測試角色的來源身分識別資訊。
4. 培訓您的使用者 – 在生產前環境中進行測試之後，請確保您的使用者知道如何傳入來源身分資訊 (如有必要)。設定您要求使用者在生產環境中提供來源身分的截止日期。
5. 設定生產政策 – 為您的生產環境設定政策，然後將政策新增至您的生產使用者和角色。
6. 監視活動 — 使用 CloudTrail 記錄監視您的生產角色活動。

來源身分的須知事項

使用來源身分時請記住下列事項。

- 連線至身分提供者 (IdP) 之所有角色的信任政策均須具有 sts:SetSourceIdentity 許可。對於角色信任政策中沒有此許可的角色，AssumeRole* 操作將會失敗。如果您不想更新每個角色的角色信任政策，則您可以使用個別的 IdP 執行個體傳遞來源身分。然後將 sts:SetSourceIdentity 許可僅新增至連線至個別 IdP 的角色。
- 當身分設定來源身分時，sts:SourceIdentity 金鑰會存在於請求中。針對在角色工作階段期間採取的後續動作，aws:SourceIdentity 金鑰會出現在請求中。 AWS 不會在 sts:SourceIdentity 或 aws:SourceIdentity 金鑰中控制來源身分的值。如果您選擇要求來源身分，則必須選擇要讓使用者或 IdP 提供的屬性。基於安全考量，您必須確保您可以控制提供這些值的方式。
- 來源身分中的值長度必須介於 2 到 64 個字元之間，只能包含字母數位字元、底線和以下字元：. , + = @ - (連字號)。您不能使用以文字 **aws:** 為開頭的值。此前綴保留供 AWS 內部使用。

- CloudTrail 當 AWS 服務或服務連結角色代表聯合身分或員工身分執行動作時，不會擷取來源身分識別資訊。

> **Important**
>
> 假設角色時，您無法切換至中需要設定來源識別的角色。 AWS Management Console 若要擔任此類角色，您可以使用 AWS CLI 或 AWS API 呼叫`AssumeRole`作業並指定來源識別參數。

設定來源身分所需的許可

除了符合 API 操作的動作外，您必須在您的政策中具備下列僅許可動作：

```
sts:SetSourceIdentity
```

- 若要指定來源身分，主體 (IAM 使用者和角色) 必須具有 `sts:SetSourceIdentity` 的許可。身為系統管理員，您可以在角色信任政策和主體的權限政策中設定此選項。
- 當您擔任另一個角色的角色時，稱為角色鏈結，在主體 (擔任角色) 的許可政策和目標角色的角色信任政策中都需要 `sts:SetSourceIdentity` 的許可。否則，擔任角色操作將會失敗。
- 使用來源身分時，連線至 IdP 之所有角色的角色信任政策必須具有 `sts:SetSourceIdentity` 許可。任何連接至 IdP 的角色，在沒有此許可的情況下，`AssumeRole*` 操作將會失敗。如果您不想更新每個角色的角色信任政策，則您可以使用個別的 IdP 執行個體傳遞來源身分，然後將 `sts:SetSourceIdentity` 許可僅新增至連線至個別 IdP 的角色
- 若要跨帳戶界限設定來源身分，您必須在兩個地方均包含 `sts:SetSourceIdentity` 許可。此許可必須存在於原始帳戶中主體的許可政策，以及目標帳戶中角色的角色信任政策。您可能需要執行此操作，例如，當一個角色被用來在另一個具有角色鏈結的帳戶中擔任一個角色時。

身為帳戶管理員，假設您想要允許您帳戶中的 IAM 使用者 `DevUser` 在同一個帳戶中擔任 `Developer_Role`。但是，您希望只有在使用者已將來源身分設定為其 IAM 使用者名稱時，才允許此動作。您可將下列政策連接至 IAM 使用者。

Example 附加至以身分為基礎的原則範例 DevUser

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Sid": "AssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role"
    },
    {
      "Sid": "SetAwsUserNameAsSourceIdentity",
      "Effect": "Allow",
      "Action": "sts:SetSourceIdentity",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role",
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": "${aws:username}"
        }
      }
    }
  ]
}
```

若要強制執行可接受的來源身分值，您可以設定下列角色信任政策。政策會為 IAM 使用者提供DevUser許可來擔任角色並設定來源身分。sts:SourceIdentity 條件索引鍵定義了可接受的來源身分值。

Example 來源身分的角色信任政策範例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevUserAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/DevUser"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "sts:SourceIdentity": "DevUser"
        }
```

```
      }
    }
  ]
}
```

使用者使用 IAM 使用者的登入資料DevUser，嘗試假設DeveloperRole使用下列 AWS CLI 要求。

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer_Role \
--role-session-name Dev-project \
--source-identity DevUser \
```

AWS 評估請求時，請求內容會包含sts:SourceIdentity的DevUser。

擔任角色時指定來源身分

當您使用其中一個 AWS STS AssumeRole* API 作業取得角色的臨時安全登入資料時，您可以指定來源身分識別。您使用的 API 操作會視您的使用案例而有所不同。例如，如果您使用 IAM 角色授與 IAM 使用者存取他們通常無法存取的 AWS 資源，則可以使用該AssumeRole作業。如果您使用企業聯合身分來管理人力使用者，可以使用 AssumeRoleWithSAML 操作。如果您使用 OIDC 聯盟來允許使用者存取您的行動或 Web 應用程式，則可以使用此AssumeRoleWithWebIdentity作業。以下各節說明如何在每項操作中使用來源身分。若要進一步了解暫時憑證的常見案例，請參閱暫時性憑證的常見案例。

使用來源身分 AssumeRole

此AssumeRole作業會傳回一組可用來存取 AWS 資源的暫時認證。您可以使用 IAM 使用者或角色憑證來呼叫 AssumeRole。若要在擔任角色時傳遞來源身分識別，請使用--source-identity AWS CLI 選項或 SourceIdentity AWS API 參數。以下範例顯示如何使用 AWS CLI來指定來源身分。

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/developer \
--role-session-name Audit \
--source-identity Admin \
```

搭配 AssumeRoleWith SAML 使用來源身分識別

主體呼叫 `AssumeRoleWithSAML` 操作是使用 SAML 類型的聯合進行身分驗證。此作業會傳回一組可用來存取 AWS 資源的暫時認證。如需有關使用 SAML 型聯合進行存取的詳細資訊，請參 AWS Management Console 閱。啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console如需有關 AWS CLI 或 AWS API 存取的詳細資訊，請參閱SAML 2.0 聯合身分。如需為您的 Active Directory 使用者設定 SAML 聯盟的教學課程，請參閱安全性部落格中的使用中目錄AWS 同盟服務 (ADFS) 的同盟驗證。 AWS

身為系統管理員，您可以允許公司目錄的成員 AWS 使用此 AWS STS `AssumeRoleWithSAML`作業聯合。若要執行此操作，您必須完成下列任務：

1. 在組織中設定 SAML 提供者。
2. 在 IAM 中建立 SAML 提供者
3. 在中設定同盟使用者 AWS 的角色及其權限。
4. 完成配置 SAML IdP 並為 SAML 身分驗證回應建立聲明。

若要設定來源身分的 SAML 屬性，請包含 `Attribute` 元素與設定為 `https://aws.amazon.com/SAML/Attributes/SourceIdentity` 的 `Name` 屬性。使用 `AttributeValue` 元素指定來源身分的值。例如，假設您希望將下列身分屬性作為來源身分傳遞。

```
SourceIdentity:DiegoRamirez
```

如要傳遞此屬性，請在您的 SAML 聲明中包含下列元素。

Example SAML 聲明的程式碼片段範例

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
<AttributeValue>DiegoRamirez</AttributeValue>
</Attribute>
```

使用來源身分 AssumeRoleWithWebIdentity

調用`AssumeRoleWithWebIdentity`操作的主體使用 OpenID Connect（OIDC）兼容的聯合進行身份驗證。此操作會傳回一組暫時憑證，讓您用來存取 AWS 資源。如需使用 OIDC 同盟進行存取的詳細資訊，請參 AWS Management Console 閱。OIDC 聯盟

若要從 OpenID Connect (OIDC) 傳遞來源身分，您必須在 JSON Web 權杖 (JWT) 中包含來源身分。在您提交 `AssumeRoleWithWebIdentity` 請求時，您必須在權杖中的 `https://`

aws.amazon.com/ source_identity 命名空間內包含來源身分。若要進一步了解 OIDC 權杖和要求，請參閱《Amazon Cognito 開發人員指南》中的搭配使用者集區使用權杖。

例如，以下解碼後的 JWT 是搭配 Admin 來源身分呼叫 AssumeRoleWithWebIdentity 的權杖。

Example 解碼後的 JSON Web 權杖範例

```
{
    "sub": "johndoe",
    "aud": "ac_oic_client",
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/source_identity":"Admin"
}
```

使用來源身分資訊控制存取

初始設定來源識別時，sts: 索SourceIdentity引鍵會出現在要求中。設置源身份後，aws: SourceIdentity 密鑰存在於角色會話期間發出的所有後續請求中。身為管理員，您可以撰寫原則來授與條件式授權，以根據來源識別屬性的存在或值來執行 AWS 動作。

想像一下，您想要求開發人員設定來源身分識別，以擔任具有寫入生產關鍵 AWS 資源之權限的關鍵角色。還要想像一下，您授予 AWS 訪問您的員工身份使用AssumeRoleWithSAML. 您只希望資深開發人員 Saanvi 和 Diego 能夠存取角色，因此您為該角色建立下列信任政策。

Example 來源身分的角色信任政策範例 (SAML)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "SAMLProviderAssumeRoleWithSAML",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
      },
      "Action": [
```

```
        "sts:AssumeRoleWithSAML"
      ],
      "Condition": {
        "StringEquals": {
          "SAML:aud": "https://signin.aws.amazon.com/saml"
        }
      }
    },
    {
      "Sid": "SetSourceIdentitySrEngs",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
      },
      "Action": [
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
}
```

信任政策包含 `sts:SourceIdentity` 的條件，需要 Saanvi 或 Diego 的來源身分，才能擔任關鍵角色。

或者，如果您使用 OIDC 提供者進行聯合，且使用者經過驗證`AssumeRoleWithWebIdentity`，則您的角色信任原則可能如下所示。

Example 來源身分的角色信任政策範例 (OIDC 提供者)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/server.example.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "server.example.com:aud": "oidc-audience-id"
        },
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
}
```

角色鏈結和跨帳戶需求

假設您想允許已擔任 CriticalRole 的使用者在另一個帳戶中擔任 CriticalRole_2。為擔任 CriticalRole 而取得的角色工作階段憑證用於將角色鏈結至另一個帳戶中的第二個角色 CriticalRole_2。角色是跨帳戶界限擔任。因此，必須在 CriticalRole 的許可政策和 CriticalRole_2 的角色信任政策中授予 sts:SetSourceIdentity 許可。

Example 範例權限原則 CriticalRole

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleAndSetSourceIdentity",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Resource": "arn:aws:iam::222222222222:role/CriticalRole_2"
```

```
    }
  ]
}
```

為了確保跨帳戶界限設定來源身分，以下角色信任政策僅信任 CriticalRole 的角色主體來設定來源身分。

Example CriticalRole_2 上的範例角色信任原則

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/CriticalRole"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi","Diego"]
        }
      }
    }
  ]
}
```

使用者使用從假設取得的角色工作階段認證進行下列呼叫 CriticalRole。來源識別是在假設期間設定的 CriticalRole，因此不需要再次明確設定。如果使用者嘗試設定的來源身分與擔任 CriticalRole 時設定的值不同，則擔任角色請求將會遭到拒絕。

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::222222222222:role/CriticalRole_2 \
--role-session-name Audit \
```

當呼叫主體擔任角色時，請求中的來源身分會從第一個擔任角色工作階段開始一直存在。因此，aws:SourceIdentity 和 sts:SourceIdentity 索引鍵均會出現在請求內容中。

檢視來源身分 CloudTrail

您可以使用 CloudTrail 來檢視為擔任角色或同盟使用者所做的要求。您也可以檢視角色或使用者請求，以在 AWS中採取行動。 CloudTrail 記錄檔包含針對假定角色或同盟使用者工作階段設定之來源識別的相關資訊。如需更多資訊，請參閱使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail

例如，假設使用者提出 AWS STS AssumeRole要求，並設定來源身分識別。您可以在 CloudTrail 日誌中的requestParameters密鑰中找到sourceIdentity信息。

Example 記錄檔中的範例 requestParameters 區段 AWS CloudTrail

```
"eventVersion": "1.05",
    "userIdentity": {
        "type": "AWSAccount",
        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
        "accountId": "111122223333"
    },
    "eventTime": "2020-04-02T18:20:53Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.64",
    "userAgent": "aws-cli/1.16.96 Python/3.6.0 Windows/10 botocore/1.12.86",
    "requestParameters": {
        "roleArn": "arn:aws:iam::123456789012:role/DevRole",
        "roleSessionName": "Dev1",
        "sourceIdentity": "source-identity-value-set"
    }
```

如果使用者使用假定的角色工作階段來執行動作，則來源識別資訊會出現在 CloudTrail 記錄檔中的userIdentity金鑰中。

Example 記錄檔中的 userIdentity 金鑰範例 AWS CloudTrail

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAJ45Q7YFFAREXAMPLE:Dev1",
```

```
    "arn": "arn:aws:sts::123456789012:assumed-role/DevRole/Dev1",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAJ45Q7YFFAREXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/DevRole",
        "accountId": "123456789012",
        "userName": "DevRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-02-21T23:46:28Z"
      },
      "sourceIdentity": "source-identity-value-present"
    }
  }
}
```

若要查看 CloudTrail 記錄檔中的 AWS STS API 事件範例，請參閱 CloudTrail記錄檔中的 IAM API 事件範例。若要取得有關 CloudTrail 記錄檔中包含的資訊的詳細資訊，請參閱《AWS CloudTrail 使用指南》中的〈CloudTrail 事件參考〉。

GetFederationToken 的許可

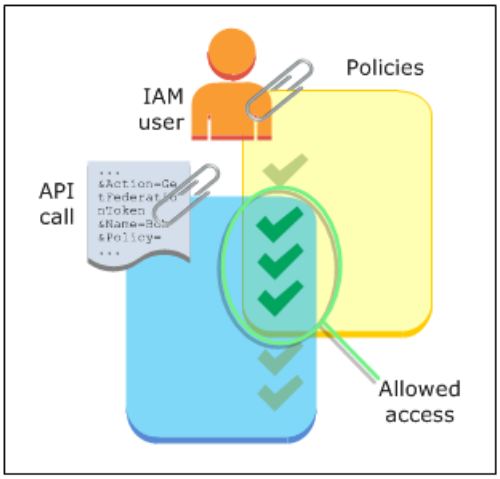
此 `GetFederationToken` 操作是由 IAM 使用者所呼叫並會傳回該使用者的臨時憑證。此操作會聯合使用者。指派給聯合身分使用者的許可在以下其中兩個地方定義：

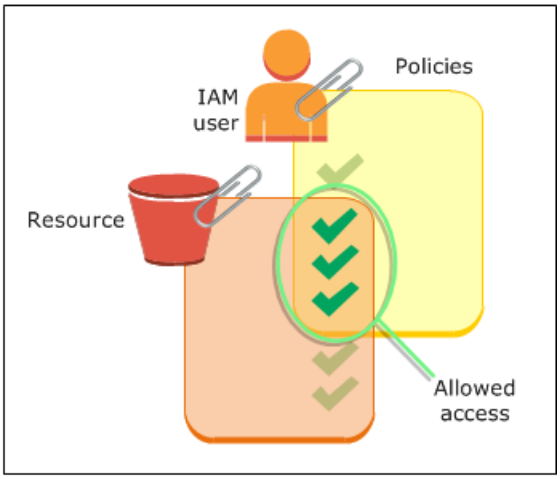
- 該工作階段政策會以 `GetFederationToken` API 呼叫的參數傳遞。(這是最常見的)。
- 資源為基礎的政策，而聯合身分使用者明確名稱在 `Principal` 元素政策。(這是最少見的)。

工作階段政策是一種進階政策，且您會在以程式設計方式建立臨時工作階段時，以參數方式傳遞此政策。建立聯合身分使用者工作階段並傳遞工作階段政策時，所產生工作階段的許可會是 使用者的身分類型政策和工作階段政策的交集。您無法使用工作階段政策來授予超出即將聯合使用者之以身分為基礎政策所允許的許可。

在大部分情況下，如果您未通過 `GetFederationToken` API 呼叫的政策，則所產生的暫時安全憑證沒有許可。不過，以資源為基礎的政策可提供該工作階段的額外許可。您可以使用以資源為基礎的政策存取資源，該政策會將您的工作階段指定為允許的主體。

下圖顯示政策互動的視覺化呈現，以判斷對 GetFederationToken 呼叫傳回的暫時安全憑證的許可。





範例：使用指派權限 GetFederationToken

您可以將 GetFederationToken API 動作與不同類型的政策一起使用。以下是幾個範例。

連接至 IAM 使用者的政策

在這個範例中，您有一個以瀏覽器為基礎的用戶端應用程式，它依賴於兩個後端的 Web 服務。一個後端服務是您自己的身分驗證伺服器，使用您自己的身分系統來驗證用戶端應用程式。另一個後端服務是 AWS 服務，可提供一些用戶端應用程式的功能。用戶端應用程式由您的伺服器進行身分驗證，您的伺服器會建立或擷取適當的許可政策。然後，您的伺服器呼叫 GetFederationToken API 以取得暫時安全憑證，並將這些憑證傳回給用戶端應用程式。然後，用戶端應用程式可以使用臨時安全登入資料直接向 AWS 服務發出要求。此架構可讓用戶端應用程式在不內嵌長期 AWS 認證的情況下 AWS 提出要求。

您的身分驗證伺服器使用名為 token-app 之 IAM 使用者的長期安全憑證來呼叫 GetFederationToken API。但長期 IAM 使用者憑證會保持在伺服器上且絕對不會發佈到用戶端。以下範例政策連接到 token-app IAM 使用者，並定義聯合身分使用者 (用戶端) 將需要的最廣泛的許可集。請注意，身分驗證服務需要 sts:GetFederationToken 許可才能取得聯合身分使用者的暫時安全憑證。

> **Note**
>
> AWS 提供了一個示例 Java 應用程序來實現此目的，您可以在此處下載：用於身份註冊的令牌自動售貨機-示例 Java Web 應用程序。

Example 連接到呼叫 **GetFederationToken** 的 IAM 使用者 **token-app** 的政策範例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:GetFederationToken",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:ReceiveMessage",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:ListSubscriptions",
      "Resource": "*"
    }
  ]
}
```

上述政策會將多個許可授予 IAM 使用者。不過，單靠此政策不會將任何許可授予聯合身分使用者。如果此 IAM 使用者呼叫 GetFederationToken，並且未將政策作為 API 呼叫的參數傳遞，則產生的聯合身分使用者將不具有有效許可。

工作階段政策作為參數傳遞

確保為聯合身分使用者指派適當許可的最常見方法，是在 GetFederationToken API 呼叫中傳遞工作階段政策。擴展前面的範例，假設使用 IAM 使用者 token-app 憑證來呼叫 GetFederationToken。假設以下工作階段政策做為 API 呼叫的參數傳遞。所產生的聯合身分

使用者具有列出名為 productionapp 的 Amazon S3 儲存貯體內容的許可。該使用者無法對 productionapp 儲存貯體中的項目執行 Amazon S3 GetObject、PutObject 和 DeleteObject 動作。

聯合身分使用者會獲派這些許可，因為這些許可是 IAM 使用者政策和您傳遞之工作階段政策的交集。

聯合身分使用者無法在 Amazon SNS、Amazon SQS、Amazon DynamoDB 或任何 S3 儲存貯體中執行動作，但 productionapp 除外。即使這些許可已授予給與 GetFederationToken 呼叫相關聯的 IAM 使用者，這些動作仍然會遭到拒絕。

Example 工作階段政策作為 **GetFederationToken** API 呼叫的參數傳遞範例

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::productionapp"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::productionapp/*"]
    }
  ]
}
```

資源型政策

某些 AWS 資源支援以資源為基礎的政策，而且這些原則提供另一種機制，可直接將權限授與同盟使用者。只有部分 AWS 服務支援資源型政策。例如，Amazon S3 具有儲存貯體，Amazon SNS 具有主題，並且 Amazon SQS 具有可以連接政策的佇列。如需有關支援以資源為基礎的政策的所有服務的清單，請參閱 AWS 與 IAM 搭配使用的服務 並查看表格中的「以資源為基礎的政策」資料列。您可以使用以資源為基礎的政策來將許可直接指派至聯合身分使用者。若要執行此操作，需在以資源為基礎之政策的 Principal 元素中指定聯合身分使用者的 Amazon Resource Name (ARN)。以下範例使用名為 productionapp 的 S3 儲存貯體說明此範例並擴展前面的範例。

以下以資源為基礎的政策已連接到儲存貯體。此儲存貯體政策可讓名為 Carol 的聯合身分使用者存取儲存貯體。當先前所述的範例政策連接到 token-app IAM 使用者時，名為 Carol 的聯合身分使用者便具有對名為 productionapp 的儲存貯體執行 s3:GetObject、s3:PutObject 和 s3:DeleteObject 動作的許可。即使沒有工作階段政策做為 GetFederationToken API 呼叫的參數傳遞時，也是如此。這是因為在這種情況下，名為 Carol 的聯合身分使用者已透過以下以資源為基礎的政策明確授予許可。

請記住，只有當將這些許可明確授予 IAM 使用者和聯合身分使用者時，才會授予聯合身分使用者許可。在以資源為基礎的政策中，運用 Principal 元素來明確命名聯合身分使用者，也可以授予這些許可 (在帳戶內)，如下面範例所示。

Example 允許存取聯合身分使用者的儲存貯體政策範例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Principal": {"AWS": "arn:aws:sts::account-id:federated-user/Carol"},
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::productionapp/*"]
  }
}
```

如需有關如何評估政策的詳細資訊，請參閱政策評估邏輯。

GetSessionToken 的許可

呼叫 GetSessionToken API 操作或 get-session-token CLI 命令的主要時間是，當使用者必須透過多重要素驗證 (MFA) 進行身分驗證時。可以編寫只允許特定操作的一項政策，且只有在這些操作是由經過 MFA 身分驗證的使用者提出的請求時，才允許執行。為成功通過 MFA 身分驗證檢查，使用者必須先呼叫 GetSessionToken，並且包含可選的SerialNumber 和 TokenCode 參數。如果使用者成功通過 MFA 裝置身分驗證，GetSessionToken API 操作傳回的憑證包含 MFA 內容。此內容表示使用者通過 MFA 身分驗證，並且獲得 API 操作授權，其需要 MFA 身分驗證。

所需的權限 GetSessionToken

使用者要取得工作階段權杖並不需要許可。GetSessionToken 操作的目的是使用 MFA 驗證使用者的身分。您不能使用政策來控制驗證操作。

若要授與執行大部分 AWS 作業的權限，您可以將具有相同名稱的動作新增至原則。例如，若要建立使用者，您必須使用 CreateUser API 操作、create-user CLI 命令或 AWS Management Console。若要執行這些操作，您必須擁有一個政策，其可讓您存取 CreateUser 動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateUser",
            "Resource": "*"
        }
    ]
}
```

您可以在您的政策中包含 GetSessionToken 動作，但不會影響使用者執行 GetSessionToken 操作的能力。

GetSessionToken 授予的許可

如果使用 IAM 使用者的憑證呼叫 GetSessionToken，臨時安全性憑證與 IAM 使用者會有相同的許可。同樣地，如果GetSessionToken使用 AWS 帳戶根使用者 認證呼叫，則臨時安全性認證具有 root 使用者權限。

> **Note**
>
> 我們建議您不要使用根使用者憑證呼叫 GetSessionToken。相反的，請遵循我們的最佳實務，並建立具有所需許可的 IAM 使用者。然後使用這些 IAM 使用者進行日常互動 AWS。

當您呼叫 GetSessionToken 時您獲得的臨時憑證有下列功能和限制：

- 您可以使用認證來存取，方 AWS Management Console 法是將認證傳遞至同盟單一登入端點，位於https://signin.aws.amazon.com/federation。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

- 您無法使用憑證來呼叫 IAM 或 AWS STS API 操作。您可以使用它們來呼叫其他 AWS 服務的 API 作業。

將此 API 操作和其限制及功能與在 比較 AWS STS API 操作 建立臨時安全性憑證的 API 操作相較

如需有關使用 GetSessionToken 的受 MFA 保護之 API 存取的詳細資訊，請參閱 設定受 MFA 保護的 API 存取。

停用臨時安全性憑證的許可

暫時安全憑證在過期之前持續有效。這些憑證在指定的持續時間內有效，從 900 秒 (15 分鐘) 至最長 129,600 秒 (36 小時)。預設工作階段持續時間為 43,200 秒 (12 小時)。您可以撤銷這些憑證，但也必須變更角色的許可，以停止使用遭洩漏的憑證進行惡意帳戶活動。每次使用臨時安全登入資料提出 AWS 要求時，都會評估指派給臨時安全登入資料的權限。從認證移除所有權限後，使用這些權限的 AWS 要求就會失敗。

政策更新可能需要幾分鐘的時間才會生效。撤銷角色的暫時安全憑證以強制所有擔任該角色的使用者重新驗證並請求新的憑證。

您無法變更 AWS 帳戶根使用者. 同樣地，以根使用者身分登入時，您無法更改呼叫 GetFederationToken 或 GetSessionToken 建立的暫時安全憑證許可。因此，我們建議您不要以根使用者身分呼叫 GetFederationToken 或 GetSessionToken。

> **Important**
>
> 您無法編輯從 IAM 身分中心權限集建立的 IAM 中的角色。您必須在 IAM 身分中撤銷使用者的作用中權限集工作階段。如需詳細資訊，請參閱 IAM 身分中心使用者指南中的撤銷由權限集建立的作用中 IAM 角色工作階段。

主題

- 拒絕存取與角色關聯的所有工作階段
- 拒絕存取特定工作階段
- 使用條件內容索引鍵拒絕使用者工作階段
- 使用以資源為基礎的政策拒絕工作階段使用者

拒絕存取與角色關聯的所有工作階段

當您擔心被以下身分進行可疑存取時，請使用此方法：

- 使用跨帳戶存取權的其他帳戶中的主體
- 具有存取帳戶 AWS 資源之權限的外部使用者身分識別
- 已在 OIDC 提供商的移動或 Web 應用程序中進行身份驗證的用戶

此程序拒絕向所有有權擔任角色的使用者授予許可。

若要變更或移除為呼叫 `AssumeRole`、`AssumeRoleWithSAML`、`AssumeRoleWithWebIdentity`、`GetFederationToken` 或 `GetSessionToken` 而取得的暫時安全憑證分配的許可，您可以編輯或刪除為角色定義許可的許可政策。

> **Important**
>
> 如果存在允許主體存取的以資源為基礎的政策，您還必須為該資源新增明確拒絕。如需詳細資訊，請參閱 使用以資源為基礎的政策拒絕工作階段使用者。

1. 登入 AWS Management Console 並開啟 IAM 主控台。
2. 在導覽窗格中，選擇要編輯的角色名稱。您可以使用搜尋方塊來篩選清單。
3. 選取相關政策。
4. 選擇許可索引標籤標籤。
5. 選擇 JSON 標籤並更新政策以拒絕所有資源和動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*"
    }
  ]
}
```

6. 在 Review (檢視) 頁面上，查看政策 Summary (摘要)，然後選擇 Save changes (儲存政策) 以儲存您的工作。

在更新政策時，所做出的變更將影響與該角色關聯的所有暫時安全憑證的許可，包括在更改該角色的許可政策之前發行的憑證。更新政策之後，您可以撤銷角色的暫時安全憑證以立即撤銷角色所發行憑證的所有許可。

拒絕存取特定工作階段

當您使用全部拒絕政策更新 IdP 可擔任的角色或完全刪除該角色時，有權存取該角色的所有使用者都會中斷。您可以根據 `Principal` 元素拒絕存取，而不影響與該角色關聯之所有其他工作階段的許可。

您可以使用條件內容索引鍵或以資源為基礎的政策拒絕 `Principal` 的許可。

> **Tip**
>
> 您可以使用記錄檔尋找聯合身分使用 AWS CloudTrail 者的 ARN。如需詳細資訊，請參閱如何使用輕鬆識別聯合身分使用 AWS CloudTrail者。

使用條件內容索引鍵拒絕使用者工作階段

您可以在想要拒絕存取特定暫時安全憑證工作階段的情況下使用條件內容索引鍵，而不影響建立憑證的 IAM 使用者或角色的許可。

如需條件內容索引鍵的詳細資訊，請參閱 AWS 全域條件內容索引鍵。

> **Note**
>
> 如果存在允許主體存取的以資源為基礎的政策，在完成這些步驟後，您還必須在以資源為基礎的政策上新增明確拒絕陳述式。

更新政策之後，您可以撤銷角色的暫時安全憑證以立即撤銷所有發行的憑證。

AWS：PrincipalArn

您可以使用條件內容索引鍵 AWS：PrincipalArn 拒絕存取特定的主體 ARN。您可以透過在政策的「條件」元素中指定暫時安全憑證關聯的 IAM 使用者、角色或聯合身分使用者的唯一識別符 (ID)，來執行此操作。

1. 在 IAM 主控台導覽窗格中，選擇要編輯的角色名稱。您可以使用搜尋方塊來篩選清單。
2. 選取相關政策。
3. 選擇許可索引標籤標籤。
4. 選擇 JSON 標籤並為主體 ARN 新增拒絕陳述式，如下列範例所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:role/ROLENAME",
            "arn:aws:iam::222222222222:user/USERNAME",
            "arn:aws:sts::222222222222:federated-user/USERNAME"
          ]
        }
      }
    }
  ]
}
```

5. 在 Review (檢視) 頁面上，查看政策 Summary (摘要)，然後選擇 Save changes (儲存政策) 以儲存您的工作。

aws:userid

您可以使用條件內容索引鍵 aws:userid 拒絕存取與 IAM 使用者或角色關聯的所有或特定暫時安全憑證工作階段。您可以透過在政策的 Condition 元素中指定暫時安全憑證關聯的 IAM 使用者、角色或聯合身分使用者的唯一識別符 (ID)，來執行此操作。

下列政策顯示如何使用條件內容索引鍵 aws:userid 拒絕存取暫時安全憑證工作階段的範例。

- AIDAXUSER1 代表 IAM 使用者的唯一識別符。指定 IAM 使用者的唯一識別符作為內容索引鍵 aws:userid 的值將拒絕與 IAM 使用者關聯的所有工作階段。
- AROAXROLE1 代表 IAM 角色的唯一識別符。指定 IAM 角色的唯一識別符作為內容索引鍵 aws:userid 的值將拒絕與角色關聯的所有工作階段。

- AROAXROLE2 代表擔任角色工作階段的唯一識別符。在假設角色唯一識別碼的 caller-specified-role-session-name 部分中，您可以指定角色工作階段名稱或萬用字元 (如果使用 StringLike條件運算子)。如果您指定角色工作階段名稱，則它會拒絕具名角色工作階段，而不會影響建立憑證的角色的許可。如果您為角色工作階段名稱指定萬用字元，則它會拒絕與角色關聯的所有工作階段。
- account-id:<federated-user-caller-specified-name> 代表聯合身分使用者工作階段的唯一識別符。聯合身分使用者是由呼叫 GetFederationToken API 的 IAM 使用者所建立。如果您為聯合身分使用者指定唯一識別符，它會拒絕具名聯合身分使用者工作階段，而不會影響建立憑證的角色的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:userId": [
            "AIDAXUSER1",
            "AROAXROLE1",
            "AROAXROLE2:<caller-specified-role-session-name>",
            "account-id:<federated-user-caller-specified-name>"
          ]
        }
      }
    }
  ]
}
```

如需主體索引鍵值的特定範例，請參閱 主體索引鍵值。如需 IAM 唯一識別符的資訊，請參閱 唯一識別碼。

使用以資源為基礎的政策拒絕工作階段使用者

如果主體 ARN 也包含在任何以資源為基礎的政策中，您還必須根據以資源為基礎的政策的 Principal 元素中特定使用者的 principalId 或 sourceIdentity 值撤銷存取權。如果您僅更新角色的許可政策，使用者仍然可以執行以資源為基礎的政策中允許的動作。

1. 請參閱 AWS 與 IAM 搭配使用的服務 以查看服務是否支援以資源為基礎的政策。

2. 登入 AWS Management Console 並開啟服務的主控台。每個服務在主控台中都有不同的位置用於附加政策。
3. 編輯政策陳述式以指定憑證的識別資訊：
   a. 在 `Principal` 中，輸入要拒絕的憑證的 ARN。
   b. 在 `Effect` 中，輸入 "Deny"。
   c. 在 `Action` 中，輸入服務命名空間和要拒絕的動作名稱。若要拒絕所有動作，請使用萬用字元 (*)。例如："s3:*"。
   d. 在 `Resource` 中，輸入目標資源的 ARN。例如："arn:aws:s3:::EXAMPLE-BUCKET"。

```
{
"Version": "2012-10-17",
  "Statement": {
    "Principal": [
            "arn:aws:iam::222222222222:role/ROLENAME",
            "arn:aws:iam::222222222222:user/USERNAME",
            "arn:aws:sts::222222222222:federated-user/USERNAME"
    ],
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::EXAMPLE-BUCKET"
  }
}
```

4. 儲存您的工作。

授予許可以建立暫時性安全憑證

在預設情況下，IAM 使用者沒有許可來建立聯合身分使用者和角色的暫時性安全憑證。您必須使用政策提供您的使用者這些許可。雖然您可以直接授予許可給使用者，我們強烈建議您將許可授予群組。這可讓許可的管理更輕鬆。當有人不再需要執行與許可相關的任務時，您只需從群組中移除許可。如果其別人需要執行該任務，將它們新增到群組以授予許可。

若要授予 IAM 群組許可以建立聯合身分使用者或角色的臨時安全性憑證，您可連接會授予以下一個或兩個權限的政策：

- 對於要存取 IAM 角色的聯合身分使用者，將存取權授予 AWS STS `AssumeRole`。
- 對於不需要角色的同盟使用者，請授與存取 AWS STS `GetFederationToken`權。

如需有關 AssumeRole 和 GetFederationToken API 操作間差異的詳細資訊，請參閱 請求暫時性安全憑證。

IAM 使用者也可以呼叫 GetSessionToken 建立臨時安全憑證。使用者呼叫 GetSessionToken 不需要許可。此操作的目的是使用 MFA 驗證使用者的身分。您不能使用政策來控制身分驗證。這表示您無法避免 IAM 使用者呼叫 GetSessionToken 來建立臨時憑證。

Example 授予許可以擔任角色的政策範例

下列範例原則授與在中呼叫AssumeRoleUpdateApp角色的權限 AWS 帳戶 123123123123。當使用 AssumeRole 時，代表聯合身分使用者建立安全憑證的使用者 (或應用程式)，無法委派角色許可政策中未指定的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"
  }]
}
```

Example 授予許可以建立聯合身分使用者的暫時性安全憑證的政策範例

下列範例政策會授予許可以存取 GetFederationToken。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": "*"
  }]
}
```

Important

當您提供 IAM 使用者許可，建立具 GetFederationToken 之聯合身分使用者的暫時性安全憑證時，請留意，這會讓那些使用者委派自己的許可。如需有關跨 IAM 使用者委派許可的詳細

資訊 AWS 帳戶，請參閱委派存取權限的政策範例。如需有關如何控制暫時性安全憑證的許可的詳細資訊，請參閱控制臨時安全安全憑證的許可。

Example 授予使用者有限許可以建立聯合身分使用者的暫時性安全憑證的政策範例

當您讓 IAM 使用者呼叫 GetFederationToken，最佳實務是限制 IAM 使用者可以委派的許可。例如，以下政策顯示如何讓 IAM 使用者僅建立適用於聯合身分使用者的暫時性安全憑證，該使用者的名稱以管理員為開頭。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": ["arn:aws:sts::123456789012:federated-user/Manager*"]
  }]
}
```

授與使用識別感知主控台工作階段的權限

身分識別感知主控台工作階段可讓 AWS IAM Identity Center 使用者和工作階段 ID 納入使用者登入時的 AWS 主控台工作階段中。例如，Amazon Q Developer Pro使用身分識別感知主控台工作階段來個人化服務體驗。如需有關身分識別感知主控台工作階段的詳細資訊，請參閱《使用手冊》中的啟用身分識別感知主控台AWS IAM Identity Center 如需 Amazon Q 開發人員設定的相關資訊，請參閱 Amazon Q 開發人員使用指南中的設定 Amazon Q 開發人員。

若要讓使用者可以使用身分識別感知主控台工作階段，您必須使用以身分為基礎的政策，將代表其主控台工作階段之資源的`sts:SetContext`權限授與 IAM 主體。

> **Important**
>
> 依預設，使用者沒有權限為其身分識別感知主控台工作階段設定內容。若要允許此操作，您必須在以下政策範例所示的身分型政策中授予 IAM 主體`sts:SetContext`權限。

下列範例以身分識別為基礎的政策授與 IAM 主體的`sts:SetContext`權限，允許主體為自己的主控台工作階段設定身分識別感知主控台工作階段內容。 AWS 原則資源代表呼叫者的 AWS 工作階段。`arn:aws:sts::`*`account-id`*`:self`如果*在多個帳戶中部署相同權限政策 (例如，使用 IAM Identity Center 權限集部署此政策時)，則可以使用萬用字元取代 account-id ARN 區段。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sts:SetContext",
      "Resource": "arn:aws:sts::account-id:self"
    }
  ]
}
```

AWS STS 在一個管理 AWS 區域

預設情況下， AWS Security Token Service (AWS STS) 可作為全域服務使用，且所有 AWS STS 要求都會傳送至位於的單一端點`https://sts.amazonaws.com`。 AWS 建議使用區域 AWS STS 端點而非全域端點，以減少延遲、建置備援，並提高工作階段 Token 有效性。

- 減少延遲 — 透過 AWS STS 呼叫地理位置較接近服務和應用程式的端點，您可以以較低的延遲和更短的回應時間存取 AWS STS 服務。
- 內建備援 – 您可以將工作負載內失敗的影響限制為有限數量的元件，且具有可預測的影響限制範圍。使用地區 AWS STS 端點可讓您將組件的範圍與會話令牌的範圍對齊。如需有關此可靠性支柱的詳細資訊，請參閱 AWS Well-Architected Framework 中的使用故障隔離來保護工作負載。
- 增加會話令牌有效性 — 來自區域 AWS STS 端點的會話令牌在所有有效 AWS 區域。來自全域 STS 端點的工作階段權杖僅在預設啟 AWS 區域 用的情況下才有效。如果您打算為您的帳戶啟用新的區域，則可以使用區域 AWS STS 端點中的會話令牌。如果您選擇使用全域端點，則必須變更全域端點 AWS STS 工作階段權杖的區域相容性。這樣做可以確保令牌在所有有效 AWS 區域。

管理全域端點的工作階段權杖

默認 AWS 區域 情況下，大多數都啟用了 AWS 服務 所有操作。這些區域會自動啟用，以便搭配使用 AWS STS。亞太區域 (香港) 之類的某些區域必須手動啟用。若要深入瞭解啟用和停用 AWS 區域，請參閱《AWS Account Management 參考指南》中的「指定 AWS 區域 您的帳戶可以使用的項目」。當您啟用這些 AWS 區域時，它們會自動啟用以搭配使用 AWS STS。您無法為已停用的區域啟動 AWS STS 端點。在所有中都有效的會話令牌 AWS 區域 包含的字符比在默認情況下啟用的區域中有效的令牌更多。變更此設定可能會影響暫時存放權杖的現有系統。

您可以使用 AWS Management Console、 AWS CLI或 AWS API 變更此設定。

變更全域端點工作階段權杖的區域相容性 (主控台)

1. 以具有 IAM 管理任務執行許可的根使用者或使用者身分來登入。若要變更工作階段權杖的相容性，您必須擁有允許 `iam:SetSecurityTokenServicePreferences` 動作的政策。
2. 開啟 IAM 主控台。在導覽窗格中，選擇 Account settings (帳戶設定)。
3. 在 Security Token Service (STS) (安全權杖服務 (STS)) 區段的 Session Tokens from the STS endpoints (來自 STS 端點的工作階段權杖) 下方。Global endpoint (全域端點) 表示 `Valid only in AWS ## enabled by default`。選擇 Change (變更)。
4. 在 [變更區域相容性] 對話方塊中，選取 [全部] AWS 區域。接著選擇 Save changes (儲存變更)。

> **Note**
>
> 在所有中都有效的會話令牌 AWS 區域 包含的字符比在默認情況下啟用的區域中有效的令牌更多。變更此設定可能會影響暫時存放權杖的現有系統。

變更全域端點工作階段權杖的區域相容性 (AWS CLI)

設定工作階段字符版本。版本 1 令牌僅在默認情況下 AWS 區域 可用的令牌有效。這些權杖不適用於手動啟用的區域，例如亞太區域 (香港)。版本 2 權杖在所有區域都有效。不過，版本 2 權杖包含較多字元且可能會影響暫時存放權杖的系統。

- `aws iam set-security-token-service-preferences`

變更全域端點工作階段權杖的區域相容性 (AWS API)

設定工作階段字符版本。版本 1 令牌僅在默認情況下 AWS 區域 可用的令牌有效。這些權杖不適用於手動啟用的區域，例如亞太區域 (香港)。版本 2 權杖在所有區域都有效。不過，版本 2 權杖包含較多字元且可能會影響暫時存放權杖的系統。

- `SetSecurityTokenServicePreferences`

在中啟用和停 AWS STS 用 AWS 區域

當您啟用某個區域的 STS 端點時， AWS STS 可以向您帳戶中提出 AWS STS 要求的使用者和角色發出臨時認證。然後，您就可以在預設啟用或手動啟用的任何區域中使用這些憑證。針對預設情況下便啟用的區域，您必須在產生臨時憑證的帳戶中啟用區域性 STS 端點。提出請求時使用者登入的是相同帳

戶或不同帳戶並不重要。針對需手動啟用的區域，您必須在提出請求的帳戶以及產生臨時憑證的帳戶中啟用區域。

例如，假設帳戶 A 中的使用者想要將 `sts:AssumeRole` API 要求傳送至 AWS STS 區域端點`https://sts.ap-east-1.amazonaws.com`。該請求用於帳戶 B 中名為 `Developer` 的角色的暫時憑證。由於請求是為帳戶 B 中的實體建立憑證，因此帳戶 B 必須啟用 `ap-east-1` 區域。來自帳戶A (或任何其他帳戶) 的使用者可以呼叫 `ap-east-1` AWS STS 端點以請求帳戶 B 的憑證，無論該帳戶中是否已啟用該區域。

> **Note**
>
> 在該帳戶中使用臨時性憑證的每個人都可使用作用中的區域。若要控制哪些 IAM 使用者或角色可以存取區域，請在許可政策中使用 `aws:RequestedRegion` 條件金鑰。

在預設啟用 AWS STS 的區域 (主控台) 中啟用或停用

1. 以具有 IAM 管理任務執行許可的根使用者或使用者身分來登入。
2. 開啟 IAM 主控台，然後在導覽窗格中選擇 Account settings (帳戶設定)。
3. 在 Security Token Service (STS) 的 Endpoints (端點) 區段中，找到您想要設定的區域，然後在 STS status (STS 狀態) 欄中選擇 Active (作用中) 或 Inactive (非作用中)。
4. 在開啟的對話方塊中，選擇 Activate (啟用) 或 Deactivate (停用)。

對於必須啟用的地區，我們會在您啟用「地區」時 AWS STS 自動啟用。啟用「地區」後， AWS STS 該地區始終處於作用中狀態，且您無法停用該區域。若要瞭解如何啟用預設為停用的區域，請參閱《AWS Account Management 參考指南》中 AWS 區域 的指定帳戶可以使用的區域。

撰寫程式碼以使用 AWS STS 區域

啟用區域後，您可以將 AWS STS API 呼叫導向至該區域。下列 Java 程式碼片段示範如何設定`AWSSecurityTokenService`物件以向歐洲 (Milan) (eu-south-1) 區域發出要求。

```
EndpointConfiguration regionEndpointConfig = new EndpointConfiguration("https://sts.eu-
south-1.amazonaws.com", "eu-south-1");
AWSSecurityTokenService stsRegionalClient =
 AWSSecurityTokenServiceClientBuilder.standard()
.withCredentials(credentials)
.withEndpointConfiguration(regionEndpointConfig)
```

```
.build();
```

AWS STS 建議您撥打地區端點的電話。若要瞭解如何手動啟用區域，請參閱《AWS Account Management 參考指南》中的「指定 AWS 區域 您的帳戶可以使用的項目」。

在這個範例中，第一行會執行個體化稱為 `regionEndpointConfig` 的 `EndpointConfiguration` 物件，並將端點和 AWS 區域 的 URL 做為參數傳遞。

要了解如何使用 SDK 的環境變量設置 AWS STS 區域端點，請參閱 AWS SDK 和工具參考指AWS STS 南中的AWS 區域化端點。

對於所有其他語言和程式設計環境組合，請參閱相關軟體開發套件的文件。

區域與端點

下表列出區域及其端點。它會指出哪些是預設啟用，以及哪些可以啟用或停用。

區域名稱	端點	預設為作用中	手動啟動/停用
--全球服務--	sts.amazonaws.com	是	否
美國東部 (俄亥俄)	sts.us-east-2.amazonaws.com	是	是
美國東部 (維吉尼亞北部)	sts.us-east-1.amazonaws.com	是	否
美國西部 (加利佛尼亞北部)	sts.us-west-1.amazonaws.com	是	是

區域名稱	端點	預設為作用中	手動啟動/停用
美國西部 (奧勒岡)	sts.us-west-2.amazonaws.com	 是	 是
Africa (Cape Town)	sts.af-south-1.amazonaws.com	 否[1]	 否
亞太區域 (香港)	sts.ap-east-1.amazonaws.com	 否[1]	 否
亞太區域 (海德拉巴)	sts.ap-south-2.amazonaws.com	 否[1]	 否
亞太區域 (雅加達)	sts.ap-southeast-3.amazonaws.com	 否[1]	 否
亞太區域 (墨爾本)	sts.ap-southeast-4.amazonaws.com	 否[1]	 否
亞太區域 (孟買)	sts.ap-south-1.amazonaws.com	 是	 是

區域名稱	端點	預設為作用中	手動啟動/停用
亞太區域 (大阪)	sts.ap-northeast-3.amazonaws.com	是	是
亞太區域 (首爾)	sts.ap-northeast-2.amazonaws.com	是	是
亞太區域 (新加坡)	sts.ap-southeast-1.amazonaws.com	是	是
亞太區域 (雪梨)	sts.ap-southeast-2.amazonaws.com	是	是
亞太區域 (東京)	sts.ap-northeast-1.amazonaws.com	是	是
加拿大 (中部)	sts.ca-central-1.amazonaws.com	是	是
加拿大西部 (卡加利)	sts.ca-west-1.amazonaws.com	是	是

區域名稱	端點	預設為作用中	手動啟動/停用
中國 (北京)	sts.cn-north-1.amazonaws.com.cn	 是[2]	 否
中國 (寧夏)	sts.cn-northwest-1.amazonaws.com.cn	 是[2]	 是
歐洲 (法蘭克福)	sts.eu-central-1.amazonaws.com	 是	 是
歐洲 (愛爾蘭)	sts.eu-west-1.amazonaws.com	 是	 是
歐洲 (倫敦)	sts.eu-west-2.amazonaws.com	 是	 是
歐洲 (米蘭)	sts.eu-south-1.amazonaws.com	 否[1]	 否
Europe (Paris)	sts.eu-west-3.amazonaws.com	 是	 是

區域名稱	端點	預設為作用中	手動啟動/停用
歐洲 (西班牙)	sts.eu-south-2.amazonaws.com	 否[1]	 否
歐洲 (斯德哥爾摩)	sts.eu-north-1.amazonaws.com	 是	 是
歐洲 (蘇黎世)	sts.eu-central-2.amazonaws.com	 否[1]	 否
以色列 (特拉維夫)	sts.il-central-1.amazonaws.com	 否[1]	 否
Middle East (Bahrain)	sts.me-south-1.amazonaws.com	 否[1]	 否
中東 (阿拉伯聯合大公國)	sts.me-central-1.amazonaws.com	 否[1]	 否
南美洲 (聖保羅)	sts.sa-east-1.amazonaws.com	 是	 是

[1] 您必須啟用區域才能使用它。這會自動啟動 AWS STS。您無法手動啟動或停用這些區域中的 AWS STS 。

[2] 要 AWS 在中國使用，您需要一個特定 AWS 於中國的帳戶和憑據。

AWS CloudTrail 和區域端點

對區域和全域端點的呼叫會記錄在 AWS CloudTrail 的 `tlsDetails` 欄位中。對區域端點的呼叫 (例如`us-east-2.amazonaws.com`) 會登 CloudTrail 入其適當的區域。對此全球端點 `sts.amazonaws.com` 的呼叫都會記錄為對全球服務的呼叫。全域 AWS STS 端點的事件會記錄到 us-east-1。

> **Note**
>
> 只有支援此欄位的服務可檢視 `tlsDetails`。請參閱AWS CloudTrail 使用者指南中 CloudTrail 的支援 TLS 詳細資料的服務
> 如需更多詳細資訊，請參閱 使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail。

使用持有人權杖

某些 AWS 服務需要您具有獲得 AWS STS 服務承載令牌的權限，然後才能以編程方式訪問其資源。這些服務支援的通訊協定會要求您使用持有人權杖，而不是使用傳統的 Signature 第 4 版簽署要求。當您執行 AWS CLI 或需要承載令牌的 AWS API 操作時， AWS 服務會代表您請求承載令牌。該服務會提供您權杖，接著您就可以使用該權杖在該服務中執行後續操作。

AWS STS 服務承載令牌包括來自原始主體驗證的信息，這些信息可能會影響您的權限。此資訊可能包括主體標籤、工作階段標籤和工作階段政策。權杖的存取金鑰 ID 會以 `ABIA` 字首開頭。這有助於您識別 CloudTrail日誌中使用服務承載令牌執行的操作。

> **Important**
>
> 持有人權杖只能用於對產生該權杖之服務的呼叫，以及產生該權杖的區域中。您無法在其他服務或區域中使用持有人權杖執行操作。

支持承載令牌的服務示例是 AWS CodeArtifact。您必須先呼叫`aws codeartifact get-authorization-token`作業，才能 AWS CodeArtifact 使用套件管理員 (例如 NPM、Maven 或 PIP)

進行互動。此操作返回一個承載令牌，您可以使用它來執行 AWS CodeArtifact 操作。或者，您也可以使用能完成相同操作並自動設定您用戶端的 `aws codeartifact login` 命令。

如果您在為您產生承載權杖的 AWS 服務中執行動作，則 IAM 政策中必須具有以下許可：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServiceBearerToken",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*"
        }
    ]
}
```

如需服務持有人權杖範例，請參閱 AWS CodeArtifact 使用者指南中的將以身分為基礎的政策用於 AWS CodeArtifact。

使用臨時憑證的應用程式範例

您可以使用 AWS Security Token Service (AWS STS) 建立並為受信任的使用者提供可控制資源存取權的臨時安全登入 AWS 資料。如需 AWS STS的相關資訊，請參閱 IAM 中的暫時安全憑證。若要瞭解如何使用 AWS STS 來管理臨時安全登入資料，您可以下載下列實作完整範例案例的範例應用程式：

- 啟用聯合以 AWS 使用視窗作用中目錄、ADFS 和 SAML 2.0。示範如何使用 Windows Active Directory (AD)、Active Directory Federation Services (ADFS) 2.0 和 SAML (安全性聲明標記語言) 2.0，將企業聯合存取權委派給 AWS 。
- 啟用自訂身分識別代理存取主 AWS 控台。示範如何建立啟用單一登入 (SSO) 的自訂聯合代理程式，使現有 Active Directory 使用者能夠登入 AWS Management Console。
- 如何使用單一登入到. AWS Management Console 。示範如何使用 Shibboleth 與 SAML 來提供使用者存取 AWS Management Console的單一登入 (SSO) 許可。

OIDC 聯盟的範例

下面的示例應用程序說明如何使用 OIDCFederation 與提供商，如 Login with Amazon, Amazon Cognito, Facebook, 或谷歌. 您可以從這些提供商交易身份驗證以獲取臨時 AWS 安全憑據以訪問 AWS 服務。

- Amazon Cognito 教學課程 — 我們建議您將 Amazon Cognito 與軟體開發套件搭配使用，以進行行動 AWS 開發。Amazon Cognito 是進行行動應用程式身分管理的最便捷的方式，並提供了同步和跨裝置身分驗證等連接功能。如需有關 Amazon Cognito 的詳細資訊，請參閱 Amplify 文件 中的 Amplify 身分驗證。

啟用自訂身分識別代理存取主 AWS 控台

您可以運用編寫和執行程式碼來建立 URL 以使登入到您組織網路的使用者能夠安全存取 AWS Management Console。該 URL 包含您從中 AWS 獲取的登錄令牌，用於對用戶進行身份驗證。AWS產生的主控台工作階段可能因為聯合而包含不同的 `AccessKeyId`。若要透過相關 CloudTrail 事件追蹤聯合登入的存取金鑰使用情況，請參閱使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail和AWS Management Console 登入事件。

> **Note**
>
> 如果您的組織使用與 SAML 相容的身分提供者 (IdP)，則無需編寫程式碼即可設定對主控台的存取許可。這適用於 Microsoft 的 Active Directory 聯合身分驗證服務或開放原始碼 Shibboleth 等提供者。如需詳細資訊，請參閱 啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console。

若要讓組織的使用者能夠存取 AWS Management Console，您可以建立執行下列步驟的自訂身分識別代理程式：

1. 確認您的本機身分系統已對使用者進行身分驗證。
2. 呼叫 AWS Security Token Service (AWS STS) AssumeRole(建議) 或 GetFederationTokenAPI 作業，以取得使用者的臨時安全登入資料。要了解在擔任角色時使用的各種方法，請參閱 使用 IAM 角色。若要了解如何在取得您安全憑證時傳遞選用工作階段標籤，請參閱 傳遞工作階段標籤 AWS STS。
   - 如果使用某個 `AssumeRole*` API 操作取得角色的臨時安全性憑證，您可以在呼叫中包含 `DurationSeconds` 參數。此參數指定 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定之間的角色工作階段持續時間。當您在 `AssumeRole*` 操作中使用 `DurationSeconds` 時，您必須稱之為具有長期憑證的 IAM 使用者。否則，在步驟 3 中對聯合端點的呼叫將失敗。如要了解如何查看或更改角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。
   - 如果使用某個 `GetFederationToken` API 操作取得憑證，您可以在呼叫中包含 `DurationSeconds` 參數。此參數指定您的角色工作階段的持續時間。該值的範圍是 900 秒 (15 分鐘) 到 129,600 秒 (36 小時)。您只能使用 IAM 使用者的長期 AWS 安全登入資料進行此 API 呼

叫。您也可以使用 AWS 帳戶根使用者 認證進行這些呼叫，但我們不建議這樣做。如果您以根使用者身分執行呼叫，則工作階段預設將持續 1 小時。或者，您可以指定 900 秒 (15 分鐘) 到 3,600 秒 (1 小時) 之間的工作階段。

3. 調用 AWS 聯合端點並提供臨時安全憑據以請求登錄令牌。
4. 建構包含該權杖的主控台 URL：
   - 如果在 URL 中使用某個 `AssumeRole*` API 操作，您可以包含 `SessionDuration` HTTP 參數。此參數指定主控台工作階段持續時間，範圍是 900 秒 (15 分鐘) 到 43200 秒 (12 小時)。
   - 如果在 URL 中使用 `GetFederationToken` API 操作，您可以包含 `DurationSeconds` 參數。此參數指定您的聯合身分主控台工作階段的持續時間。該值的範圍是 900 秒 (15 分鐘) 到 129,600 秒 (36 小時)。

   > **Note**
   >
   > - 如果使用 `SessionDuration` 取得臨時憑證，請不要使用 `GetFederationToken` HTTP 參數。這會導致操作失敗。
   > - 使用一個角色的憑證來擔任不同的角色稱為角色鏈結。當您使用角色鏈結時，新憑證的最大持續時間限制為一小時。使用角色授予許可給在 EC2 執行個體上執行的應用程式時，那些應用程式不受此限制。

5. 將 URL 分配給使用者或代表使用者呼叫 URL。

聯合端點提供的 URL 在建立後的 15 分鐘內有效。這與 URL 關聯的臨時安全性憑證工作階段的持續時間 (以秒為單位) 不同。這些憑證在建立時指定的持續時間內有效，從建立的時間算起。

> **Important**
>
> 如果您已在關聯的臨時安全登入 AWS 資料中啟用權限， AWS Management Console 則 URL 會透過授與對您資源的存取權。因此，您應該將 URL 視為機密。我們建議您透過安全的重新引導傳回 URL，例如，在 SSL 連接上使用 302 HTTP 回應狀態碼。如需有關 302 HTTP 回應狀態碼的詳細資訊，請參閱 RFC 2616, section 10.3.3。

若要完成這些任務，您可以使用適用於 AWS Identity and Access Management (IAM) 的 HTTPS Query API 和 AWS Security Token Service (AWS STS)。或者，您可以使用程式設計語言 (例如 Java、Ruby 或 C#) 與適當的 AWS 開發套件。這些方法皆在下列主題中有詳述。

主題

- 使用 IAM 查詢 API 操作的範例程式碼
- 使用 Python 的範例程式碼
- 使用 Java 的範例程式碼
- 範例顯示建構 URL 的方法 (Ruby)

使用 IAM 查詢 API 操作的範例程式碼

您可以建構 URL，讓同盟使用者直接存取 AWS Management Console. 此工作會使用 IAM 和 AWS STS HTTPS 查詢 API。如需有關提出查詢請求的詳細資訊，請參閱提出查詢請求。

> **Note**
>
> 以下過程包括了文字字串的範例。為增加可讀性，一些較長的範例中加入了分行符號。如果您要建立並使用這些字串，必須省略所有分行符號。

若要讓聯合使用者存取您的資源， AWS Management Console

1. 在您的身分驗證系統裡驗證該使用者。
2. 為使用者取得臨時安全性憑證。臨時性憑證由存取金鑰 ID、私密存取金鑰和工作階段字符組成。如需有關建立暫時性憑證檔案的詳細資訊，請參閱 IAM 中的暫時安全憑證。

   若要取得臨時登入資料，您可以呼叫 AWS STS AssumeRoleAPI (建議) 或 GetFederationTokenAPI。如需有關這些 API 作業之間差異的詳細資訊，請參閱 AWS 安全性部落格中了解用於安全地委派 AWS 帳戶存取權的 API 選項。

   > **Important**
   >
   > 當您使用 GetFederationTokenAPI 建立臨時安全登入資料時，您必須指定認證授與給擔任該角色的使用者的權限。對於任何以 `AssumeRole*` 開頭的 API 操作，可使用 IAM 角色來分配許可。對於其他 API 操作，機制會因 API 而異。如需詳細資訊，請參閱 控制臨時安全安全憑證的許可。此外，如果使用 `AssumeRole*` API 操作，您還必須以具有長期憑證的 IAM 使用者身分來呼叫這些操作。否則，在步驟 3 中對聯合端點的呼叫將失敗。

3. 在取得臨時安全性憑證後，將其建構到 JSON 工作階段字串以將其交換為登入權杖。以下範例顯示如何為憑證編碼。請將預留位置文字替換為上一步驟中接收的憑證中的適用值。

```
{"sessionId":"*** temporary access key ID ***",
```

```
"sessionKey":"*** temporary secret access key ***",
"sessionToken":"*** session token ***"}
```

4. URL 編碼前一步驟中的工作階段字串。由於您編碼的資訊是敏感資訊，因此建議您避免對此編碼使用 Web 服務。請改用開發套件中在本機安裝的函數或功能來安全地編碼這些資訊。您可以使用 Python 的 `urllib.quote_plus` 函數、Java 的 `URLEncoder.encode` 函數或 Ruby 的 `CGI.escape` 函數。請參閱本主題後述的範例。
5.

> **Note**
>
> AWS 在這裡支持 POST 請求。

將您的請求發送到 AWS 聯合端點：

`https://region-code.signin.aws.amazon.com/federation`

對於可能的 *region-code* 值清單，請參閱 AWS 登入端點中的 Region (區域) 欄位。您可以選擇使用預設的 AWS 登入聯合端點：

`https://signin.aws.amazon.com/federation`

該請求必須包含 `Action` 與 `Session` 參數；(選擇性) 如果已使用 `AssumeRole*` API 操作，還必須包含 `SessionDuration` HTTP 參數，如下列範例所示。

```
Action = getSigninToken
SessionDuration = time in seconds
Session = *** the URL encoded JSON string created in steps 3 & 4 ***
```

> **Note**
>
> 此步驟中的下列指示僅適用於 GET 請求。

`SessionDuration` HTTP 參數指定主控台工作階段的持續時間。它不同於使用 `DurationSeconds` 參數指定的臨時性憑證的持續時間。您可指定 `SessionDuration` 最高值為 43,200 (12 小時)。如果遺失`SessionDuration`參數，則階段作業會預設為您 AWS STS 在步驟 2 中擷取的認證持續時間 (預設值為一小時)。如需如何使用 `DurationSeconds` 參數指定持續時間的詳細資訊，請參閱 `AssumeRole` API 的文件。建立超過 1 小時的主控台工作階段的功能是聯合端點的 `getSigninToken` 操作所內建的特性。

> **Note**
>
> - 如果使用 SessionDuration 取得臨時憑證，請不要使用 GetFederationToken HTTP 參數。這會導致操作失敗。
> - 使用一個角色的憑證來擔任不同的角色稱為角色鏈結。當您使用角色鏈結時，新憑證的最大持續時間限制為一小時。使用角色授予許可給在 EC2 執行個體上執行的應用程式時，那些應用程式不受此限制。

在啟用具有更長持續時間的主控台工作階段時，您會增加洩露憑證的風險。為了幫助緩解這種風險，您可以在 Role Summary (角色摘要) IAM 主控台頁面上選擇 Revoke Sessions (撤銷工作階段) 來立即停用所有角色的使用中的主控台工作階段。如需詳細資訊，請參閱 撤銷 IAM 角色暫時性安全憑證。

以下為請求具體形式的範例。在此處換行以便閱讀，但您應將其做為單行字串提交。

```
https://signin.aws.amazon.com/federation
?Action=getSigninToken
&SessionDuration=1800
&Session=%7B%22sessionId%22%3A+%22ASIAJUMHIZPTOKTBMK5A%22%2C+%22sessionKey%22
%3A+%22LSD7LWI%2FL%2FN%2BgYpan5QFz0XUpc8s7HYjRsgcsrsm%22%2C+%22sessionToken%2
2%3A+%22FQoDYXdzEBQaDLbj3VWv2u50NN%2F3yyLSASwYtWhPnGPMNmzZFfZsL0Qd3vtYHw5A5dW
AjOsrkdPkghomIe3mJip5%2F0djDBbo7SmO%2FENDEiCdpsQKodTpleKA8xQq0CwFg6a69xdEBQT8
FipATnLbKoyS4b%2FebhnsTUjZZQWp0wXXqFF7gSm%2FMe2tXe0jzsdP0O12obez9lijPSdF1k2b5
PfGhiuyAR9aD5%2BubM0pY86fKex1qsytjvyTbZ9nXe6DvxVDcnCOhOGETJ7XFkSFdH0v%2FYR25C
UAhJ3nXIkIbG7Ucv9cOEpCf%2Fg23ijRgILIBQ%3D%3D%22%7D
```

來自聯合身分端點的回應是一個具有 SigninToken 值的 JSON 文件。看起來與下列範例類似。

```
{"SigninToken":"*** the SigninToken string ***"}
```

6.

> **Note**
>
> AWS 在這裡支持 POST 請求。

最後，建立聯合身分使用者可用於存取 AWS Management Console的 URL。此 URL 與您在 Step 5 中使用的聯合身分 URL 端點相同，外加以下參數：

```
?Action = login
&Issuer = *** the form-urlencoded URL for your internal sign-in page ***
&Destination = *** the form-urlencoded URL to the desired AWS console page ***
&SigninToken = *** the value of SigninToken received in the previous step ***
```

> **Note**
>
> 此步驟中的下列指示僅適用於 GET API。

以下範例顯示 URL 的最終形式。URL 的有效期為 15 分鐘，自建立時算起。在 URL 中嵌入的臨時安全性憑證和主控台工作階段的有效期為在最初請求它們時在 `SessionDuration` HTTP 參數中指定的持續時間。

```
https://signin.aws.amazon.com/federation
?Action=login
&Issuer=https%3A%2F%2Fexample.com
&Destination=https%3A%2F%2Fconsole.aws.amazon.com%2F
&SigninToken=VCQgs5qZZt3Q6fn8Tr5EXAMPLEmLnwB7JjUc-SHwnUUWabcRdnWsi4DBn-dvC
CZ85wrD0nmldUcZEXAMPLE-vXYH4Q__mleuF_W2BE5HYexbe9y4Of-kje53SsjNNecATfjIzpW1
WibbnH6YcYRiBoffZBGExbEXAMPLE5aiKX4THWjQKC6gg6alHu6JFrnOJoK3dtP6I9a6hi6yPgm
iOkPZMmNGmhsvVxetKzr8mx3pxhHbMEXAMPLETv1pij0rok3IyCR2YVcIjqwfWv32HU2Xlj471u
3fU6uOfUComeKiqTGX974xzJOZbdmX_t_lLrhEXAMPLEDDIisSnyHGw2xaZZqudm4mo2uTDk9Pv
9l5K0ZCqIgEXAMPLEcA6tgLPykEWGUyH6BdSC6166n4M4JkXIQgac7_7821YqixsNxZ6rsrpzwf
nQoS14O7R0eJCCJ684EXAMPLEZRdBNnuLbUYpz2Iw3vIN0tQgOujwnwydPscM9F7foaEK3jwMkg
Apeb1-6L_OB12MZhuFxx55555EXAMPLEhyETEd4ZulKPdXHkgl6T9ZkIlHz2Uy1RUTUhhUxNtSQ
nWc5xkbBoEcXqpoSIeK7yhje9Vzhd61AEXAMPLElbWeouACEMG6-Vd3dAgFYd6i5FYoyFrZLWvm
0LSG7RyYKeYN5VIzUk3YWQpyjP0RiT5KUrsUi-NEXAMPLExMOMdoODBEgKQsk-iu2ozh6r8bxwC
RNhujg
```

使用 Python 的範例程式碼

以下範例顯示如何使用 Python 以程式設計方式構建授予聯合身分使用者直接存取 AWS Management Console權限的 URL。以下是兩個範例：

- 通過 GET 請求聯合 AWS
- 通過 POST 請求聯合 AWS

這兩個範例都使用AWS SDK for Python (Boto3)和 AssumeRoleAPI 來取得臨時安全登入資料。

使用 GET 請求

```
import urllib, json, sys
import requests # 'pip install requests'
import boto3 # AWS SDK for Python (Boto3) 'pip install boto3'

# Step 1: Authenticate user in your own identity system.

# Step 2: Using the access keys for an IAM user in your AWS ##,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in Amazon EC2 instance metadata, in environment variables,
# or in a configuration file, and will be discovered automatically by the
# client('sts') function. For more information, see the Python SDK docs:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html
# http://boto3.readthedocs.io/en/latest/reference/services/
sts.html#STS.Client.assume_role
sts_connection = boto3.client('sts')

assumed_role_object = sts_connection.assume_role(
    RoleArn="arn:aws:iam::account-id:role/ROLE-NAME",
    RoleSessionName="AssumeRoleSession",
)

# Step 3: Format resulting temporary credentials into JSON
url_credentials = {}
url_credentials['sessionId'] =
 assumed_role_object.get('Credentials').get('AccessKeyId')
url_credentials['sessionKey'] =
 assumed_role_object.get('Credentials').get('SecretAccessKey')
url_credentials['sessionToken'] =
 assumed_role_object.get('Credentials').get('SessionToken')
json_string_with_temp_credentials = json.dumps(url_credentials)
```

```
# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
 parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
 temporary credentials
# as parameters.
request_parameters = "?Action=getSigninToken"
request_parameters += "&SessionDuration=43200"
if sys.version_info[0] < 3:
    def quote_plus_function(s):
        return urllib.quote_plus(s)
else:
    def quote_plus_function(s):
        return urllib.parse.quote_plus(s)
request_parameters += "&Session=" +
 quote_plus_function(json_string_with_temp_credentials)
request_url = "https://signin.aws.amazon.com/federation" + request_parameters
r = requests.get(request_url)
# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)

# Step 5: Create URL where users can use the sign-in token to sign in to
# the console. This URL must be used within 15 minutes after the
# sign-in token was issued.
request_parameters = "?Action=login"
request_parameters += "&Issuer=Example.org"
request_parameters += "&Destination=" + quote_plus_function("https://
console.aws.amazon.com/")
request_parameters += "&SigninToken=" + signin_token["SigninToken"]
request_url = "https://signin.aws.amazon.com/federation" + request_parameters

# Send final URL to stdout
print (request_url)
```

使用 POST 請求

```
import urllib, json, sys
import requests # 'pip install requests'
import boto3 # AWS SDK for Python (Boto3) 'pip install boto3'
import os
from selenium import webdriver # 'pip install selenium', 'brew install chromedriver'

# Step 1: Authenticate user in your own identity system.
```

```
# Step 2: Using the access keys for an IAM user in your A AWS ##,
# call "AssumeRole" to get temporary access keys for the federated user

# Note: Calls to AWS STS AssumeRole must be signed using the access key ID
# and secret access key of an IAM user or using existing temporary credentials.
# The credentials can be in Amazon EC2 instance metadata, in environment variables,

# or in a configuration file, and will be discovered automatically by the
# client('sts') function. For more information, see the Python SDK docs:
# http://boto3.readthedocs.io/en/latest/reference/services/sts.html
# http://boto3.readthedocs.io/en/latest/reference/services/
sts.html#STS.Client.assume_role
if sys.version_info[0] < 3:
    def quote_plus_function(s):
        return urllib.quote_plus(s)
else:
    def quote_plus_function(s):
        return urllib.parse.quote_plus(s)

sts_connection = boto3.client('sts')

assumed_role_object = sts_connection.assume_role(
    RoleArn="arn:aws:iam::account-id:role/ROLE-NAME",
    RoleSessionName="AssumeRoleDemoSession",
)

# Step 3: Format resulting temporary credentials into JSON
url_credentials = {}
url_credentials['sessionId'] =
 assumed_role_object.get('Credentials').get('AccessKeyId')
url_credentials['sessionKey'] =
 assumed_role_object.get('Credentials').get('SecretAccessKey')
url_credentials['sessionToken'] =
 assumed_role_object.get('Credentials').get('SessionToken')
json_string_with_temp_credentials = json.dumps(url_credentials)

# Step 4. Make request to AWS federation endpoint to get sign-in token. Construct the
 parameter string with
# the sign-in action request, a 12-hour session duration, and the JSON document with
 temporary credentials
# as parameters.
request_parameters = {}
request_parameters['Action'] = 'getSigninToken'
request_parameters['SessionDuration'] = '43200'
```

```
request_parameters['Session'] = json_string_with_temp_credentials

request_url = "https://signin.aws.amazon.com/federation"
r = requests.post( request_url, data=request_parameters)

# Returns a JSON document with a single element named SigninToken.
signin_token = json.loads(r.text)

# Step 5: Create a POST request where users can use the sign-in token to sign in to
# the console. The POST request must be made within 15 minutes after the
# sign-in token was issued.
request_parameters = {}
request_parameters['Action'] = 'login'
request_parameters['Issuer']='Example.org'
request_parameters['Destination'] = 'https://console.aws.amazon.com/'
request_parameters['SigninToken'] =signin_token['SigninToken']

jsrequest = '''
var form = document.createElement('form');
form.method = 'POST';
form.action = '{request_url}';
request_parameters = {request_parameters}
for (var param in request_parameters) {{
    if (request_parameters.hasOwnProperty(param)) {{
        const hiddenField = document.createElement('input');
        hiddenField.type = 'hidden';
        hiddenField.name = param;
        hiddenField.value = request_parameters[param];
        form.appendChild(hiddenField);
    }}
}}
document.body.appendChild(form);
form.submit();
'''.format(request_url=request_url, request_parameters=request_parameters)

driver = webdriver.Chrome()
driver.execute_script(jsrequest);
```

使用 Java 的範例程式碼

以下範例顯示如何使用 Java 以程式設計方式構建授予聯合身分使用者直接存取 AWS Management Console權限的 URL。下列程式碼片段使用 AWS SDK for Java。

```
import java.net.URLEncoder;
import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
// Available at http://www.json.org/java/index.html
import org.json.JSONObject;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;


/* Calls to AWS STS API operations must be signed using the access key ID
   and secret access key of an IAM user or using existing temporary
   credentials. The credentials should not be embedded in code. For
   this example, the code looks for the credentials in a
   standard configuration file.
*/
AWSCredentials credentials =
  new PropertiesCredentials(
         AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));

AWSSecurityTokenServiceClient stsClient =
  new AWSSecurityTokenServiceClient(credentials);

GetFederationTokenRequest getFederationTokenRequest =
  new GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(1800);
getFederationTokenRequest.setName("UserName");

// A sample policy for accessing Amazon Simple Notification Service (Amazon SNS) in the
 console.

String policy = "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Action\":\"sns:*\"," +
  "\"Effect\":\"Allow\",\"Resource\":\"*\"}]}";

getFederationTokenRequest.setPolicy(policy);

GetFederationTokenResult federationTokenResult =
  stsClient.getFederationToken(getFederationTokenRequest);
```

```
Credentials federatedCredentials = federationTokenResult.getCredentials();

// The issuer parameter specifies your internal sign-in
// page, for example https://mysignin.internal.mycompany.com/.
// The console parameter specifies the URL to the destination console of the
// AWS Management Console. This example goes to Amazon SNS.
// The signin parameter is the URL to send the request to.

String issuerURL = "https://mysignin.internal.mycompany.com/";
String consoleURL = "https://console.aws.amazon.com/sns";
String signInURL = "https://signin.aws.amazon.com/federation";

// Create the sign-in token using temporary credentials,
// including the access key ID,  secret access key, and session token.
String sessionJson = String.format(
  "{\"%1$s\":\"%2$s\",\"%3$s\":\"%4$s\",\"%5$s\":\"%6$s\"}",
  "sessionId", federatedCredentials.getAccessKeyId(),
  "sessionKey", federatedCredentials.getSecretAccessKey(),
  "sessionToken", federatedCredentials.getSessionToken());

// Construct the sign-in request with the request sign-in token action, a
// 12-hour console session duration, and the JSON document with temporary
// credentials as parameters.

String getSigninTokenURL = signInURL +
                           "?Action=getSigninToken" +
                           "&DurationSeconds=43200" +
                           "&SessionType=json&Session=" +
                           URLEncoder.encode(sessionJson,"UTF-8");

URL url = new URL(getSigninTokenURL);

// Send the request to the AWS federation endpoint to get the sign-in token
URLConnection conn = url.openConnection ();

BufferedReader bufferReader = new BufferedReader(new
  InputStreamReader(conn.getInputStream()));
String returnContent = bufferReader.readLine();

String signinToken = new JSONObject(returnContent).getString("SigninToken");

String signinTokenParameter = "&SigninToken=" + URLEncoder.encode(signinToken,"UTF-8");
```

```
// The issuer parameter is optional, but recommended. Use it to direct users
// to your sign-in page when their session expires.

String issuerParameter = "&Issuer=" + URLEncoder.encode(issuerURL, "UTF-8");

// Finally, present the completed URL for the AWS console session to the user

String destinationParameter = "&Destination=" + URLEncoder.encode(consoleURL,"UTF-8");
String loginURL = signInURL + "?Action=login" +
                     signinTokenParameter + issuerParameter + destinationParameter;
```

範例顯示建構 URL 的方法 (Ruby)

以下範例顯示如何使用 Ruby 以程式設計方式構建授予聯合身分使用者直接存取 AWS Management Console權限的 URL。此程式碼片段使用 AWS SDK for Ruby。

```
require 'rubygems'
require 'json'
require 'open-uri'
require 'cgi'
require 'aws-sdk'

# Create a new STS instance
#
# Note: Calls to AWS STS API operations must be signed using an access key ID
# and secret access key. The credentials can be in EC2 instance metadata
# or in environment variables and will be automatically discovered by
# the default credentials provider in the AWS Ruby SDK.
sts = Aws::STS::Client.new()

# The following call creates a temporary session that returns
# temporary security credentials and a session token.
# The policy grants permissions to work
# in the AWS SNS console.

session = sts.get_federation_token({
  duration_seconds: 1800,
  name: "UserName",
  policy: "{\"Version\":\"2012-10-17\",\"Statement\":{\"Effect\":\"Allow\",\"Action\":
\"sns:*\",\"Resource\":\"*\"}}",
})

# The issuer value is the URL where users are directed (such as
```

```
# to your internal sign-in page) when their session expires.
#
# The console value specifies the URL to the destination console.
# This example goes to the Amazon SNS console.
#
# The sign-in value is the URL of the AWS STS federation endpoint.
issuer_url = "https://mysignin.internal.mycompany.com/"
console_url = "https://console.aws.amazon.com/sns"
signin_url = "https://signin.aws.amazon.com/federation"

# Create a block of JSON that contains the temporary credentials
# (including the access key ID, secret access key, and session token).
session_json = {
  :sessionId => session.credentials[:access_key_id],
  :sessionKey => session.credentials[:secret_access_key],
  :sessionToken => session.credentials[:session_token]
}.to_json

# Call the federation endpoint, passing the parameters
# created earlier and the session information as a JSON block.
# The request returns a sign-in token that's valid for 15 minutes.
# Signing in to the console with the token creates a session
# that is valid for 12 hours.
get_signin_token_url = signin_url +
                       "?Action=getSigninToken" +
                       "&SessionType=json&Session=" +
                       CGI.escape(session_json)

returned_content = URI.parse(get_signin_token_url).read

# Extract the sign-in token from the information returned
# by the federation endpoint.
signin_token = JSON.parse(returned_content)['SigninToken']
signin_token_param = "&SigninToken=" + CGI.escape(signin_token)

# Create the URL to give to the user, which includes the
# sign-in token and the URL of the console to open.
# The "issuer" parameter is optional but recommended.
issuer_param = "&Issuer=" + CGI.escape(issuer_url)
destination_param = "&Destination=" + CGI.escape(console_url)
login_url = signin_url + "?Action=login" + signin_token_param +
  issuer_param + destination_param
```

暫時性安全憑證的其他資源。

以下案例和應用程式可以指導您使用暫時性安全憑證：

- 如何 AWS STS SourceIdentity 與您的身分提供者整合。這篇文章向您展示如何在使用 Okta，Ping 或 OneLogin 作為 IdP 時設置 AWS STS `SourceIdentity`屬性。
- OIDC 聯盟。 本節討論如何在`AssumeRoleWithWebIdentity`使用 OIDC 聯合和 API 時設定 IAM 角色。
- 設定受 MFA 保護的 API 存取. 本主題說明如何使用角色來要求多重要素驗證 (MFA) 以保護您的帳戶中的敏感 API 動作。

如需有關原則和權限的詳細資訊， AWS 請參閱下列主題：

- AWS 資源存取管理
- 政策評估邏輯.
- Amazon Simple Storage Service 使用者指南中的管理 Amazon S3 資源的存取許可。
- 若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。

標記 IAM 資源

標籤是一種自訂屬性標籤，可由您指派給 AWS 資源。每個標籤有兩個部分：

- 標籤鍵 (例如，`CostCenter`、`Environment`、`Project` 或 `Purpose`)。
- 一個名為標籤值 (例如，`111122223333`、`Production` 或團隊名稱) 的選用欄位。忽略標籤值基本上等同於使用空字串。

這些合稱為鍵值組。關於可具有 IAM 資源標籤數量的限制，請參閱 IAM 和 AWS STS 配額。

> **Note**
>
> 如需有關標籤索引鍵和標籤索引鍵值區分大小寫的詳細資訊，請參閱 Case sensitivity。

標籤可協助您識別和整理資 AWS 源。許多 AWS 服務都支援標記，因此您可以將相同標籤指派給來自不同服務的資源，以指出資源是相關的。例如，您可以將相同的標籤指派給您指派給 Amazon S3 儲存貯體的 IAM 角色。若要取得有關標籤策略的更多資訊，請參閱標籤 AWS 資源使用指南。

此外，若要使用標籤來識別、整理和追蹤 IAM 資源，您可以在 IAM 政策中使用標籤，以協助控制誰可以檢視您的資源並與其互動。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

選擇標 AWS 籤命名慣例

當您開始將標籤連接至您的 IAM 資源時，請小心選擇您的標籤命名慣例。將相同的慣例套用至所有 AWS 標籤。如果您在策略中使用標籤來控制對 AWS 資源的存取，這一點尤其重要。如果您在 AWS中已經使用標籤，請檢閱您的命名慣例和加以調整。

> **Note**
>
> 如果您的帳戶是的成員 AWS Organizations，請參閱 Organizations 使用指南中的標籤政策，以深入瞭解如何在 Organizations 中使用標籤。

標籤命名的最佳實務

以下是標籤的一些最佳實務和命名慣例。

請確保以一致的方式使用標籤名稱。例如，標籤 `CostCenter` 和 `costcenter` 是不同的，因此可能會將其中一個設定為用於財務分析與報表的成本配置標籤，而另一個則可能不會如此設定。同樣地，許多資源都會顯示在 AWS 主控台中的標Name籤，但標name籤則不會出現。如需有關標籤索引鍵和標籤索引鍵值區分大小寫的詳細資訊，請參閱 Case sensitivity。

許多標籤是由各種 AWS 服務預先定義 AWS 或自動創建的。許多 AWS定義的標籤名稱使用全部小寫字母，連字號分隔名稱中的文字，而前綴則用來識別標籤的來源服務。例如：

- `aws:ec2spot:fleet-request-id` 識別啟動執行個體的 Amazon EC2 Spot 執行個體請求。
- `aws:cloudformation:stack-name`標識創建資源的 AWS CloudFormation 堆棧。
- `elasticbeanstalk:environment-name` 識別建立資源的應用程式。

請考慮使用全部小寫字母來命名標籤，以連字號分隔文字，並且採用識別組織名稱或縮寫名稱的字首。例如，對於名稱為虛擬的公司 AnyCompany，您可以定義標籤，例如：

- `anycompany:cost-center` 可識別內部成本中心代碼
- `anycompany:environment-type` 可識別環境是否為開發、測試或生產
- `anycompany:application-id` 可識別建立資源的應用程式

前置詞可確保標籤清楚地識別為已由您的組織定義，而不是由 AWS 您可能正在使用的第三方工具定義。將所有小寫字母和連字號 (作為分隔符號) 搭配使用可避免對如何大寫標籤名稱造成混淆。例如：`anycompany:project-id` 比 `ANYCOMPANY:ProjectID`、`anycompany:projectID` 或 `Anycompany:ProjectId` 更容易記住。

IAM 和標記的規則 AWS STS

多種慣例會掌管在 IAM 和 AWS STS中標籤的建立和應用。

命名標籤

為 IAM 資源、 AWS STS 假定角色工作階段和 AWS STS 聯合使用者工作階段制定標籤命名慣例時，請遵守下列慣例：

字元要求 – 標籤鍵和值可以包含任意組合字母、數字、空格和 _ . : / = + - @ 符號。

區分大小寫 – 標籤鍵的區分大小寫會因標記的 IAM 資源類型而有所不同。IAM 使用者和角色的標籤鍵值不區分大小寫，但會保留大小寫。這表示您無法個別擁有 **`Department`** 和 **`department`** 標籤鍵。如果您使用 **`Department=finance`** 標籤為使用者加上標籤，而且您新增 **`department=hr`** 標籤，它會取代第一個標籤。不會新增第二個標籤。

對於其他 IAM 資源類型，標籤鍵值會區分大小寫。這意味著您可以有單獨的 **`Costcenter`** 和 **`costcenter`** 標籤鍵。例如，如果您已使用 **`Costcenter = 1234`** 標籤標記客戶受管政策，並新增 **`costcenter = 5678`** 標籤，則政策將同時具有 **`Costcenter`** 和 **`costcenter`** 標籤鍵。

最佳作法是，建議您避免使用類似的標籤使用不一致的大小寫處理。建議您決定大寫標籤的策略，並一致地在所有資源類型中實作該策略。若要進一步了解標記的最佳做法，請參閱 AWS 一般參考. AWS

下列清單顯示連接至 IAM 資源之標籤鍵的區分大小寫差異。

標籤鍵值不區分大小寫：

- IAM 角色

- IAM 使用者

標籤鍵值區分大小寫：

- 客戶受管政策
- 執行個體設定檔
- OpenID Connect 身分提供者
- SAML 身分提供者
- 伺服器憑證
- 虛擬 MFA 裝置

此外，適用下列規則：

- 您不能建立標籤鍵或以文字 **aws:** 為開頭的值。此標籤前綴保留供 AWS 內部使用。
- 您可以使用 **phoneNumber =** 之類的空值來建立標籤。您不能建立空的標籤鍵。
- 您可以在單一標籤不指定多個值，但您可以在單一值建立自訂多值結構。例如，假設使用者 Zhang 與工程團隊和 QA 團隊合作。如果您連接 **team = Engineering** 標籤，然後連接 **team = QA** 標籤，則您將標籤的值從 **Engineering** 變更為 **QA**。反之，您可以在單一標籤中使用自訂分隔符號來包含多個值。在這個範例中，您可以將 **team = Engineering:QA** 標籤連接至 Zhang。

> **Note**
>
> 若要在此範例中使用 **team** 標籤控制工程師的存取權，您必須建立一個政策，以允許可能包含 **Engineering** 的每個組態，包括 **Engineering:QA**。若要進一步了解在政策中使用標籤的詳細資訊，請參閱 使用標籤控制對 IAM 使用者和角色的存取。

套用和編輯標籤

將標籤連接至 IAM 資源時，請遵守下列慣例：

- 您可以標記大多數 IAM 資源，但不能標記群組、擔任的角色、存取報告或硬體式 MFA 裝置。
- 您不能使用標籤編輯器來標記 IAM 資源。標籤編輯器不支援 IAM 標籤。如需使用標籤編輯器搭配其他服務的詳細資訊，請參閱 AWS Resource Groups 使用者指南中的使用標籤編輯器。
- 若要標記 IAM 資源，您必須擁有特定的許可。若要標記或取消標記資源，您還必須有列出標籤的許可。如需詳細資訊，請參閱本頁結尾每個 IAM 資源的主題清單。

- AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。
- 您可以將相同標籤套用到多個 IAM 資源。例如，假設您有一個名為 AWS_Development 的部門，其中包含 12 名成員。您可以擁有 12 名使用者，以及具備 **department** 標籤鍵和 **awsDevelopment** 值的角色 (**department = awsDevelopment**)。您也可以在其他支援標籤的服務中對資源使用相同標籤。
- IAM 實體 (使用者或角色) 不能擁有相同標籤鍵的多個執行個體。例如，如果您有一個使用者與標籤鍵值對 **costCenter = 1234**，您可以接著連接標籤鍵值對 **costCenter = 5678**。IAM 會更新 **costCenter** 標籤的值至 **5678**。
- 若要編輯連接至 IAM 實體 (使用者或角色) 的標籤，請連接含有新值的標籤以覆寫現有標籤。例如，假設您有一個使用者，含有標籤鍵值組 **department = Engineering**。如果您需要將使用者新增到 QA 部門，則可以將 **department = QA** 標籤鍵值組連接至使用者。這會造成 **Engineering** 標籤鍵的 **department** 值取代為 **QA** 值。

主題

標記 IAM 使用者

您可以使用 IAM 標籤鍵值組向 IAM 使用者新增自訂屬性。例如，若要將位置資訊新增到使用者，您可以新增標籤鍵 **location** 和標籤值 **us_wa_seattle**。或者，您可以使用三個不同的位置的標籤鍵值組：**loc-country = us**、**loc-state = wa** 和 **loc-city = seattle**。您可以使用標籤來控制使用者對資源的存取權，或控制哪些標籤可以連接到使用者。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

標記 IAM 使用者所需的許可

您必須設定許可，以允許 IAM 使用者標記其他使用者。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

允許 IAM 實體新增、列出，或移除特定使用者的標籤

將下列陳述式新增至需要管理標籤之 IAM 使用者的許可政策。使用您的帳戶並使用需要其標籤受管之使用者的名稱取代 *<username>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

允許 IAM 使用者自行管理標籤

將下列陳述式新增至使用者的許可政策，以讓使用者能管理其自身的標籤。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::user/${aws:username}"
}
```

允許 IAM 使用者新增特定使用者的標籤

將下列陳述式新增至 IAM 使用者的許可政策，而該使用者需要新增 (非移除) 特定使用者的標籤。

> **Note**
>
> 此 `iam:TagUser` 動作需要您也包含 `iam:ListUserTags` 動作。

若要使用此政策，請使用其標籤需要受管之使用者的名稱取代 *<username>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 提供 IAM 的完整存取權。

管理 IAM 使用者的標籤 (主控台)

您可以從 AWS Management Console管理 IAM 使用者的標籤。

管理使用者的標籤 (主控台)

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台導覽窗格中，選擇 Users (使用者)，然後選擇您要編輯的使用者名稱。
3. 選擇 Tags (標籤) 標籤，然後完成以下其中一個動作：
   - 如果使用者還沒有標籤，請選擇 新增標籤。
   - 選擇 Manage tags (管理標籤) 來管理現有的一組標籤。
4. 新增或移除標籤來完成標籤的設定。接著選擇 Save changes (儲存變更)。

管理 IAM 使用者 (AWS CLI 或 AWS API) 的標籤

您可以列出、連接，或移除 IAM 使用者的標籤。您可以使用 AWS CLI 或 AWS API 來管理 IAM 使用者的標籤。

列出目前附加至 IAM 使用者 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：AWS list-user-tags
- AWS API：ListUserTags

若要將標籤附加至 IAM 使用者 (AWS CLI 或 AWS API)

- AWS CLI：aws iam tag-user
- AWS API：TagUser

若要從 IAM 使用者 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：aws iam untag-user
- AWS API：UntagUser

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記 IAM 角色

您可以使用 IAM 標籤鍵值組將自訂屬性新增至 IAM 角色。例如，若要將位置資訊新增至角色，您可以新增標籤鍵 **`location`** 和標籤值 **`us_wa_seattle`**。或者，您可以使用三個不同的位置的標籤鍵值組：**`loc-country = us`**、**`loc-state = wa`** 和 **`loc-city = seattle`**。您可以使用標籤來控制角色對資源的存取權，或控制哪些標籤可以連接到角色。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

標記 IAM 角色所需的許可

您必須設定許可，允許 IAM 角色標記其他實體 (使用者或角色)。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListRoleTags`
- `iam:TagRole`
- `iam:UntagRole`
- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

允許 IAM 角色新增、列出，或移除特定使用者的標籤

將下列陳述式新增至需要管理標籤之 IAM 角色的許可政策。使用您的帳戶並使用需要其標籤受管之使用者的名稱取代 *<username>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

允許 IAM 角色新增特定使用者的標籤

將下列陳述式新增至 IAM 角色的許可政策，而該角色需要新增 (非移除) 特定使用者的標籤。

若要使用此政策，請使用其標籤需要受管之使用者的名稱取代 *<username>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:ListUserTags",
        "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

允許 IAM 角色新增、列出，或移除特定角色的標籤

將下列陳述式新增至需要管理標籤之 IAM 角色的許可政策。用其標籤需要受管的角色名稱取代 *<rolename>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListRoleTags",
        "iam:TagRole",
        "iam:UntagRole"
    ],
    "Resource": "arn:aws:iam::<account-number>:role/<rolename>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理 IAM 角色的標籤 (主控台)

您可以從 AWS Management Console管理 IAM 角色的標籤。

管理角色的標籤 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台導覽窗格中，選擇 Roles (角色)，然後選擇您要編輯的角色名稱。
3. 選擇 Tags (標籤) 標籤，然後完成以下其中一個動作：
   - 如果角色還沒有標籤，請選擇 Add new tag (新增標籤)。
   - 選擇 Manage tags (管理標籤) 來管理現有的一組標籤。
4. 新增或移除標籤來完成標籤的設定。然後，選擇 Save changes (儲存變更)。

管理 IAM 角色 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接，或移除 IAM 角色的標籤。您可以使用 AWS CLI 或 AWS API 來管理 IAM 角色的標籤。

列出目前附加至 IAM 角色 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：list-role-tags
- AWS API：ListRoleTags

若要將標籤附加到 IAM 角色 (AWS CLI 或 AWS API)

- AWS CLI：aws iam tag-role
- AWS API：TagRole

若要從 IAM 角色 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：aws iam untag-role
- AWS API：UntagRole

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記客戶受管政策

您可以使用 IAM 標籤鍵值組，將自訂屬性新增至客戶受管政策。例如，若要使用部門資訊來標記政策，您可以新增標籤鍵 **`Department`** 和標籤值 **`eng`**。或者，您可能想要標記政策，以指出這些政策適用於特定環境，例如 **`Environment = lab`**。您可以使用標籤來控制對資源的存取權，或控制哪些標籤可以連接到資源。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

標記客戶受管政策所需的許可

您必須設定許可，以允許 IAM 實體 (使用者或角色) 標記客戶受管政策。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListPolicyTags`
- `iam:TagPolicy`
- `iam:UntagPolicy`

允許 IAM 實體 (使用者或角色) 新增、列出或移除客戶受管政策的標記

將下列陳述式新增至需要管理標籤之 IAM 實體的許可政策。使用您的帳戶號碼，用需要其標籤受管的政策名稱取代 *<policyname>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListPolicyTags",
        "iam:TagPolicy",
        "iam:UntagPolicy"
    ],
    "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

允許 IAM 實體 (使用者或角色) 新增標籤至特定客戶受管政策

將下列陳述式新增至 IAM 實體的許可政策，而該實體需要新增 (非移除) 特定政策的標籤。

> **Note**
>
> 此 `iam:TagPolicy` 動作需要您也包含 `iam:ListPolicyTags` 動作。

若要使用此政策，請用需要其標籤受管的政策名稱取代 *<policyname>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:ListPolicyTags",
        "iam:TagPolicy"
    ],
    "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理 IAM 客戶受管政策的標籤 (主控台)

您可以從 AWS Management Console管理 IAM 客戶受管政策的標籤。

管理客戶受管政策的標籤 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台導覽窗格中，選擇 Policies (政策)，然後選擇您要編輯的客戶受管政策名稱。
3. 選擇標籤索引標籤，然後選擇管理標籤。
4. 新增或移除標籤來完成標籤的設定。接著選擇 Save changes (儲存變更)。

管理 IAM 客戶受管政策 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接或移除 IAM 客戶受管政策的標籤。您可以使用 AWS CLI 或 AWS API 來管理 IAM 客戶受管政策的標籤。

列出目前附加至 IAM 客戶受管政策 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：AWS list-policy-tags
- AWS API：ListPolicyTags

將標籤附加至 IAM 客戶受管政策 (AWS CLI 或 AWS API)

- AWS CLI: aws iam tag-policy
- AWS API：TagPolicy

從 IAM 客戶受管政策 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI: aws iam untag-policy

- AWS API：UntagPolicy

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記 IAM 身分提供者

您可以使用 IAM 標籤鍵值配對將自訂屬性新增至 IAM 身分提供者 (IdPs)。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

若要進一步了解 IAM IdPs 中的標記，請參閱下列主題：

主題

- 標記 OpenID Connect (OIDC) 身分提供者
- 標記 IAM SAML 身分提供者

標記 OpenID Connect (OIDC) 身分提供者

您可以使用 IAM 標籤鍵值將自訂屬性新增至 IAM OpenID Connect (OIDC) 身分提供者。例如，若要識別 OIDC 身分提供者，您可以新增標籤鍵 **`google`** 和標籤值 **`oidc`**。您可以使用標籤來控制對資源的存取權，或控制哪些標籤可以連接到物件。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

標記 IAM OIDC 身分提供者所需的許可

您必須設定許可，允許 IAM 實體 (使用者或角色) 標記 IAM OIDC 身分提供者。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListOpenIDConnectProviderTags`
- `iam:TagOpenIDConnectProvider`
- `iam:UntagOpenIDConnectProvider`

允許 IAM 實體 (使用者或角色) 新增、列出或移除 IAM OIDC 身分提供者的標籤

將下列陳述式新增至需要管理標籤之 IAM 實體的許可政策。使用您的帳戶號碼，並將 *<OIDC ProviderName >* 替換為需要管理其標籤的 OIDC 提供者的名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListOpenIDConnectProviderTags",
        "iam:TagOpenIDConnectProvider",
        "iam:UntagOpenIDConnectProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

允許 IAM 實體 (使用者或角色) 將標籤新增至特定 IAM OIDC 身分提供者

將下列陳述式新增至 IAM 實體的許可政策，而該實體需要新增 (非移除) 特定身分提供者的標籤。

> **Note**
>
> 此 `iam:TagOpenIDConnectProvider` 動作需要您也包含 `iam:ListOpenIDConnectProviderTags` 動作。

若要使用此原則，請將 *<OIDC ProviderName >* 取代為需要管理其標籤的 OIDC 提供者名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListOpenIDConnectProviderTags",
        "iam:TagOpenIDConnectProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理 IAM OIDC 身分提供者的標籤 (主控台)

您可以從 AWS Management Console管理 IAM OIDC 身分提供者的標籤。

管理 OIDC 身分提供者 (主控台) 的標籤

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台的導覽窗格中，選擇 Identity providers (身分提供者)，然後選擇您所要編輯的身分提供者之名稱。
3. 在 Tags (標籤) 區段中，選擇 Manage tags (管理標籤)，然後完成下列其中一個動作：
   - 如果 OIDC 身分提供者尚未有標籤或新增標籤，請選擇 Add tag (新增標籤)。
   - 編輯現有標籤鍵和值。
   - 請選擇 Remove tag (移除標籤) 以移除標籤。
4. 接著選擇 Save changes (儲存變更)。

管理 IAM OIDC 身分識別提供者 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接或移除 IAM OIDC 身分提供者的標籤。您可以使用 AWS CLI 或 AWS API 來管理 IAM OIDC 身分識別提供者的標籤。

列出目前附加至 IAM OIDC 身分識別提供者 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：AWS IAM list-open-id-connect- 提供者標籤
- AWS API：ListOpen識別碼 ConnectProviderTags

若要將標籤附加至 IAM OIDC 身分識別提供者 (AWS CLI 或 AWS API)

- AWS CLI：AWS 我的供應 tag-open-id-connect商
- AWS API：TagOpen識別碼 ConnectProvider

若要從 IAM OIDC 身分識別提供者 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：AWS 我的供應 untag-open-id-connect商
- AWS API：UntagOpen識別碼 ConnectProvider

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記 IAM SAML 身分提供者

您可以使用 IAM 標籤鍵值組，將自訂屬性新增至 SAML 身分提供者。例如，若要識別提供者，您可以新增標籤鍵 **okta** 和標籤值 **saml**。您可以使用標籤來控制對資源的存取權，或控制哪些標籤可以連接到物件。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

標記 SAML 身分提供者所需的許可

您必須設定許可，以允許 IAM 實體 (使用者或角色) 標記以 SAML 2.0 為基礎的身分識別提供者 ()。IdPs您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListSAMLProviderTags`
- `iam:TagSAMLProvider`
- `iam:UntagSAMLProvider`

允許 IAM 實體 (使用者或角色) 新增、列出或移除 SAML 身分提供者的標籤

將下列陳述式新增至需要管理標籤之 IAM 實體的許可政策。使用您的帳戶號碼，並將 *<SAML ProviderName >* 取代為需要管理其標記的 SAML 提供者的名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListSAMLProviderTags",
        "iam:TagSAMLProvider",
        "iam:UntagSAMLProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

允許 IAM 實體 (使用者或角色) 將標籤新增至特定 SAML 身分提供者

將下列陳述式新增至 IAM 實體的許可政策，而該實體需要新增 (非移除) 特定 SAML 提供者的標籤。

> **Note**
>
> 此 `iam:TagSAMLProvider` 動作需要您也包含 `iam:ListSAMLProviderTags` 動作。

若要使用此原則，請將 *<SAML ProviderName >* 取代為需要管理其標記的 SAML 提供者名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListSAMLProviderTags",
        "iam:TagSAMLProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理 IAM SAML 身分提供者的標籤 (主控台)

您可以從 AWS Management Console管理 IAM SAML 身分提供者的標籤。

管理 SAML 身分提供者 (主控台) 的標籤

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在主控台的導覽窗格中，選擇 Identity providers (身分提供者)，然後選擇您所要編輯的 SAML 身分提供者之名稱。
3. 在 Tags (標籤) 區段中，選擇 Manage tags (管理標籤)，然後完成下列其中一個動作：
   - 如果 SAML 身分提供者尚未有標籤或新增標籤，請選擇 Add tag (新增標籤)。
   - 編輯現有標籤鍵和值。
   - 請選擇 Remove tag (移除標籤) 以移除標籤。
4. 新增或移除標籤來完成標籤的設定。接著選擇 Save changes (儲存變更)。

管理 IAM SAML 身分識別提供者 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接或移除 IAM SAML 身分提供者的標籤。您可以使用 AWS CLI 或 AWS API 來管理 IAM SAML 身分識別提供者的標籤。

列出目前附加至 SAML 身分識別提供者 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：AWS list-saml-provider-tags
- AWS API：列表參考 ProviderTags

將標籤附加至 SAML 身分識別提供者 (AWS CLI 或 AWS API)

- AWS CLI：AWS tag-saml-provider
- AWS API：標籤範圍提供者

從 SAML 身分識別提供者 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：AWS untag-saml-provider
- AWS API：無標籤提供者

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記 Amazon EC2 角色的執行個體描述檔

啟動 Amazon EC2 執行個體時，指定要與執行個體相關聯的 IAM 角色。執行個體描述檔是適用於 IAM 角色的容器，可讓您在執行個體啟動時將角色資訊傳遞至 Amazon EC2 執行個體。您可以在使用 AWS CLI 或 AWS API 時標記執行個體設定檔。

您可以使用 IAM 標籤鍵值組將自訂屬性新增至執行個體描述檔。例如，若要將部門資訊新增至執行個體設定檔，您可以新增標籤鍵 **`access-team`** 和標籤值 **`eng`**。這樣做可讓具有相符標籤的主體存取具有相同標籤的執行個體設定檔。您可以使用多個標籤鍵值組來指定團隊和專案：**`access-team = eng`** 和 **`project = peg`**。您可以使用標籤來控制使用者對資源的存取權，或控制哪些標籤可以連接到使用者。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

標記執行個體設定檔所需的許可

您必須設定許可，允許 IAM 實體 (使用者或角色) 標記執行個體描述檔。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListInstanceProfileTags`
- `iam:TagInstanceProfile`
- `iam:UntagInstanceProfile`

允許 IAM 實體 (使用者或角色) 新增、列出或移除執行個體描述檔的標籤

將下列陳述式新增至需要管理標籤之 IAM 實體的許可政策。使用您的帳戶號碼，並將 *< InstanceProfileName >* 替換為需要管理其標籤的實例配置文件的名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListInstanceProfileTags",
        "iam:TagInstanceProfile",
        "iam:UntagInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

允許 IAM 實體 (使用者或角色) 將標籤新增至特定執行個體描述檔

將下列陳述式新增至 IAM 實體的許可政策，而該實體需要新增 (非移除) 特定執行個體描述檔的標籤。

> **Note**
>
> 此 `iam:TagInstanceProfile` 動作需要您也包含 `iam:ListInstanceProfileTags` 動作。

若要使用此原則，請將 *< InstanceProfileName >* 取代為需要管理其標記的執行個體設定檔名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListInstanceProfileTags",
        "iam:TagInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理執行個體設定檔 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接或移除執行個體設定檔的標籤。您可以使用 AWS CLI 或 AWS API 來管理執行個體設定檔的標籤。

列出目前附加至執行個體設定檔 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：list-instance-profile-tags
- AWS API：ListInstanceProfileTags

將標籤附加至執行個體設定檔 (AWS CLI 或 AWS API)

- AWS CLI：tag-instance-profile
- AWS API：TagInstanceProfile

從執行個體設定檔 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：untag-instance-profile
- AWS API：UntagInstanceProfile

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記伺服器憑證

如果您使用 IAM 來管理 SSL/TLS 憑證，您可以使用 AWS CLI 或 AWS API 在 IAM 中標記伺服器憑證。對於 AWS Certificate Manager (ACM) 支援的區域中的憑證，建議您使用 ACM 而非 IAM 來佈建、管理和部署伺服器憑證。在不支援的區域中，您必須使用 IAM 作為憑證管理員。如需了解 ACM 支援哪些區域，請參閱 AWS 一般參考 中的 AWS Certificate Manager 端點和配額。

您可以使用 IAM 標籤鍵值組將自訂屬性新增至伺服器憑證。例如，若要新增伺服器憑證擁有者或系統管理員的相關資訊，請新增標籤鍵 **owner** 和標籤值 **net-eng**。或者，您可以透過新增標籤鍵 **CostCenter** 和標籤值 **1234** 來指定成本中心。您可以使用標籤來控制對資源的存取權，或控制哪些標籤可以連接到資源。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

標記伺服器憑證所需的許可

您必須設定許可，允許 IAM 實體 (使用者或角色) 標記其他伺服器憑證。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListServerCertificateTags`
- `iam:TagServerCertificate`
- `iam:UntagServerCertificate`

允許 IAM 實體 (使用者或角色) 新增、列出或移除伺服器憑證的標籤

將下列陳述式新增至需要管理標籤之 IAM 實體的許可政策。使用您的帳號，並將 *< CertificateName >* 取代為需要管理其標籤的伺服器憑證名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListServerCertificateTags",
        "iam:TagServerCertificate",
        "iam:UntagServerCertificate"
    ],
    "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
```

```
}
```

允許 IAM 實體 (使用者或角色) 將標籤新增至特定伺服器憑證

將下列陳述式新增至 IAM 實體的許可政策，而該實體需要新增 (非移除) 特定伺服器憑證的標籤。

> **Note**
>
> 此 `iam:TagServerCertificate` 動作需要您也包含 `iam:ListServerCertificateTags` 動作。

若要使用此原則，請將 *< CertificateName >* 取代為需要管理其標記之伺服器憑證的名稱。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListServerCertificateTags",
        "iam:TagServerCertificate"
    ],
    "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理伺服器憑證 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接或移除伺服器憑證的標籤。您可以使用 AWS CLI 或 AWS API 來管理伺服器憑證的標籤。

列出目前附加至伺服器憑證 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：AWS list-server-certificate-tags
- AWS API：ListServerCertificateTags

將標籤附加到伺服器憑證 (AWS CLI 或 AWS API)

- AWS CLI：AWS tag-server-certificate

- AWS API：TagServerCertificate

從伺服器憑證 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：AWS untag-server-certificate
- AWS API：UntagServerCertificate

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

標記虛擬 MFA 裝置

您可以使用 IAM 標籤鍵值組將自訂屬性新增至虛擬 MFA 裝置。例如，若要新增使用者虛擬 MFA 裝置的成本中心資訊，您可以新增標籤鍵 **`CostCenter`** 和標籤值 **`1234`**。您可以使用標籤來控制對資源的存取權，或控制哪些標籤可以連接到物件。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

當您擔任角色或聯合使用者時，您也可以使用中的標籤 AWS STS 來新增自訂屬性。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

標記虛擬 MFA 裝置所需的許可

您必須設定許可，允許 IAM 實體 (使用者或角色) 標記虛擬 MFA 裝置。您可在 IAM 政策中指定以下一個或所有 IAM 標籤動作：

- `iam:ListMFADeviceTags`
- `iam:TagMFADevice`
- `iam:UntagMFADevice`

允許 IAM 實體 (使用者或角色) 新增、列出或移除虛擬 MFA 裝置的標籤

將下列陳述式新增至需要管理標籤之 IAM 實體的許可政策。使用您的帳戶並使用需要其標籤受管之虛擬 MFA 裝置的名稱取代 *<MFATokenID>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
```

```
    "Action": [
        "iam:ListMFADeviceTags",
        "iam:TagMFADevice",
        "iam:UntagMFADevice"
    ],
    "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

允許 IAM 實體 (使用者或角色) 將標籤新增至特定虛擬 MFA 裝置

將下列陳述式新增至 IAM 實體的許可政策，而該實體需要新增 (非移除) 特定 MFA 裝置的標籤。

> **Note**
>
> 此 `iam:TagMFADevice` 動作需要您也包含 `iam:ListMFADeviceTags` 動作。

若要使用此政策，請用需要其標籤受管之虛擬 MFA 裝置的名稱取代 *<MFATokenID>*。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListMFADeviceTags",
        "iam:TagMFADevice"
    ],
    "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

或者，您可以使用受 AWS 管政策 (例如 IAM) FullAccess 來提供 IAM 的完整存取權。

管理虛擬 MFA 裝置 (AWS CLI 或 AWS API) 上的標籤

您可以列出、連接或移除虛擬 MFA 裝置的標籤。您可以使用 AWS CLI 或 AWS API 來管理虛擬 MFA 裝置的標籤。

列出目前連結至虛擬 MFA 裝置 (AWS CLI 或 AWS API) 的標籤

- AWS CLI：AWS list-mfa-device-tags
- AWS API：列表 MFA DeviceTags

將標籤附加至虛擬 MFA 裝置 (AWS CLI 或 AWS API)

- AWS CLI：AWS tag-mfa-device
- AWS API: 標籤設備

若要從虛擬 MFA 裝置 (AWS CLI 或 AWS API) 移除標籤

- AWS CLI：AWS untag-mfa-device
- AWS API：無標籤設備

如需將標籤附加至其他 AWS 服務之資源的相關資訊，請參閱這些服務的說明文件。

如需使用標籤來設定多個精密許可搭配 IAM 許可政策的更多資訊，請參閱 IAM 政策元素：變數與標籤。

傳遞工作階段標籤 AWS STS

工作階段標籤是您在 AWS STS中擔任 IAM 角色或與使用者聯合身分時傳遞的鍵/值對。您可以透過身分提供者 (IdP) AWS STS 或透過身分提出或 AWS API 要求來執行此操作。 AWS CLI 當您使用 AWS STS 要求臨時安全登入資料時，您會產生工作階段。工作階段會過期且包含憑證，例如存取金鑰對和工作階段權杖。當您使用工作階段憑證發出後續請求時，請求內容會包含 `aws:PrincipalTag` 內容鍵。您可以在您政策的 `aws:PrincipalTag` 元素中使用 `Condition` 鍵來根據這些標籤允許或拒絕存取。

當您使用暫時憑證發出請求時，您的主體可能會包含一組標籤。這些標籤來自下列來源：

1. 工作階段標籤 — 當您擔任角色或使用 AWS CLI 或 AWS API 聯合使用者時所傳遞的標籤。如需有關這些操作的詳細資訊，請參閱 工作階段標記操作。
2. 傳入的可轉移工作階段標籤 – 這些標籤會從角色鏈中的上一個工作階段繼承。如需詳細資訊，請參閱本主題稍後的 使用工作階段標籤鏈結角色。
3. IAM 標籤 – 連接至 IAM 的標籤擔任角色。

主題

- 工作階段標記操作
- 工作階段標籤的須知事項
- 新增工作階段標籤所需的許可

- 使用傳遞會話標籤 AssumeRole
- 使用 AssumeRoleWith SAML 傳遞工作階段標記
- 使用傳遞會話標籤 AssumeRoleWithWebIdentity
- 使用傳遞會話標籤 GetFederationToken
- 使用工作階段標籤鏈結角色
- 使用 ABAC 的工作階段標籤
- 檢視工作階段標籤 CloudTrail

工作階段標記操作

您可以使用下列 AWS CLI 或中的 AWS API 作業來傳遞工作階段標籤 AWS STS。 AWS Management Console 切換角色功能不允許您傳遞工作階段標籤。

您也可以將工作階段標籤設為可轉移。可轉移標籤會在角色鏈結期間保存。如需詳細資訊，請參閱 使用工作階段標籤鏈結角色。

比較傳遞工作階段標籤的方法

操作	誰可以擔任這個角色	傳遞標籤的方法	設定可轉移標籤的方法
`assume-role` CLI 或 `AssumeRole` API 操作	IAM 使用者或工作階段	`Tags` API 參數或 `--tags` CLI 選項	`TransitiveTagKeys` API 參數或 `--transitive-tag-keys` CLI 選項
`assume-role-with-saml` CLI 或 `AssumeRoleWithSAML` API 操作	使用 SAML 身分提供者進行身分驗證的任何使用者	`PrincipalTag` SAML 屬性	`TransitiveTagKeys` SAML 屬性
`assume-role-with-web-identity` CLI 或 `AssumeRol`	使用 OIDC 提供者驗證的任何使用者	`PrincipalTag` OIDC 令牌	`TransitiveTagKeys` OIDC 令牌

操作	誰可以擔任這個角色	傳遞標籤的方法	設定可轉移標籤的方法
eWithWebIdentity API 操作			
get-federation-token CLI 或 GetFederationToken API 操作	IAM 使用者或根使用者	Tags API 參數或 --tags CLI 選項	不支援

在以下條件下，則支援工作階段標記的操作便可能會失敗：

- 您傳遞的工作階段標籤數超過 50 個。
- 工作階段標籤索引鍵的純文字超過 128 個字元。
- 工作階段標籤值的純文字超過 256 個字元。
- 工作階段政策純文字的總大小超過 2048 個字元。
- 工作階段政策和標籤的合計總封裝大小太大。若操作失敗，錯誤訊息會顯示政策和標籤的合計大小與大小上限的接近程度 (以百分比)。

工作階段標籤的須知事項

使用工作階段標籤前，請檢閱下列工作階段和標籤的相關詳細資訊。

- 使用工作階段標籤時，所有連線至傳遞標籤之身分提供者 (IdP) 之角色的信任政策必須具有 sts:TagSession 許可。對於在信任政策中沒有此許可的角色，AssumeRole 操作即會失敗。
- 當您要求工作階段時，您可以指定主體標籤作為工作階段標籤。標籤會套用至您使用工作階段的憑證所發出的請求。
- 工作階段標籤使用鍵值對。例如，若要將聯絡資訊新增至工作階段，您可以新增工作階段標籤鍵 email 及標籤值 johndoe@example.com。
- 工作階段標籤必須遵循 IAM 和中命名標籤的規則 AWS STS。本主題包含適用於您工作階段標籤的區分大小寫及受限制字首資訊。

- 新的工作階段標籤會覆寫具有相同標籤鍵的現有擔任角色或聯合身分使用者標籤，無論字元大小寫為何。
- 您無法使用傳遞工作階段標籤 AWS Management Console。
- 工作階段標籤僅對目前的工作階段有效。
- 工作階段標籤支援角色鏈結。依預設， AWS STS 不會將標籤傳遞給後續的角色工作階段。但是，您可以將工作階段標籤設為可轉移。可轉移標籤在角色鏈結期間會持續存在，並在角色信任政策評估期間取代相符的 `ResourceTag` 值。如需詳細資訊，請參閱 使用工作階段標籤鏈結角色。
- 您可以使用工作階段標籤來控制對資源的存取，或是控制可傳遞至後續工作階段的標籤。如需詳細資訊，請參閱 IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記。
- 您可以在 AWS CloudTrail 日誌中檢視您工作階段的主體標籤，包括工作階段標籤。如需詳細資訊，請參閱 檢視工作階段標籤 CloudTrail。
- 您必須為每個工作階段標籤傳遞單一值。 AWS STS 不支援多值工作階段標籤。
- 您最多可傳遞 50 個工作階段標籤。 AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。
- AWS 轉換會將傳遞的工作階段原則和工作階段標籤合併為封裝的二進位格式，並有個別限制。如果您超過此限制， AWS CLI 或 AWS API 錯誤訊息會依百分比顯示政策和標籤合併到大小上限的接近程度。

新增工作階段標籤所需的許可

除了符合 API 操作的動作外，您必須在您的政策中具備下列僅許可動作：

```
sts:TagSession
```

> **Important**
>
> 使用工作階段標籤時，連線至身分提供者 (IdP) 之所有角色的角色信任政策必須具有 `sts:TagSession` 許可。任何連接至 IdP 的正在傳遞工作階段標籤之角色，在沒有此許可的情況下，`AssumeRole` 操作即會失敗。如果您不想更新每個角色的角色信任政策，則您可以使用個別的 IdP 執行個體傳遞工作階段標籤。然後，將 `sts:TagSession` 許可僅新增至連線至個別 IdP 的角色。

您可以使用 `sts:TagSession` 動作搭配下列條件金鑰。

- aws:PrincipalTag – 將連接至提出請求主體的標籤，與您在政策中所指定的標籤進行比較。例如，您可以允許主體只有在主體提出具備指定標籤的請求時，才能傳遞工作階段標籤。
- aws:RequestTag – 將請求中傳遞的標籤鍵值對與您在政策中所指定的標籤對進行比較。例如，您可以允許主體傳遞指定工作階段標籤，但僅限指定的值。
- aws:ResourceTag – 將您在政策中所指定的標籤鍵值對與連接到資源的鍵值對進行比較。例如，您可以允許主體僅在其擔任的角色包含指定標籤時，才能傳遞工作階段標籤。
- aws:TagKeys – 將請求中的標籤鍵與您在政策中所指定的鍵進行比較。例如，您可以只允許主體傳遞具備指定標籤鍵的工作階段標籤。此條件索引鍵會限制可傳遞的工作階段標籤組上限。
- sts:TransitiveTagKeys - 將請求中的可轉移工作階段標籤鍵與政策中指定的標籤鍵進行比較。例如，您可以撰寫政策，只允許主體將特定標籤設為可轉移。可轉移標籤會在角色鏈結期間保存。如需詳細資訊，請參閱 使用工作階段標籤鏈結角色。

例如，以下角色信任政策會允許 test-session-tags 使用者擔任連接政策的角色。當該使用者擔任該角色時，他們必須使用 AWS CLI 或 AWS API 來傳遞三個必要的工作階段標籤和必要的外部 ID。此外，使用者可以選擇將 Project 和 Department 標籤設為可轉移。

Example 工作階段標籤的角色信任政策範例

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIamUserAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
            "Condition": {
              "StringLike": {
                    "aws:RequestTag/Project": "*",
                    "aws:RequestTag/CostCenter": "*",
                    "aws:RequestTag/Department": "*"
                },
                "StringEquals": {"sts:ExternalId": "Example987"}
            }
        },
        {
            "Sid": "AllowPassSessionTagsAndTransitive",
            "Effect": "Allow",
            "Action": "sts:TagSession",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-tags"},
```

```
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/Project": "*",
                    "aws:RequestTag/CostCenter": "*"
                },
                "StringEquals": {
                    "aws:RequestTag/Department": [
                        "Engineering",
                        "Marketing"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "sts:TransitiveTagKeys": [
                        "Project",
                        "Department"
                    ]
                }
            }
        }
    ]
}
```

此政策的功能為何？

- AllowIamUserAssumeRole 陳述式允許 test-session-tags 使用者擔任連接政策的角色。當該使用者擔任角色時，他們必須傳遞必要的工作階段標籤和外部 ID。
  - 此陳述式的第一個條件區塊要求使用者傳遞 Project、CostCenter 和 Department 工作階段標籤。由於標籤值在此陳述式中不重要，因此您可以針對標籤值使用萬用字元 (*)。此區塊會確保使用者至少傳遞這三個工作階段標籤。否則，操作會失敗。使用者可傳遞其他標籤。
  - 第二個條件區塊則會要求使用者傳遞值為 Example987 的外部 ID。
- AllowPassSessionTagsAndTransitive 陳述式允許 sts:TagSession 僅許可動作。必須先允許此動作，使用者才能傳遞工作階段標籤。如果您的政策包含第一個陳述式，但沒有第二個陳述式，使用者便無法擔任角色。
  - 此陳述式的第一個條件區塊允許使用者針對 CostCenter 和 Project 工作階段標籤傳遞任何值。您可以使用萬用字元 (*) 做為原則中的標籤值，這會要求您使用StringLike條件運算子。
  - 第二個條件區塊則只會允許使用者針對 Engineering 工作階段標籤傳遞 Marketing 或 Department 值。
  - 第三個條件區塊則會列出您可設為可轉移的標籤組上限。使用者可以選擇將一部分，或是不將任何標籤設為可轉移。使用者無法將其他標籤設為可轉移。您可以新增另一個包含 "Null":

{"sts:TransitiveTagKeys":"false"} 的條件區塊，來要求使用者將其中至少一個標籤設為可轉移。

使用傳遞會話標籤 AssumeRole

AssumeRole作業會傳回一組可用來存取 AWS 資源的暫時認證。您可以使用 IAM 使用者或角色憑證來呼叫 AssumeRole。若要在擔任角色的同時傳遞工作階段標籤，請使用--tags AWS CLI 選項或 Tags AWS API 參數。

若要將標籤設定為可傳遞，請使用--transitive-tag-keys AWS CLI 選項或 TransitiveTagKeys AWS API 參數。可轉移標籤會在角色鏈結期間保存。如需詳細資訊，請參閱使用工作階段標籤鏈結角色。

以下範例顯示使用 AssumeRole 的範例請求。在此範例中，當您擔任 my-role-example 角色時，您可以建立名為 my-session 的工作階段。您可以新增工作階段標籤鍵/值對 Project = Automation、CostCenter = 12345 和 Department = Engineering。您也可以透過指定其鍵，將 Project 和 Department 標籤設為可轉移。

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/my-role-example \
--role-session-name my-session \
--tags Key=Project,Value=Automation Key=CostCenter,Value=12345
 Key=Department,Value=Engineering \
--transitive-tag-keys Project Department \
--external-id Example987
```

使用 AssumeRoleWith SAML 傳遞工作階段標記

AssumeRoleWithSAML 操作是使用 SAML 類型的聯合進行身分驗證。此作業會傳回一組可用來存取 AWS 資源的暫時認證。如需使用 SAML 型聯合進行存取的詳細資訊，請參 AWS Management Console 閱。啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console如需有關 AWS CLI 或 AWS API 存取的詳細資訊，請參閱SAML 2.0 聯合身分。如需為 Active Directory 使用者設定 SAML 聯盟的教學課程，請參閱安全性部落格中的使用中目錄AWS 同盟服務 (ADFS) 的聯合驗證。 AWS

身為管理員，您可以允許公司目錄的成員 AWS 使用此 AWS STS AssumeRoleWithSAML作業聯合到。若要執行此操作，您必須完成下列任務：

1. 將網路配置為適用於 AWS的 SAML 提供者

2. 在 IAM 中建立 SAML 提供者
3. 在 AWS 中為您的聯合身分使用者設定角色及其許可
4. 完成配置 SAML IdP 並為 SAML 身分驗證回應建立聲明

AWS 包括具有認證 end-to-end 經驗的身份提供者，以及其身份解決方案的會話標籤 如要了解如何使用這些身分提供者來設定工作階段標籤，請參閱 將第三方 SAML 解決方案提供者與 AWS。

如要將 SAML 屬性做為工作階段標籤傳遞，請在其中包含 `Attribute` 元素，並將其 `Name` 屬性設為 `https://aws.amazon.com/SAML/Attributes/PrincipalTag:{TagKey}`。使用 `AttributeValue` 元素指定標籤的值。為每個工作階段標籤包含個別的 `Attribute` 元素。

例如，假設您希望將下列身分屬性做為工作階段標籤傳遞：

- `Project:Automation`
- `CostCenter:12345`
- `Department:Engineering`

如要傳遞這些屬性，請在您的 SAML 聲明中包含下列元素。

Example SAML 聲明的程式碼片段範例

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Automation</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Department">
  <AttributeValue>Engineering</AttributeValue>
</Attribute>
```

如要將前述標籤設為可轉移，請在其中包含另一個 `Attribute` 元素，並將其 `Name` 屬性設為 `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys`。可轉移標籤會在角色鏈結期間保存。如需詳細資訊，請參閱 使用工作階段標籤鏈結角色。

如要將 `Project` 和 `Department` 標籤設為可轉移，請使用以下多值屬性：

Example SAML 聲明的程式碼片段範例

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>Department</AttributeValue>
</Attribute>
```

使用傳遞會話標籤 AssumeRoleWithWebIdentity

使用 OpenID Connect (OIDC) 相容的聯盟來驗證作業。`AssumeRoleWithWebIdentity`此作業會傳回一組可用來存取 AWS 資源的暫時認證。如需有關使用 Web 身分同盟進行 AWS Management Console 存取的詳細資訊，請參閱OIDC 聯盟。

如要從 OpenID Connect (OIDC) 傳遞工作階段標籤，您必須在 JSON Web 權杖 (JWT) 中包含工作階段標籤。在您提交 `https://aws.amazon.com/ tags` 請求時，您必須在權杖中的 `AssumeRoleWithWebIdentity` 命名空間內包含工作階段標籤。若要進一步了解 OIDC 權杖和要求，請參閱《Amazon Cognito 開發人員指南》中的搭配使用者集區使用權杖。

例如，以下解碼後的 JWT 會使用權杖搭配 `Project`、`CostCenter` 和 `Department` 工作階段標籤來呼叫 `AssumeRoleWithWebIdentity`。權杖也會將 `Project` 和 `CostCenter` 標籤設為可轉移。可轉移標籤會在角色鏈結期間保存。如需詳細資訊，請參閱 使用工作階段標籤鏈結角色。

Example 解碼後的 JSON Web 權杖範例

```
{
    "sub": "johndoe",
    "aud": "ac_oic_client",
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags": {
        "principal_tags": {
            "Project": ["Automation"],
            "CostCenter": ["987654"],
            "Department": ["Engineering"]
        },
        "transitive_tag_keys": [
            "Project",
            "CostCenter"
```

```
        ]
    }
}
```

使用傳遞會話標籤 GetFederationToken

GetFederationToken 允許您聯合您的使用者。此作業會傳回一組可用來存取 AWS 資源的暫時認證。若要將標籤新增至聯合使用者工作階段，請使用--tags AWS CLI 選項或 Tags AWS API 參數。當您使用時，您無法將工作階段標籤設為可轉移GetFederationToken，因為您無法使用臨時憑證來擔任角色。在這種情況下，您無法使用角色鏈結。

以下範例顯示使用 GetFederationToken 的範例請求。在此範例中，當您請求權杖時，您會建立名為 my-fed-user 的工作階段。您可以新增工作階段標籤鍵/值對 Project = Automation 和 Department = Engineering。

Example GetFederationToken CLI 要求範例

```
aws sts get-federation-token \
--name my-fed-user \
--tags key=Project,value=Automation key=Department,value=Engineering
```

當您使用 GetFederationToken 操作傳回的暫時憑證時，工作階段的主體標籤會包含使用者的標籤和傳遞的工作階段標籤。

使用工作階段標籤鏈結角色

您可以擔任一個角色，然後使用暫時憑證來擔任另一個角色。您可以從一個工作階段繼續前往另一個工作階段。這稱為角色鏈結。當您在擔任角色期間傳遞工作階段標籤時，您可以將鍵設為可轉移。這可確保會在角色鏈中，將這些工作階段標籤傳遞到後續的工作階段。您無法將角色標籤設為可轉移。如要將這些標籤傳遞至後續工作階段，請將這些標籤指定為工作階段標籤。

> **Note**
>
> 可轉移標籤在角色鏈結期間會持續存在，並在角色信任政策評估期間取代相符的 ResourceTag 值。

下列範例顯示如何將工作階段標籤、轉移標籤和角色標籤 AWS STS 傳遞至角色鏈中的後續工作階段。

在此範例角色鏈結案例中，您可以使用中的 IAM 使用者存取金鑰 AWS CLI 來承擔名為Role1的角色。您接著會使用產生的工作階段憑證來擔任名為 Role2 的第二個角色。您接著可以使用第二個工作階段憑證來擔任名為 Role3 的第三個角色。這些請求會以三個不同操作的形式發生。每個角色都已在 IAM 中加上標籤。而在每個請求期間，您會傳遞其他工作階段標籤。

鏈結角色時，您可以確保將先前工作階段的標籤保存到後續工作階段。若要執行此操作，請使用 `assume-role` CLI 命令；您必須將標籤作為工作階段標籤傳遞，並將標籤設為可轉移。您將標籤 `Star = 1` 設為工作階段標籤。命令還會將標籤 `Heart = 1` 連接到角色，並在您使用工作階段時作為主體標籤套用。但是，您也希望 `Heart = 1` 標籤能自動傳遞到第二個或第三個工作階段。若要執行此作業，您可以手動將其做為工作階段標籤包含在其中。產生的工作階段主體標籤包含這兩個標籤，並將其設定為可轉移。

您可以使用下列 AWS CLI 命令來執行此要求：

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role1 \
--role-session-name Session1 \
--tags Key=Star,Value=1 Key=Heart,Value=1 \
--transitive-tag-keys Star Heart
```

然後，您可以使用該工作階段的憑證來擔任 Role2。命令會將標籤 Sun = 2 連接至第二個角色，並在您使用第二個工作階段時作為主體標籤套用。Heart 和 Star 標籤會繼承自第一個工作階段中的可轉移工作階段。第二個工作階段所產生的主體標籤是 Heart = 1、Star = 1 及 Sun = 2。Heart 和 Star 會繼續處於可轉移狀態。連接至 Role2 的 Sun 標籤並未標記為可轉移，因為其不是工作階段標籤。未來的工作階段不會繼承此標籤。

您可以使用下列 AWS CLI 命令執行第二個要求：

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role2 \
--role-session-name Session2
```

您接著會使用第二個工作階段憑證來擔任 Role3。第三個工作階段的主體標籤是來自任何新的工作階段標籤、繼承的可轉移工作階段標籤，以及角色標籤。第二個工作階段上的 Heart = 1 和 Star = 1 標籤是繼承自第一個工作階段中的可轉移工作階段標籤。如您嘗試傳遞 Sun = 2 工作階段標籤，操作將會失敗。繼承的 Star = 1 工作階段標籤會覆寫角色的 Star = 3 標籤。在角色鏈結過程中，進行角色

信任政策評估之後，可轉移標籤的值會覆寫與 ResourceTag 值相符的角色。在此範例中，若 Role3 在角色信任政策中使用 Star 作為 ResourceTag，並將 ResourceTag 值設定為來自呼叫角色工作階段的可轉移標籤值。角色的 Lightning 標籤也會套用到第三個工作階段，且並未設為可轉移。

您可以使用下列 AWS CLI 命令執行第三個要求：

Example AssumeRole CLI 要求範例

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role3 \
--role-session-name Session3
```

使用 ABAC 的工作階段標籤

屬性類型存取控制 (ABAC) 會使用一種授權策略，根據標籤屬性定義許可。

如果您的公司使用 OIDC 或 SAML 類型的身分提供者 (IdP) 來管理使用者身分，您可以將您的聲明設為將工作階段標籤傳遞到 AWS。例如，對於公司使用者識別，當您的員工聯合到時 AWS，會將其屬性 AWS 套用至其產生的主參與者。您接著可以使用 ABAC 來根據這些屬性允許或拒絕許可。如需詳細資訊，請參閱 IAM 教學課程：針對 ABAC 使用 SAML 工作階段標記。

如需有關 IAM Identity Center 搭配 ABAC 使用的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的存取控制的屬性。

檢視工作階段標籤 CloudTrail

您可以使用 AWS CloudTrail 來檢視用來擔任角色或同盟使用者的要求。記 CloudTrail錄檔案包含指定角色或同盟使用者工作階段之主要標籤的相關資訊。如需詳細資訊，請參閱 使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail。

例如，假設您提出 AWS STS AssumeRoleWithSAML要求、傳遞工作階段標籤，並將這些標籤設定為可傳遞。您可以在 CloudTrail 日誌中找到以下信息。

Example AssumeRoleWithSAML 記錄檔 CloudTrail 範例

```
    "requestParameters": {
        "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyRoleSessionName",
        "principalTags": {
            "CostCenter": "987654",
            "Project": "Unicorn"
```

```
        },
        "transitiveTagKeys": [
            "CostCenter",
            "Project"
        ],
        "durationSeconds": 3600,
        "roleArn": "arn:aws:iam::123456789012:role/SAMLTestRoleShibboleth",
        "principalArn": "arn:aws:iam::123456789012:saml-provider/Shibboleth"
    },
```

您可以檢視下列範例 CloudTrail 記錄，以檢視使用工作階段標記的事件。

- CloudTrail 記錄檔中 AWS STS 角色鏈結 API 事件的範例
- CloudTrail 記錄檔中的 SAML AWS STS API 事件範例
- 記錄檔中的範例 OIDC AWS STS API 事件 CloudTrail

使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail

IAM 與 AWS STS 服務整合 AWS CloudTrail，可提供 IAM 使用者或角色所採取之動作記錄的服務。CloudTrail 擷取 IAM 和 AWS STS 作為事件的所有 API 呼叫，包括來自主控台和 API 呼叫的呼叫。如果您建立追蹤，您可以啟用將 CloudTrail 事件持續傳遞到 Amazon S3 儲存貯體。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail主控台中最近的事件。您可以使 CloudTrail 用取得有關向 IAM 或發出之請求的資訊 AWS STS。例如，您可以檢視發出請求的 IP 地址、發出請求的人員及時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

主題

- IAM 和 AWS STS 資訊 CloudTrail
- 記錄 IAM 和 AWS STS API 請求
- 記錄其他 AWS 服務的 API 請求
- 記錄使用者登入事件
- 記錄暫時憑證的登入事件
- CloudTrail記錄檔中的 IAM API 事件範例
- CloudTrail記錄檔中的 AWS STS API 事件範例
- CloudTrail 日誌中的範例登入事件

- IAM 角色信任政策行為

IAM 和 AWS STS 資訊 CloudTrail

CloudTrail 在您創建帳戶 AWS 帳戶 時啟用。當活動發生在 IAM 中 AWS STS，或者，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以查看，搜索和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需您的事件的持續記錄 AWS 帳戶，包括 IAM 的事件 AWS STS，並建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 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 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定的 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail記錄檔

所有 IAM 和 AWS STS 動作均由 IAM API 參考 CloudTrail 和 API 參考記錄，並記錄在其中。AWS Security Token Service

記錄 IAM 和 AWS STS API 請求

CloudTrail 將所有經過驗證的 API 請求記錄到 IAM 和 AWS STS API 操作。 CloudTrail 也會將未驗證的要求記錄至 AWS STS 動作`AssumeRoleWithWebIdentity`，`AssumeRoleWithSAML`並記錄身分識別提供者提供的資訊。但是，某些未經驗證的 AWS STS 請求可能不會被記錄，因為它們不符合足夠有效以被信任為合法請求的最低期望。

您可以使用記錄的資訊，將具有假定角色的聯合使用者所發出的呼叫對應回原始外部聯合呼叫者。在的情況下`AssumeRole`，您可以將呼叫對應回原始 AWS 服務或原始使用者的帳戶。 CloudTrail 記錄項目中 JSON 資料的`userIdentity`區段包含您將要AssumeRole*求對應至特定同盟使用者所需的資訊。若要取得更多資訊，請參閱AWS CloudTrail 使用者指南中的使用者 CloudTrail userIdentity 元素。

例如，對 IAM `CreateUser` `DeleteRole` `ListGroups`、和其他 API 操作的呼叫全部記錄 CloudTrail。

本主題稍後會提供此類日誌項目的範例。

記錄其他 AWS 服務的 API 請求

其他 AWS 服務 API 作業的驗證要求會由記錄 CloudTrail，而這些記錄項目包含產生要求者的相關資訊。

例如：假設您發出請求，要求列出 Amazon EC2 執行個體或建立 AWS CodeDeploy 部署群組。發出請求的人員或服務相關詳細資訊都會包含在該請求的日誌項目中。此資訊可協助您判斷要求是由 IAM 使用者 AWS 帳戶根使用者、角色還是其他 AWS 服務提出。

如需有關 CloudTrail 記錄項目中使用者身分資訊的詳細資訊，請參閱使AWS CloudTrail 用者指南中的使用者 userIdentity 元素。

記錄使用者登入事件

CloudTrail 將登入事件記錄到 AWS Management Console、 AWS 討論區和 AWS Marketplace. CloudTrail記錄 IAM 使用者和聯合身分使用者的成功和失敗登入嘗試。

若要檢視成功和未成功的 root 使用者登入 CloudTrail 事件範例，請參閱《使用指南》中針對 root 使用AWS CloudTrail 者的範例事件記錄。

安全性最佳做法是，當登入失敗是由於錯誤的使用者名稱造成時，不 AWS 會記錄輸入的 IAM 使用者名稱文字。使用者名稱文字是由值 HIDDEN_DUE_TO_SECURITY_REASONS 遮蓋的。如需此範例，請參閱本主題後述的因使用者名稱不正確以致登入失敗的事件範例。因為這類失敗可能是使用者錯誤所造成，所以會隱蔽使用者名稱文字。記錄這些錯誤可能會公開潛在的敏感資訊。例如：

- 您不小心在使用者名稱方塊中輸入密碼。
- 您可以選擇其中一個登入頁面的連結 AWS 帳戶，然後輸入其他帳號 AWS 帳戶。
- 您忘記了正在登入的帳戶，並且無意中輸入了您的個人電子郵件帳戶的帳戶名稱，銀行登入識別碼或其他私有 ID。

記錄暫時憑證的登入事件

當主體要求暫時認證時，主體類型會決定 CloudTrail記錄事件的方式。當主體擔任另一個帳戶中的角色時，情況會很複雜。有多個 API 呼叫會執行與角色跨帳戶操作相關的操作。首先，主體會呼叫 AWS STS API 來擷取暫時登入資料。該操作記錄在呼叫帳戶和執行 AWS STS 操作的帳戶中。然後，主體會使用該角色，在擔任角色的帳戶中執行其他 API 呼叫。

您可以使用角色信任政策中的 sts:SourceIdentity 條件金鑰，請求使用者在擔任角色時指定身分。例如，您可以請求 IAM 使用者將自己的使用者名稱指定為其來源身分。這可以協助您判斷哪位

使用者在 AWS中執行了特定動作。如需詳細資訊，請參閱 sts:SourceIdentity。您亦可以使用 sts:RoleSessionName，請求使用者在擔任角色時指定工作階段名稱。這可協助您區分檢閱 AWS CloudTrail 記錄檔時，不同主參與者所使用之角色的角色工作階段。

下表顯示如何為每個產生臨時認證的 AWS STS API CloudTrail 記錄不同的使用者身分資訊。

主體類型	STS API	來電者帳戶的 CloudTrail 日誌中的用戶身份	假定角色帳戶的 CloudTrail 記錄中的使用者身分	角色後續 API 呼叫的 CloudTrail 記錄檔中的使用者身分
AWS 帳戶根使用者 認證	GetSessionToken	根使用者身分	角色擁有者帳戶與呼叫帳戶相同	根使用者身分
IAM 使用者	GetSessionToken	IAM 使用者身分	角色擁有者帳戶與呼叫帳戶相同	IAM 使用者身分
IAM 使用者	GetFederationToken	IAM 使用者身分	角色擁有者帳戶與呼叫帳戶相同	IAM 使用者身分
IAM 使用者	AssumeRole	IAM 使用者身分	帳戶號碼和主體 ID（如果是使用者），或 AWS 服務主體	僅限角色身分 (非使用者)
外部驗證的使用者	AssumeRoleWith薩姆爾	N/A	SAML 使用者身分	僅限角色身分 (非使用者)
外部驗證的使用者	AssumeRoleWithWebIdentity	N/A	OIDC/Web 使用者身分	僅限角色身分 (非使用者)

CloudTrail 如果動作對資源沒有任何變異影響，則會將動作視為唯讀。記錄唯讀事件時，會 CloudTrail 標記記錄檔中的`responseElements`資訊。當 CloudTrail 記錄不是唯讀的事件時，完整的`responseElements`會顯示在記錄項目中。但是，對於 AWS STS API`AssumeRole`，和 `AssumeRoleWithSAML`、`AssumeRoleWithWebIdentity`，即使它們以唯讀方式記錄，也 CloudTrail 會在這些 API 的日誌`responseElements`中包含完整內容。

下表顯示產生臨時認證之每個 AWS STS API 的 CloudTrail 記錄檔`responseElements`和`readOnly`資訊的方式。

STS API	回應元素資訊	唯讀
AssumeRole	已包含	true
AssumeRoleWith薩姆爾	已包含	true
AssumeRoleWithWebIdentity	已包含	true
GetFederationToken	已包含	false
GetSessionToken	已包含	false

CloudTrail記錄檔中的 IAM API 事件範例

CloudTrail 記錄檔包含使用 JSON 格式化的事件。一個 API 事件代表單一 API 請求，並包含主體、請求動作、任何參數以及動作的日期和時間等資訊。

CloudTrail 記錄檔中的 IAM API 事件範例

下列範例顯示針對 IAM `GetUserPolicy` 動作提出的請求的 CloudTrail 記錄項目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn": "arn:aws:iam::444455556666:user/JaneDoe",
    "accountId": "444455556666",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "userName": "JaneDoe",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2014-07-15T21:39:40Z"
      }
    },
    "invokedBy": "signin.amazonaws.com"
  },
```

```
  "eventTime": "2014-07-15T21:40:14Z",
  "eventSource": "iam.amazonaws.com",
  "eventName": "GetUserPolicy",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "signin.amazonaws.com",
  "userAgent": "signin.amazonaws.com",
  "requestParameters": {
    "userName": "JaneDoe",
    "policyName": "ReadOnlyAccess-JaneDoe-201407151307"
  },
  "responseElements": null,
  "requestID": "9EXAMPLE-0c68-11e4-a24e-d5e16EXAMPLE",
  "eventID": "cEXAMPLE-127e-4632-980d-505a4EXAMPLE"
}
```

從這個事件資訊，您可以判斷提出請求是為了取得使用者 ReadOnlyAccess-JaneDoe-201407151307 的一個名為 JaneDoe 的使用者政策，如 requestParameters 元素中所指定。您也可以看到該請求由名為 JaneDoe 的 IAM 使用者於 2014 年 7 月 15 日下午 9:40 (UTC) 提出。在這種情況下，請求源於 AWS Management Console，正如您可以從userAgent元素中看到的那樣。

CloudTrail記錄檔中的 AWS STS API 事件範例

CloudTrail 記錄檔包含使用 JSON 格式化的事件。一個 API 事件代表單一 API 請求，並包含主體、請求動作、任何參數以及動作的日期和時間等資訊。

CloudTrail記錄檔中的跨帳戶 AWS STS API 事件範例

帳戶 777788889999 JohnDoe 中指定的身分與存取權與存取權管理使用者會呼叫此 AWS STS AssumeRole動作來擔任帳戶 111122223333 中的角色。EC2-dev帳戶管理員會要求使用者在擔任角色時，將來源身分設定為等於其使用者名稱。使用者傳入 JohnDoe 的來源身分值。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "arn": "arn:aws:iam::777788889999:user/JohnDoe",
    "accountId": "777788889999",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "JohnDoe"
  },
```

```
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2023, 4:07:39 PM"
      },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
    },
  "sourceIdentity": "JohnDoe"
  },
  "resources": [
    {
      "ARN": "arn:aws:iam::111122223333:role/EC2-dev",
      "accountId": "111122223333",
      "type": "AWS::IAM::Role"
    }
  ],
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

第二個範例顯示相同要求的假設角色帳戶 (111122223333) CloudTrail 記錄項目。

```
{
  "eventVersion": "1.05",
```

```
  "userIdentity": {
    "type": "AWSAccount",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "accountId": "777788889999"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "JohnDoe-EC2-dev",
    "sourceIdentity": "JohnDoe",
    "serialNumber": "arn:aws:iam::777788889999:mfa"
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2014, 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:JohnDoe-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/JohnDoe-EC2-dev"
      },
  "sourceIdentity": "JohnDoe"
  },
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
}
```


CloudTrail 記錄檔中 AWS STS 角色鏈結 API 事件的範例

下列範例顯示 Doe 在帳戶 1111111111 中所提出之要求的 CloudTrail 記錄項目。John 以前用過他的 JohnDoe 使用者擔任 JohnRole1 角色。對此請求而言，他使用該角色的憑證擔任 JohnRole2 角色。這就是所謂的角色鏈接。他在擔任 JohnDoe1 角色時所設定的來源身分將保留在請求中以擔任 JohnRole2。如果 John 嘗試在擔任角色時設定不同的來源身分，則請求將會遭到拒絕。John 將兩個工作階段標籤傳遞給請求。並將這兩個標籤設為轉移。因為 John 在擔任 Department 時將其設為

轉移，所以此請求繼承 JohnRole1 標籤也為轉移。如需有關來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。如需有關角色鏈結中轉移索引鍵的詳細資訊，請參閱 使用工作階段標籤鏈結角色。


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIN5ATK5U7KEXAMPLE:JohnRole1",
        "arn": "arn:aws:sts::111111111111:assumed-role/JohnDoe/JohnRole1",
        "accountId": "111111111111",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-10-02T21:50:54Z"
            },
                "sessionIssuer": {
                "type": "Role",
               "principalId": "AROAIN5ATK5U7KEXAMPLE",
                "arn": "arn:aws:iam::111111111111:role/JohnRole1",
                "accountId": "111111111111",
                "userName": "JohnDoe"
            },
            "sourceIdentity": "JohnDoe"
        }
    },
    "eventTime": "2019-10-02T22:12:29Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "123.145.67.89",
    "userAgent": "aws-cli/1.16.248 Python/3.4.7
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.12.239",
    "requestParameters": {
        "incomingTransitiveTags": {
            "Department": "Engineering"
        },
        "tags": [
            {
                "value": "johndoe@example.com",
                "key": "Email"
            },
```

```
            {
                "value": "12345",
                "key": "CostCenter"
            }
        ],
        "roleArn": "arn:aws:iam::111111111111:role/JohnRole2",
        "roleSessionName": "Role2WithTags",
        "sourceIdentity": "JohnDoe",
        "transitiveTagKeys": [
            "Email",
            "CostCenter"
        ],
        "durationSeconds": 3600
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
            "expiration": "Oct 2, 2019, 11:12:29 PM",
            "sessionToken": "AgoJb3JpZ2luX2VjEB4aCXVzLXdlc3QtMSJHMEXAMPLETOKEN
+//rJb8Lo30mFc5MlhFCEbubZvEj0wHB/mDMwIgSEe9gk/Zjr09tZV7F1HDTMhmEXAMPLETOKEN/iEJ/
rkqngII9//////////
ARABGgw0MjgzMDc4NjM5NjYiDLZjZFKwP4qxQG5sFCryASO4UPz5qE97wPPH1eLMvs7CgSDBSWfonmRTCfokm2FN1+hWUdQ
+C+WKFZb701eiv9J5La2EXAMPLETOKEN/c7S5Iro1WUJ0q3Cxuo/8HUoSxVhQHM7zF7mWWLhXLEQ52ivL
+F6q5dpXu4aTFedpMfnJa8JtkWwG9x1Axj0Ypy2ok8v5unpQGWych1vwdvj6ez1Dm8Xg1+qIzXILiEXAMPLETOKEN/
vQGqu8H+nxp3kabcrtOvTFTvxX6vsc8OGwUfHhzAfYGEXAMPLETOKEN/
L6v1yMM3B1OwFOrQBno1HEjf1oNI8RnQiMNFdUOtwYj7HUZIOCZmjfN8PPHq77N7GJl9lzvIZKQA0Owcjg
+mc78zHCj8y0siY8C96paEXAMPLETOKEN/
E3cpksxWdgs91HRzJWScjN2+r2LTGjYhyPqcmFzzo2mCE7mBNEXAMPLETOKEN/oJy
+2o83YNW5tOiDmczgDzJZ4UKR84yGYOMfSnF4XcEJrDgAJ3OJFwmTcTQICAlSwLEXAMPLETOKEN"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROAIFR7WHDTSOYQYHFUE:Role2WithTags",
            "arn": "arn:aws:sts::111111111111:assumed-role/test-role/Role2WithTags"
        },
    "sourceIdentity": "JohnDoe"
    },
    "requestID": "b96b0e4e-e561-11e9-8b3f-7b396EXAMPLE",
    "eventID": "1917948f-3042-46ec-98e2-62865EXAMPLE",
    "resources": [
        {
            "ARN": "arn:aws:iam::111111111111:role/JohnRole2",
            "accountId": "111111111111",
            "type": "AWS::IAM::Role"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

CloudTrail 記錄檔中的 AWS 服務 AWS STS API 事件範例

下列範例顯示使用服務角色權限呼叫其他 AWS 服務 API 的服務所發出之要求的 CloudTrail 記錄項目。它顯示了在 CloudTrail 帳戶 777788889999 中提出的請求的日誌條目。

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAQRSTUVWXYZEXAMPLE:devdsk",
    "arn": "arn:aws:sts::777788889999:assumed-role/AssumeNothing/devdsk",
    "accountId": "777788889999",
    "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2016-11-14T17:25:26Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAQRSTUVWXYZEXAMPLE",
        "arn": "arn:aws:iam::777788889999:role/AssumeNothing",
        "accountId": "777788889999",
        "userName": "AssumeNothing"
      }
    }
  },
  "eventTime": "2016-11-14T17:25:45Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "DeleteBucket",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.1",
  "userAgent": "[aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67]",
  "requestParameters": {
    "bucketName": "my-test-bucket-cross-account"
  },
  "responseElements": null,
```

```
  "requestID": "EXAMPLE463D56D4C",
  "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "777788889999"
}
```

CloudTrail 記錄檔中的 SAML AWS STS API 事件範例

下列範例顯示針對 AWS STS AssumeRoleWithSAML動作所發出之請求的 CloudTrail 記錄項目。此請求包含 SAML 屬性 CostCenter 和 Project，這些屬性透過 SAML 聲明以工作階段標籤的形式傳遞。這些標籤設為轉移，以便能在角色鏈接藍本中繼續。請求包括可選的 API 參數DurationSeconds，以 CloudTrail 日誌durationSeconds中的形式表示，並設置為1800秒。此請求亦包含 SAML 屬性 sourceIdentity，會在 SAML 聲明中傳遞。如果有人使用產生的角色工作階段憑證來擔任另一個角色，則此來源身分會持續存在。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "SAMLUser",
        "principalId": "SampleUkh1i4+ExamplexL/jEvs=:SamlExample",
        "userName": "SamlExample",
        "identityProvider": "bdGOnTesti4+ExamplexL/jEvs="
    },
    "eventTime": "2023-08-28T18:30:58Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoleWithSAML",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "aws-internal/3 aws-sdk-java/1.12.479
 Linux/5.10.186-157.751.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/17.0.7+11 java/17.0.7
 kotlin/1.3.72 vendor/Amazon.com_Inc. cfg/retry-mode/standard",
    "requestParameters": {
        "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyAssignedRoleSessionName",
        "sourceIdentity": "MySAMLUser",
        "principalTags": {
            "CostCenter": "987654",
            "Project": "Unicorn",
            "Department": "Engineering"
        },
        "transitiveTagKeys": [
            "CostCenter",
```

```
            "Project"
        ],
        "roleArn": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
        "principalArn": "arn:aws:iam::444455556666:saml-provider/Shibboleth",
        "durationSeconds": 1800
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "<encoded session token blob>",
            "expiration": "Aug 28, 2023, 7:00:58 PM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROAD35QRSTUVWEXAMPLE:MyAssignedRoleSessionName",
            "arn": "arn:aws:sts::444455556666:assumed-role/SAMLTestRoleShibboleth/
MyAssignedRoleSessionName"
        },
        "packedPolicySize": 1,
        "subject": "SamlExample",
        "subjectType": "transient",
        "issuer": "https://server.example.com/idp/shibboleth",
        "audience": "https://signin.aws.amazon.com/saml",
        "nameQualifier": "bdGOnTesti4+ExamplexL/jEvs=",
        "sourceIdentity": "MySAMLUser"
    },
    "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
    "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "444455556666",
            "type": "AWS::IAM::Role",
            "ARN": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth"
        },
        {
            "accountId": "444455556666",
            "type": "AWS::IAM::SAMLProvider",
            "ARN": "arn:aws:iam::444455556666:saml-provider/test-saml-provider"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management",
```

```
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "sts.us-east-2.amazonaws.com"
    }
}
```

記錄檔中的範例 OIDC AWS STS API 事件 CloudTrail

下列範例顯示針對 AWS STS AssumeRoleWithWebIdentity動作所發出之請求的 CloudTrail 記錄項目。此請求包含屬性 CostCenter 和 Project，這些屬性透過身分提供者權杖以工作階段標籤的形式傳遞。這些標籤設為轉移，以便能在角色鏈接中繼續。請求包含從身分提供者權杖的 sourceIdentity 屬性。如果有人使用產生的角色工作階段憑證來擔任另一個角色，則此來源身分會持續存在。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "WebIdentityUser",
    "principalId": "accounts.google.com:<id-of-
application>.apps.googleusercontent.com:<id-of-user>",
    "userName": "<id of user>",
    "identityProvider": "accounts.google.com"
  },
  "eventTime": "2016-03-23T01:39:51Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRoleWithWebIdentity",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.3.23 Python/2.7.6 Linux/2.6.18-164.el5",
  "requestParameters": {
    "sourceIdentity": "MyWebIdentityUser",
    "durationSeconds": 3600,
    "roleArn": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
    "roleSessionName": "MyAssignedRoleSessionName"
        "principalTags": {
            "CostCenter": "24680",
            "Project": "Pegasus"
        },
        "transitiveTagKeys": [
            "CostCenter",
            "Project"
        ],
```

```
  },
  "responseElements": {
    "provider": "accounts.google.com",
    "subjectFromWebIdentityToken": "<id of user>",
    "sourceIdentity": "MyWebIdentityUser",
    "audience": "<id of application>.apps.googleusercontent.com",
    "credentials": {
      "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
      "expiration": "Mar 23, 2016, 2:39:51 AM",
      "sessionToken": "<encoded session token blob>"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:MyAssignedRoleSessionName",
      "arn": "arn:aws:sts::444455556666:assumed-role/FederatedWebIdentityRole/
MyAssignedRoleSessionName"
    }
  },
  "resources": [
    {
      "ARN": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
      "accountId": "444455556666",
      "type": "AWS::IAM::Role"
    }
  ],
  "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
  "eventID": "bEXAMPLE-0b30-4246-b28c-e3da3EXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "444455556666"
}
```

CloudTrail 日誌中的範例登入事件

CloudTrail 記錄檔包含使用 JSON 格式化的事件。登入事件代表單一登入請求，並包含登入主體、區域以及動作的日期和時間等資訊。

CloudTrail 記錄檔中的登入成功事件範例

下列範例顯示成功登入事件的 CloudTrail 記錄項目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
```

```
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::111122223333:user/JohnDoe",
    "accountId": "111122223333",
    "userName": "JohnDoe"
  },
  "eventTime": "2014-07-16T15:49:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.110",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Success"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/s3/  ",
    "MFAUsed": "No"
  },
  "eventID": "3fcfb182-98f8-4744-bd45-10a395ab61cb"
}
```

若要取得有關 CloudTrail 記錄檔中包含的資訊的詳細資訊，請參閱《AWS CloudTrail 使用指南》中的〈CloudTrail 事件參考〉。

CloudTrail 記錄檔中的登入失敗事件範例

下列範例顯示失敗登入事件的 CloudTrail 記錄項目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::111122223333:user/JaneDoe",
    "accountId": "111122223333",
    "userName": "JaneDoe"
  },
  "eventTime": "2014-07-08T17:35:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
```

```
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.100",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "errorMessage": "Failed authentication",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/sns",
    "MFAUsed": "No"
  },
  "eventID": "11ea990b-4678-4bcd-8fbe-62509088b7cf"
}
```

根據此資訊，您可以判斷嘗試登入的人是名為 JaneDoe 的 IAM 使用者 ，如 userIdentity 元素所示。您還可以看到登入嘗試失敗，如 responseElements 元素所示。您可以看到 JaneDoe 於 UTC 時間 2014 年 7 月 8 日下午 5:35 嘗試登入 Amazon SNS 主控台。

因使用者名稱不正確以致登入失敗的事件範例

下列範例顯示因使用者輸入錯誤的使用者名稱而導致登入失敗事件的 CloudTrail 記錄項目。 AWS 遮罩userName文字，HIDDEN_DUE_TO_SECURITY_REASONS以協助防止暴露潛在的敏感資訊。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "accountId": "123456789012",
    "accessKeyId": "",
    "userName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "eventTime": "2015-03-31T22:20:42Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "errorMessage": "No username found in supplied account",
  "requestParameters": null,
```

```
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs
%23&isauthcode=true",
    "MobileVersion": "No",
    "MFAUsed": "No"
  },
  "eventID": "a7654656-0417-45c6-9386-ea8231385051",
  "eventType": "AwsConsoleSignin",
  "recipientAccountId": "123456789012"
}
```

IAM 角色信任政策行為

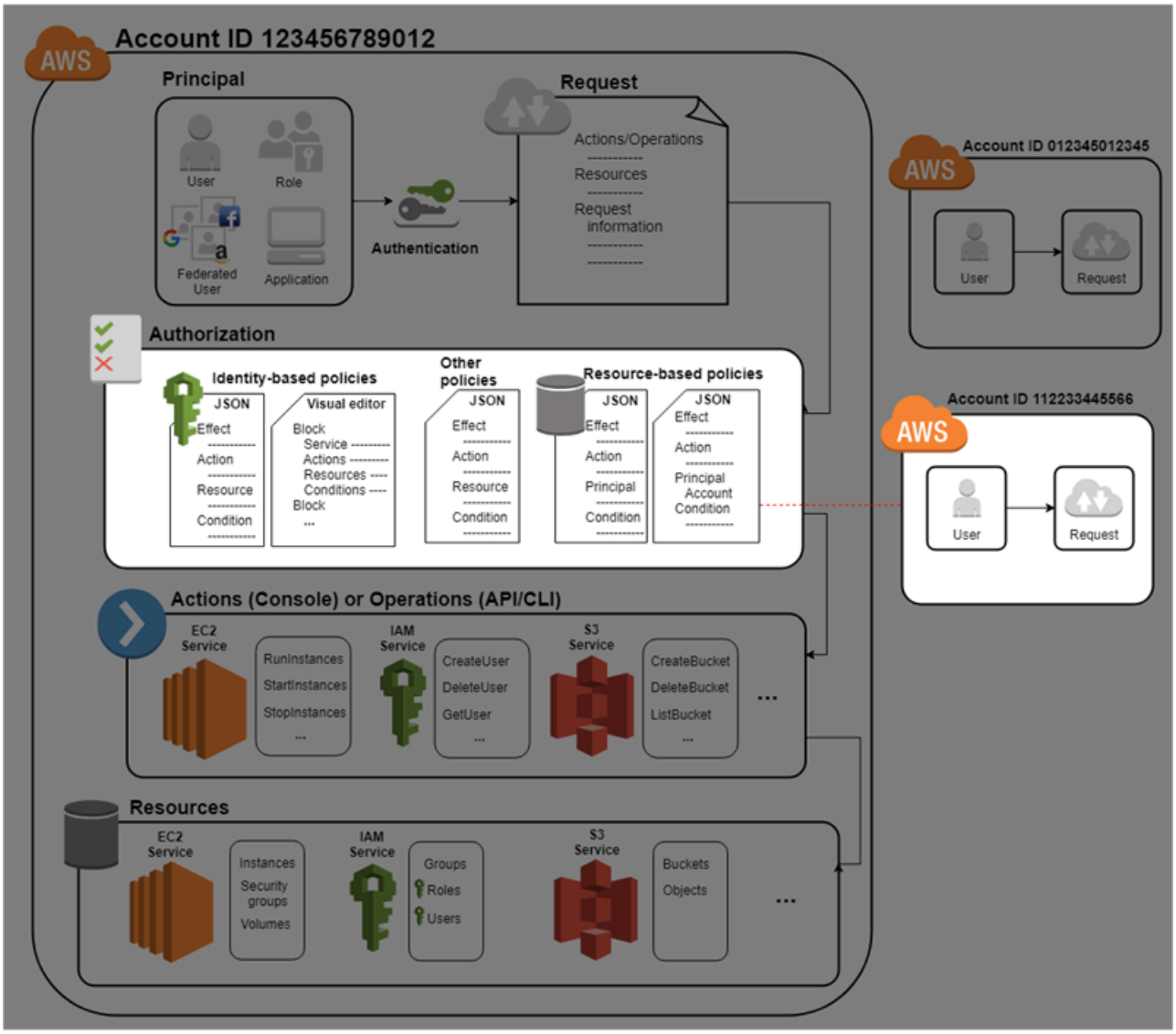
2022 年 9 月 21 日，對 IAM 角色信任政策行為進行 AWS 了變更，以在角色擔任自身時要求角色信任政策中明確允許。舊版行為中的 IAM 角色允許清單具有適用於`AssumeRole`事件 explicitTrustGrant 的 additionalEventData 欄位。當舊版允許清單上的角色假設自己使用舊版行為時，值為 false。`explicitTrustGrant`當舊版允許清單上的角色假設自己，但角色信任原則行為已更新為明確允許角色假設自己時，的值`explicitTrustGrant`為 true。

舊版行為的允許清單中只有極少數 IAM 角色，而且只有這些角色假設自己時，此欄位才會出現在這些角色的 CloudTrail 記錄中。在大多數情況下，IAM 角色不需要假定自己。 AWS 建議您更新處理程序、程式碼或組態，以移除此行為，或更新角色信任原則以明確允許此行為。如需詳細資訊，請參閱宣告 IAM 角色信任政策行為的更新。

AWS 資源存取管理

AWS Identity and Access Management (IAM) 是可協助您安全地控制 AWS 資源存取的 Web 服務。當主體在中提出要求時 AWS， AWS 強制執行程式碼會檢查主體是否通過驗證 (已登入) 和授權 (具有權限)。您可以透 AWS 過建立政策並將其附加到 IAM 身分或 AWS 資源來管理中的存取。策略是 JSON 文檔 AWS ，當附加到身份或資源時，定義其權限。如需有關政策類型及其使用的詳細資訊，請參閱 IAM 中的政策和許可。

如需有關其他身分驗證和授權程序的詳細資訊，請參閱 IAM 的運作方式。



在授權期間， AWS 強制執行程式碼會使用要求內容中的值來檢查相符的原則，並決定是否允許或拒絕要求。

AWS 檢查適用於請求內容的每個策略。如果單一原則拒絕要求，則會 AWS 拒絕整個要求並停止評估原則。此稱為明確拒絕。由於請求是預設拒絕，因此只有在適用的政策允許請求的每個部分時，IAM 才會授權該請求。用於單一帳戶中請求的評估邏輯遵循以下規則：

- 根據預設，所有的請求將以隱含方式拒絕。(此外，在預設情況下， AWS 帳戶根使用者 具有完整存取權限。)
- 若是以身分為基礎或以資源為基礎的政策，當中的明確允許會覆寫此預設值。
- 如果有許可界限、Organizations SCP 或工作階段政策，則其可能會以隱含拒絕覆寫該允許。
- 任何政策中的明確拒絕會覆寫任何允許。

在您的請求經過驗證和授權後， AWS 核准請求。如果您需要運用不同的帳戶來發出請求，則必須透過其他帳戶中的政策來允許您存取資源。此外，您用來發出請求的 IAM 實體必須具備可允許該請求的以身分為基礎政策。

存取管理資源

如需有關許可和建立政策的詳細資訊，請參閱以下資源：

AWS 安全部落格中的下列項目涵蓋撰寫 Amazon S3 儲存貯體和物件存取政策的常用方法。

- 撰寫 IAM 政策：如何授予存取 Amazon S3 儲存貯體的許可
- 撰寫 IAM 政策：授予存取 Amazon S3 儲存貯體中使用者特定資料夾的許可
- IAM 政策和儲存貯體政策和 ACL！天啊！(控制 S3 資源的存取)。
- RDS 資源層級許可的入門
- EC2 資源層級許可的說明

IAM 中的政策和許可

您可以 AWS 透過建立政策並將其附加到 IAM 身分 (使用者、使用者群組或角色) 或 AWS 資源來管理中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當 IAM 主體 (使用者或角色) 提出要求時，評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。 AWS 支援六種類型的原則：以身分識別為基礎的原則、以資源為基礎的原則、權限界限、Organizations SCP、ACL 和工作階段原則。

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，如果政策允許該GetUser動作，則具有該策略的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得使用者資訊。

當您建立 IAM 使用者時，您可以選擇允許其主控台存取或程式化存取。如果允許主控台存取，則 IAM 使用者可以利用登入認證登入主控台。若在允許程式化存取情況下，使用者可以使用存取金鑰來操作 CLI 或 API。

政策類型

以下政策類型會以最常使用到較不常使用的順序列出，可以在 AWS中使用。如需更多詳細資訊，請參閱以下各節中的每個政策類型。

- 身分型政策 – 將受管和內嵌政策連接到 IAM 身分 (使用者、使用者所屬的群組或角色)。身分型政策可為身分授予許可。
- 資源型政策 – 將內嵌政策連接到資源。資源型政策的最常見範例是 Amazon S3 儲存貯體政策和 IAM 角色信任政策。資源型政策會將許可授予政策中所指定的主體。主體可以位在與資源相同的帳戶，或位在其他的帳戶中。
- 許可界限 – 使用受管政策作為 IAM 實體 (使用者或角色) 的許可界限。該政策會定義身分型政策可為實體授予的最大許可，但並不會授予許可。許可界限不會定義資源型政策可以授予實體的許可上限。
- 組 Organ@@ izations SCP — 使用 AWS Organizations 服務控制策略 (SCP) 來定義組織或組織單位 (OU) 帳戶成員的最大權限。SCP 會限制身分型政策或資源型政策為帳戶中實體 (使用者或角色) 授予的許可，但不會授予許可。
- 存取控制清單 (ACL) – 使用 ACL，可以控制其他帳戶中的哪些主體可以存取其中已連接 ACL 的資源。ACL 類似資源型政策，雖然是不使用 JSON 政策文件結構的唯一政策類型。ACL 是跨帳戶許可的政策，負責為指定的主體授予許可。ACL 不可以為相同帳戶中的實體授予許可。
- 工作階段原則 — 當您使用 AWS CLI 或 AWS API 擔任角色或同盟使用者時，請傳遞進階工作階段原則。工作階段政策限制以角色或使用者身分型政策授予給該工作階段的許可。工作階段政策會限制已建工作階段的許可，但不會限制授予許可。如需詳細資訊，請參閱工作階段政策。

身分型政策

身分型政策是 JSON 許可政策文件，可控制身分 (使用者、使用者群組和角色) 會在何種條件下對哪些資源執行哪些動作。身分型政策可進一步分類：

- 受管理的策略 — 獨立以身分識別為基礎的策略，您可以將其附加到中的多個使用者、群組和角色。AWS 帳戶受管政策有兩種：
  - AWS 受管理的策略 — 由建立和管理的受管理策略 AWS。
  - 客戶管理政策：您在 AWS 帳戶中建立和管理的受管政策。與受管政策相比 AWS ，客戶管理的政策可以更精確地控制您的政策。

- 內嵌政策 – 您直接新增至單一使用者、群組或角色的政策。內嵌原則會維護原則與身分識別之間的嚴格 one-to-one 關係。當您刪除身分時，它們即會被刪除。

若要了解如何選擇受管政策和內嵌政策的相關資訊，請參閱 在受管政策與內嵌政策之間進行選擇。

資源型政策

資源型政策是連接到資源 (如 Amazon S3 儲存貯體) 的 JSON 政策文件。這些政策會授予指定的主體許可，允許在該資源上執行特定的動作，並且定義資源所適用的條件。資源型政策是內嵌政策。不存在受管的資源型政策。

若要啟用跨帳戶存取，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。當主參與者與資源分開時 AWS 帳戶，您也必須使用以識別為基礎的原則來授與主參與者對資源的存取權。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需授予跨服務存取權限的逐步指示，請參閱 IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶。

IAM 服務支援一種稱為角色信任政策，且已連接到 IAM 角色的資源型政策。IAM 角色既是一種身分，也是一項資源，可支援資源型政策。因此，您必須同時將信任政策和身分型政策連接到 IAM 角色。信任政策會定義哪些主體實體 (帳戶、使用者、角色和聯合身分使用者) 可擔任該角色。若要了解 IAM 角色與其他以資源為基礎之政策之間的差別，請參閱 IAM 中的跨帳戶資源存取。

若要查看哪些其他服務可以支援資源型政策，請參閱AWS 與 IAM 搭配使用的服務。若要進一步了解資源型政策的詳細資訊，請參閱 以身分為基礎和以資源為基礎的政策。若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？

IAM 許可界限

許可界限是一種進階功能，可供您設定身分型政策可以授予 IAM 實體的最大許可。當您為實體設定許可界限時，實體只能執行由身分型政策和其許可界限同時允許的動作。指定使用者或角色作為主體的資源型政策，不會受到許可界限的限制。所有這類政策中的明確拒絕都會覆寫該允許。如需有關許可界限的詳細資訊，請參閱 IAM 實體的許可界限。

服務控制政策 (SCP)

AWS Organizations 是一種用於分組和集中管理您企業所 AWS 帳戶 擁有的服務。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 是一種 JSON 政策，負責指定組織或組織單位 (OU) 的最大許可。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。所有這類政策中的明確拒絕都會覆寫該允許。

如需有關 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。

存取控制清單 (ACL)

存取控制清單 (ACL) 是一種服務政策，可讓您控制另一個帳戶中的哪些主體可以存取資源。ACL 不能用於控制相同帳戶中主體的存取。ACL 類似資源型政策，雖然是不使用 JSON 政策文件格式的唯一政策類型。Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF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。

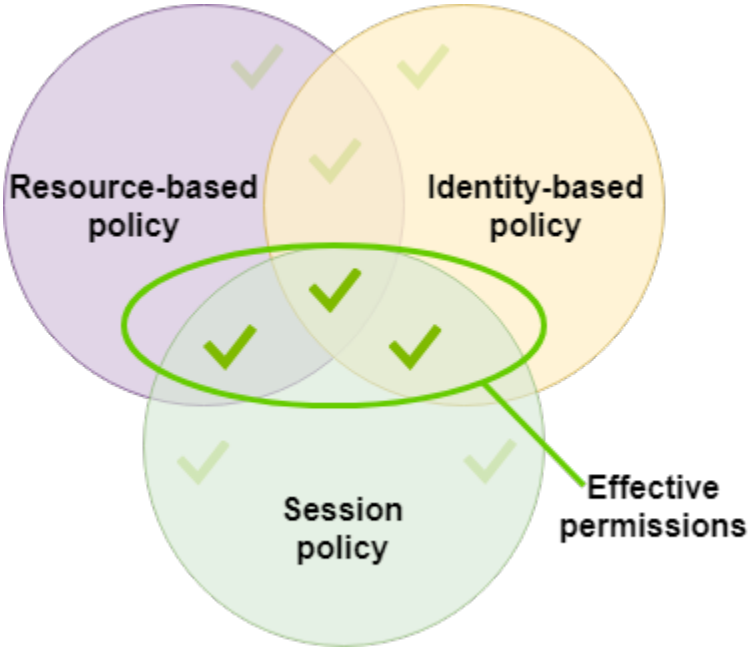
工作階段政策

工作階段政策是一種進階政策，且您會在以程式設計方式建立角色或聯合身分使用者的臨時工作階段時，以參數方式傳遞。工作階段的許可是用來建立工作階段及工作階段政策 IAM 實體 (使用者或角色) 之身分型政策的交集。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。

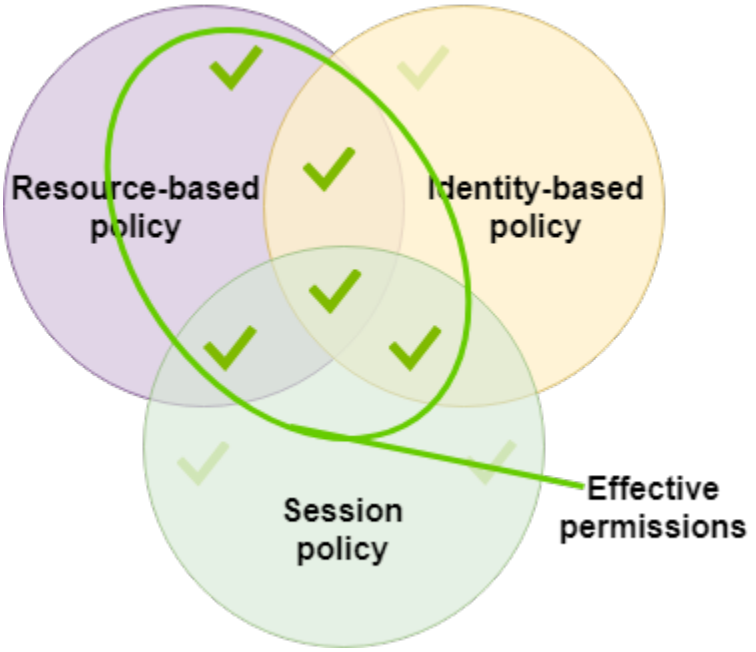
您可以建立角色工作階段，以及透過程式設計方式使用 `AssumeRole`、`AssumeRoleWithSAML` 或 `AssumeRoleWithWebIdentity` API 操作來傳遞工作階段政策。您可以使用 `Policy` 參數傳遞單一 JSON 內嵌工作階段政策文件。您可以使用 `PolicyArns` 參數，指定多達 10 個受管工作階段政策。如需有關建立角色工作階段的詳細資訊，請參閱 請求暫時性安全憑證。

建立聯合身分使用者工作階段時，您要使用 IAM 使用者的存取金鑰，以程式設計的方式來呼叫 `GetFederationToken` API 操作。您也必須通過工作階段政策。所產生工作階段的許可會是身分類型政策和工作階段政策的交集。如需有關建立聯合身分使用者工作階段的詳細資訊，請參閱 GetFederationToken—透過自訂身分經紀人聯合。

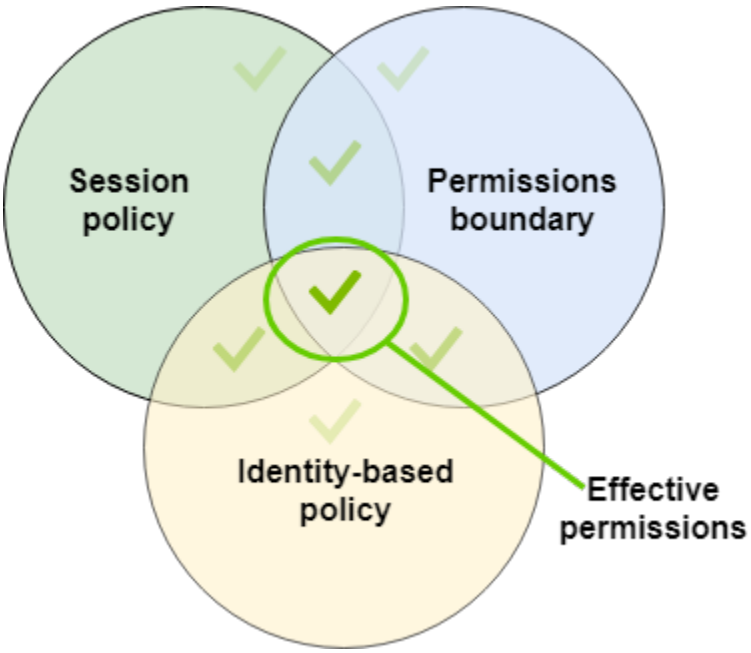
資源型政策可以指定使用者或角色 ARN 作為主體。在該情況下，在工作階段建立之前，依據資源型政策的許可會新增到該角色或使用者的身分型政策。工作階段政策限制由資源型政策、身分型政策所授予的總許可數。產生的工作階段的許可是工作階段政策與資源型政策的交集，加上工作階段政策與身分型政策的交集。



資源型政策可以指定工作階段 ARN 作為主體。在該情況下，會在建立工作階段後，從資源型政策新增許可。資源型政策許可不會受到工作階段政策的限制。產生的工作階段擁有資源型政策的所有許可，加上身分型政策與工作階段政策的交集。



許可界限可以為使用者或角色設定建立工作階段的許可上限。在該情況下，產生的工作階段的許可是工作階段政策、許可界限與身分型政策的交集。不過，許可界限不會限制由資源型政策所授予的許可，指定產生工作階段的 ARN。



政策和根使用者

受到 AWS 帳戶根使用者 某些策略類型的影響，但不會受到其他策略類型 您無法將身分型政策連接到根使用者，而且您無法為根使用者設定許可界限。不過，您可以在資源型政策或 ACL 中將根使用者指定為主體。根使用者仍是帳戶的成員。如果該帳戶是中組織的成員 AWS Organizations，則根使用者會受到該帳戶的任何 SCP 影響。

JSON 政策概觀

大多數原則會 AWS 以 JSON 文件的形式儲存在中。身分型政策以及用於設定許可界限的政策，都是您連接到使用者或角色的 JSON 政策文件。資源型政策是連接到資源的 JSON 政策文件。SCP 是 JSON 原則文件，其中包含您附加至 AWS Organizations 組織單位 (OU) 的受限語法。ACL 也會連接到資源，但您必須使用不同的語法。工作階段政策是您在擔任角色或聯合身分使用者工作階段時所需要提供的 JSON 政策。

您不需要了解 JSON 語法。您可以使用中的視覺化編輯器 AWS Management Console 來建立和編輯客戶管理的政策，而無需使用 JSON。不過，如果將內嵌政策用於群組或複雜政策，您仍然必須在 JSON 編輯器中利用主控台建立和編輯這些政策。如需有關使用視覺化編輯器的詳細資訊，請參閱 建立 IAM 政策 和 編輯 IAM 政策。

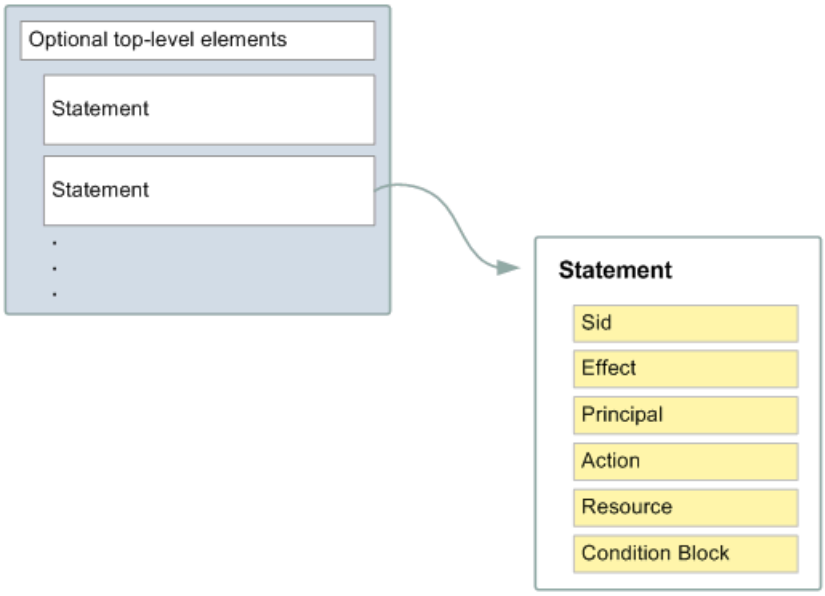
當您建立或編輯 JSON 政策時，IAM 可以執行政策驗證以協助您建立有效的政策。IAM 會識別 JSON 語法錯誤，而 IAM Access Analyzer 會提供額外的政策檢查及建議，協助您進一步改良政策。若要進一步了解政策驗證的資訊，請參閱 驗證 IAM 政策。若要進一步了解 IAM Access Analyzer 政策檢查和可動作的建議，請參閱 IAM Access Analyzer 政策驗證。

JSON 政策文件結構

如下圖所示，JSON 政策文件包含這些元素：

- 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。如果原則包含多個陳述式，則OR在評估陳述式時，會將邏輯 AWS 套用至所有陳述式。如果要求有多個原則套 AWS 用至要求，則OR在評估這些原則時，會將邏輯套用至所有這些原則。



陳述式中的資訊包含在一系列的元素內。

- Version – 指定您要使用的政策語言版本。建議您使用最新的 2012-10-17 版本。如需更多資訊，請參閱IAM JSON 政策元素：Version
- Statement – 使用此主要政策元素作為以下元素的容器。您可以在政策中包含多個陳述式。
- Sid (選用) – 包含選用的陳述式 ID 來區分您的陳述式。
- Effect – 使用 Allow 或 Deny 來表示政策是否允許或拒絕存取。
- Principal (僅在某些情況下需要) – 如果您建立以資源為基礎政策，則必須指示您想要允許或拒絕存取的帳戶、使用者、角色或聯合身分使用者。如果您要建立 IAM 許可政策來連接到使用者或角色，您不能包含此元素。主體意味著該使用者或角色。
- Action – 包含政策允許或拒絕的動作清單。

- Resource (僅在某些情況下需要) – 如果您建立 IAM 許可政策，則必須指示要套用動作的資源清單。如果您建立資源型政策，此元素是選用的。如果您未包含此元素，則此動作適用的資源是連接此政策的資源。
- Condition (選用) – 指定政策授予許可的條件。

若要了解這些內容及其他更進階的政策元素，請參閱IAM JSON 政策元素參考。

多項陳述式和多個政策

如果您想要定義一個以上的實體 (使用者或角色) 許可，您可以在單一政策使用多個陳述式。您也可以連接多個政策。如果您嘗試在單一陳述式定義多個許可，您的政策可能不會授予您預期的存取權。建議您透過資源類型將政策分類。

由於政策的大小限制，可能需要針對更複雜的許可採用多重政策。在不同的客戶管理政策中建立許可的功能群組，也是不錯的想法。例如，建立一個政策給 IAM 使用者管理、一個給自我管理，另一個給 S3 儲存貯體管理。無論多個陳述式和多個原則的組合為何，都會以相同的方式 AWS 評估您的原則。

例如，以下政策具有三個陳述式，每一個定義單一帳戶內不同組的許可。陳述式定義下列動作：

- 第一種陳述式具有 `Sid` 的 `FirstStatement` (陳述式 ID)，可讓具有連接政策的使用者變更自己的密碼。此陳述式的 `Resource` 元素是 `*` (其表示「所有資源」)。但事實上 `ChangePassword`API 操作 (或同等 `change-password` CLI 命令) 只對提出請求的使用者的密碼有影響。
- 第二個陳述式可讓使用者在其 AWS 帳戶列出所有的 Amazon S3 儲存貯體。此陳述式的 `Resource` 元素是 `"*"` (其表示「所有資源」)。但是，因為政策不授予其他帳戶的資源存取權，使用者只能列出自己的 AWS 帳戶中的儲存貯體。
- 第三個陳述式可讓使用者列出並擷取儲存貯體中的任何物件，名為 `confidential-data`，但前提是使用者必須是以 多重要素驗證 (MFA) 驗證。政策中的 `Condition` 元素強制執行 MFA 身分驗證。

  當政策陳述式包含 `Condition` 元素，陳述式僅於 `Condition` 元素評估為 true 是生效。在這種情況下，當使用者經過 MFA 身分驗證，`Condition` 評估為 true。如果使用者未經過 MFA 身分驗證，這項 `Condition` 評估設為 false。在這種情況下，此政策的第三個陳述式將不適用，而且使用者無法存取 `confidential-data` 儲存貯體。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "ThirdStatement",
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::confidential-data",
        "arn:aws:s3:::confidential-data/*"
      ],
      "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
    }
  ]
}
```

JSON 政策語法的範例

以下身分型政策允許暗示的主體列出單一 Amazon S3 儲存貯體，其名稱是 `example_bucket`：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

以下資源型政策可以連接到 Amazon S3 儲存貯體。此政策允許特定 AWS 帳戶 的成員在名為的儲存貯體中執行任何 Amazon S3 動作`mybucket`。它允許可在儲存貯體或其中的物件上執行的任何動作。(由於政策僅對帳戶授予信任，帳戶中的個別使用者仍必須被授予特定 Amazon S3 動作的許可)。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "1",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::account-id:root"]},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::mybucket",
      "arn:aws:s3:::mybucket/*"
    ]
  }]
}
```

若要檢視常見案例的範例政策，請參閱以身分為基礎的 IAM 政策範例。

授予最低權限

當您建立 IAM 政策時，請遵循授予最低權限的標準安全建議，或者只授予執行任務所需的許可。決定使用者和角色需要執行哪些任務，然後為他們制定政策，讓使用者只執行這些任務。

以最小一組許可開始，然後依需要授予額外的許可。這比一開始使用太寬鬆的許可，稍後再嘗試將他們限縮更為安全。

作為最低權限的替代方案，您可以使用 AWS 受管政策或具有萬用字元 * 許可的政策，以開始使用政策。若授與主體的許可超出執行任務所需的許可，需考量其相關的安全性風險。監控這些主體，以了解他們正在使用的許可。然後寫入最低權限政策。

IAM 提供數個選項，協助您精簡您授予的許可。

- 了解存取層級群組 – 您可以使用存取層級的方法進行分組，以了解政策授予的存取層級。政策動作分為 `List`、`Read`、`Write`、`Permissions management` 或 `Tagging`。例如，您可以從 `List` 和 `Read` 存取層級選擇動作以授予唯讀存取權給您的使用者。若要了解如何使用政策摘要，以理解存取層級許可，請參閱瞭解原則摘要中的存取層級。
- 驗證您的政策 – 您可以在建立和編輯 JSON 政策時，使用 IAM Access Analyzer 執行政策驗證。我們建議您檢閱並驗證所有現有政策。IAM Access Analyzer 提供超過 100 項的政策檢查，以驗證您的政策。當您政策中的聲明允許我們認為過度寬鬆的存取權時，即會產生安全性警告。在授予最低權限時，您可以使用透過安全性警告提供的可操作建議。若要進一步了解有關 IAM Access Analyzer 所提供的政策檢查，請參閱 IAM Access Analyzer 政策驗證。

- 根據存取活動產生政策 – 為了協助您精簡所授予的許可，您可以根據 IAM 實體 (使用者或角色) 的存取活動產生 IAM 政策。IAM Access Analyzer 會檢閱您的 AWS CloudTrail 記錄檔，並產生政策範本，其中包含實體在指定時間範圍內使用的許可。您可以使用範本建立具有精細許可的受管政策，然後將其連接至 IAM 實體。如此一來，您只會針對特定使用案例授與使用者或角色與 AWS 資源互動所需的權限。如需進一步了解，請參閱 根據存取活動產生政策。
- 使用上次存取的資訊 – 另一個有助於實現最低權限的功能是最近存取的資訊。在 IAM 主控台詳細資訊頁面的 Access Advisor (存取 Advisor) 索引標籤上檢視 IAM 使用者、群組、角色或政策的此資料。上次存取的資訊也包括最近存取某些服務之動作的相關資訊，例如 Amazon EC2、IAM、Lambda 和 Amazon S3。如果您使用 AWS Organizations 管理帳戶登入資料登入，則可以在 IAM 主控台的AWS Organizations區段中檢視上次存取的服務資訊。您也可以使用 AWS CLI 或 AWS API 擷取 IAM 或 Organizations 中實體或政策上次存取資訊的報告。您可以使用此資訊來找出不必要的許可，以便您能夠微調 IAM 或 Organizations 政策以更完善地遵循最低權限的原則。如需詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。
- 檢閱中的帳戶事件 AWS CloudTrail— 若要進一步降低權限，您可以在 AWS CloudTrail 事件歷史記錄中檢視帳戶的事件。 CloudTrail 事件記錄檔包含詳細的事件資訊，您可以使用這些資訊來降低策略的權限。此日誌只包含 IAM 實體所需的動作和資源。如需詳細資訊，請參閱《使用指南》中的〈在 CloudTrail主控台中檢視 CloudTrail 事件AWS CloudTrail〉。

如需詳細資訊，請參閱下列適用於個別服務的政策主題，其中提供如何為服務特定的資源撰寫政策的範例。

- 《Amazon DynamoDB 開發人員指南》中的Amazon DynamoDB 的身分驗證與存取控制
- 《Amazon Simple Storage Service 使用者指南》中的使用儲存貯體政策和使用者政策
- 《Amazon Simple Storage Service 使用者指南》中的存取控制清單 (ACL) 概觀

受管政策與內嵌政策

當您在 IAM 中為身分設定許可時，必須決定要使用 AWS 受管政策、客戶管理政策還是內嵌政策。以下幾個主題提供了有關每個類型的身分型政策以及使用時機的更多資訊。

主題

- AWS 受管理政策
- 客戶受管政策
- 內嵌政策

- 在受管政策與內嵌政策之間進行選擇
- 受管政策入門
- 將內嵌政策轉換為受管政策
- 已取代的 AWS 受管理

AWS 受管理政策

「AWS 受管政策」為獨立的政策，由 AWS建立並管理。獨立政策表示政策有自己的 Amazon Resource Name (ARN)，其中包含政策名稱。例如，`arn:aws:iam::aws:policy/IAMReadOnlyAccess`是 AWS 受管理的策略。如需有關 ARN 的詳細資訊，請參閱 IAM ARN。如需的 AWS 受管理原則清單 AWS 服務，請參閱AWS 受管理的原則。

AWS 受管理的原則可讓您方便地將適當的權限指派給使用者、群組和角色。這比您自己撰寫政策更快，並且包含常見使用案例的許可。

您無法變更受 AWS 管理策略中定義的權限。 AWS 偶爾會更新受 AWS 管理策略中定義的權限。AWS 執行此操作時，更新會影響附加原則的所有主參與者實體 (使用者、群組和角色)。 AWS 當新服務啟動或新的 API 呼叫可供現有 AWS 服務使用時，最有可能更新 AWS 受管理政策。例如，名為的 AWS 受管理策略ReadOnlyAccess提供對所有 AWS 服務和資源的唯讀存取權。 AWS 啟動新服務時，會 AWS 更新ReadOnlyAccess原則以新增新服務的唯讀權限。更新的許可會套用於政策連接到的所有主體實體。

完整存取 AWS 受管理的原則會授與服務的完整存取權，以定義服務管理員的權限。

- AmazonDynamo資料庫 FullAccess
- IAM FullAccess

進階使用者 AWS 管理的原則可提供 AWS 服務和資源的完整存取權，但不允許管理使用者和群組。

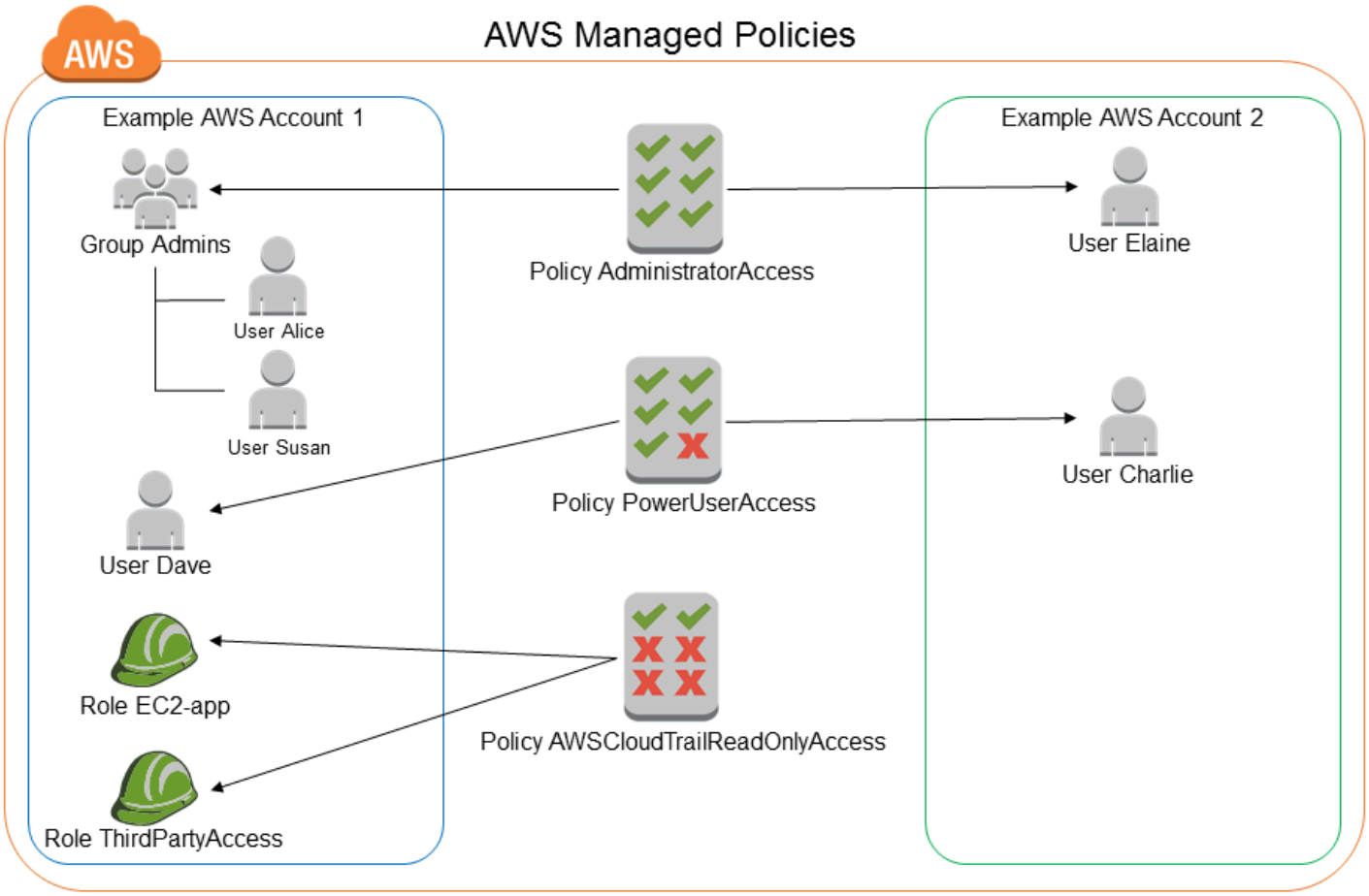
- AWSCodeCommitPowerUser
- AWSKeyManagementServicePowerUser

部分存取 AWS 受管理的原則可提供特定層級的 AWS 服務存取權，而不允許權限管理存取層級權限。

- AmazonMobileAnalyticsWriteOnlyAccess
- 亚马逊 ReadOnlyAccess

AWS 受管理策略的一個特別有用的類別是那些專為工作職能而設計的。這些政策與 IT 業界的常用工作職能緊密貼合，並有助於為這些工作職能授予許可。使用工作功能政策的一個主要優點是，在引入新服務和 API 操 AWS 作時，它們會被維護和更新。例如，AdministratorAccess工作功能可為中的每個服務和資源提供完整存取權和權限委派 AWS。我們建議僅將此政策用於帳戶管理員。對於需要完全存取每項服務的進階使用者，但 IAM 和 Organizations 的有限存取權限除外，請使用PowerUserAccess工作功能。如需有關工作職能政策的清單和說明，請參閱AWS 受管理的工作職能政策。

下圖說明 AWS 受管理的策略。此圖表顯示三個 AWS 受管理的政策：AdministratorAccessPowerUserAccess、和AWS CloudTrailReadOnlyAccess。請注意，單一 AWS 受管理的原則可以附加至不同的主參與者實體 AWS 帳戶，以及單一的不同主參與者實體 AWS 帳戶。



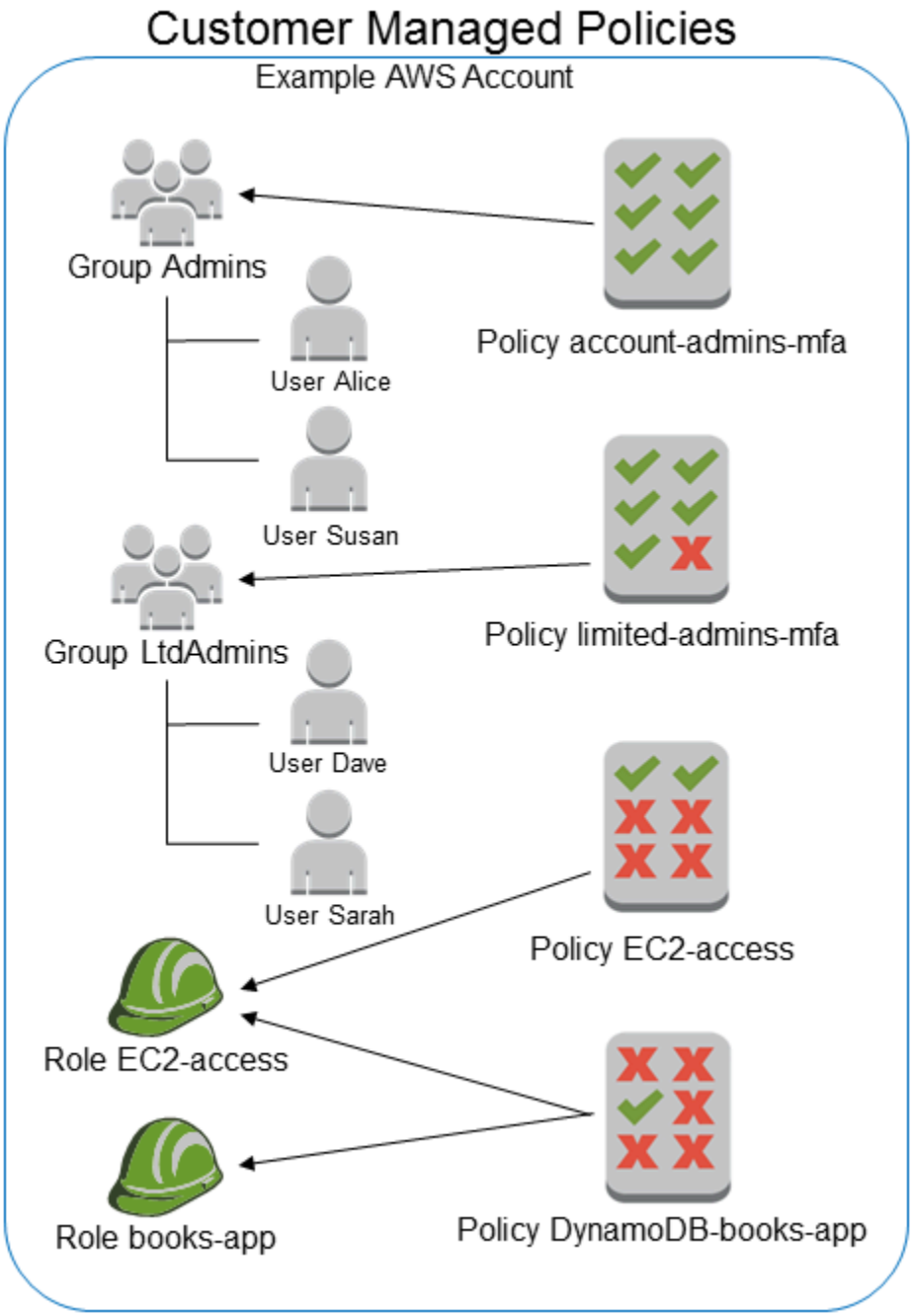
客戶受管政策

您可以自 AWS 帳戶 行建立可附加至主參與者實體 (使用者、群組和角色) 的獨立原則。您可以針對特定使用案例建立這些客戶管理政策，並且可以隨時進行變更和更新。就像 AWS 受管理的原則一樣，當您將原則附加至主參與者實體時，您會將原則中定義的權限授與實體。更新政策中的許可時，變更會套用於政策連接到的所有主體實體。

建立客戶管理政策的理想方式是從複製一個現有 AWS 受管政策開始。這樣從一開始您就可以確信政策是正確的，只需根據您的環境進行自訂即可。

下圖說明客戶管理政策。每個政策都是 IAM 中的一個實體，有自己的 Amazon Resource Name (ARN)，其中包含政策名稱。請注意，同一政策可以連接到多個主體實體，例如，同一個 DynamoDB-books-app 政策可連接到兩個不同的 IAM 角色。

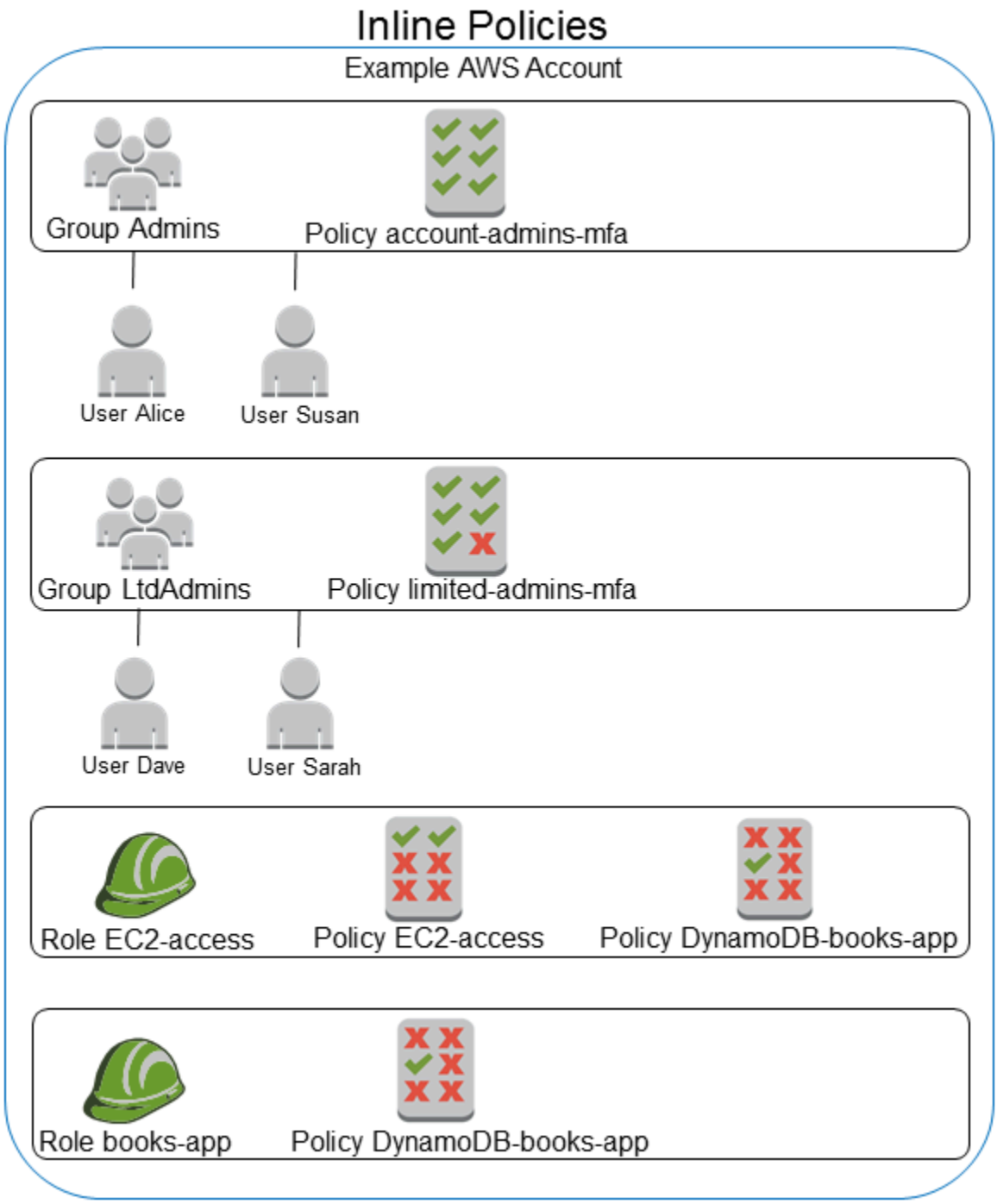
如需更多資訊，請參閱建立 IAM 政策



內嵌政策

內嵌政策是為單個 IAM 身分 (使用者、群組或角色) 建立的政策。內嵌原則會維護原則與身分識別之間的嚴格 one-to-one 關係。當您刪除身分時，它們即會被刪除。在建立身分時或在建立之後，您可以建立政策並將其嵌入身分。如果政策可套用至多個實體，最好使用受管政策。

下圖說明內嵌政策。每個政策都是使用者、組或角色的固有部分。請注意，兩個角色包含同一政策 (DynamoDB-books-app 政策)，但是它們不共用單一政策。每個角色都有自己的政策複本。



在受管政策與內嵌政策之間進行選擇

在決定是使用受管政策還是內嵌政策時，考慮您的使用案例。在大多數情況下，我們建議使用受管政策而非內嵌政策。

> **Note**
>
> 您可以同時使用受管政策和內嵌政策，來定義主體實體的一般許可和唯一許可。

受管政策具備以下功能：

可重複使用性

單一受管政策可以連接到多個主體實體 (使用者、群組和角色)。您可以建立為您定義有用權限的原則庫 AWS 帳戶，然後視需要將這些原則附加至主參與者實體。

集中變更管理

更改受管政策時，更改會套用於政策連接到的所有主體實體。例如，如果您想要新增 AWS API 的權限，您可以更新客戶管理的政策或建立受 AWS 管理政策的關聯以新增權限。如果您使用受 AWS 管原則，請 AWS 更新原則。更新受管理的原則時，變更會套用至附加受管理原則的所有主參與者實體。相反地，若要變更內嵌原則，您必須個別編輯包含內嵌原則的每個識別。例如，如果一個群組和一個角色都包含同一內嵌政策，則您必須分別編輯這兩個主體實體以變更該政策。

版本控制和還原

在更改客戶管理政策時，更改的政策不會覆蓋現有的政策。IAM 反而會建立新版本的受管政策。IAM 最多可以儲存五個版本的客戶管理政策。如果需要，可以使用政策版本將政策還原為較早版本。

> **Note**
>
> 政策版本與 `Version` 政策元素不同。`Version` 政策元素是在政策內使用，並定義政策語言的版本。若要進一步了解政策版本，請參閱 the section called “版本控制 IAM 政策”。若要進一步了解 `Version` 政策元素，請參閱 IAM JSON 政策元素：Version。

委派許可管理

您可以允許您中的使用者附 AWS 帳戶 加和卸離原則，同時保持對這些策略中定義的權限的控制權限。因此，您可以將一些使用者指定為完全管理員，也就是建立、更新和刪除政策的管理員。隨後可以將其使用者指定為受限管理員。受限管理員可以將政策 (限於您允許連接的政策) 連接到其他主體實體。

如需有關委派許可管理的詳細資訊，請參閱 控制對政策的存取。

較大政策字元限制

受管政策的字元大小上限大於內嵌政策的字元限制。如果達到內嵌政策的字元大小限制，您可以建立更多 IAM 群組，並將受管政策附加到群組。

如需有關配額和限制的詳細資訊，請參閱 IAM 和 AWS STS 配額。

AWS 受管理策略的自動更新

AWS 維護 AWS 受管策略並在必要時更新它們，例如，為新 AWS 服務添加權限，而無需進行更改。更新會自動套用至您已附加 AWS 受管理原則的主參與者實體。

使用內嵌政策

如果您想要在原則與套用原則的識別之間維持嚴格的 one-to-one 關係，內嵌原則非常有用。例如，如果您要確保政策中的許可不會無意中分配給預期身分之外的身分。使用內嵌政策時，政策中的許可不能意外分配給錯誤的身分。此外，當您使用刪除該識別時，內嵌在識別中的原則也會一併刪除，因為它們是主參與者實體的一部分。 AWS Management Console

受管政策入門

我們建議使用授予最低權限的政策，或僅授予執行任務所需的許可。授予最低權限的最安全方式是撰寫僅具有團隊所需許可的客戶管理政策。您必須建立程序，以允許您的團隊在必要時要求更多許可。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。

若要開始將許可新增至您的 IAM 身分 (使用者、使用者群組和角色)，您可以使用AWS 受管理政策。AWS 受管理的政策不會授與最低權限。若授予主體的許可超出執行任務所需的許可，您必須考量其相關的安全性風險。

您可以將 AWS 受管政策 (包括工作職能) 附加至任何 IAM 身分。如需詳細資訊，請參閱 新增和移除 IAM 身分許可。

若要切換到最低權限權限，您可以執行 AWS Identity and Access Management 存取分析器來監視具有 AWS 受管理原則的主體。了解他們正在使用哪些許可後，您可以撰寫或產生僅具有團隊所需許可的客戶管理政策。這不太安全，但是當您了解團隊的使用方式時，會提供更大的靈活性 AWS。如需詳細資訊，請參閱 產生 IAM Access Analyzer 政策。

AWS 受管理的政策旨在為許多常見使用案例提供權限。如需針對特定工作職能所設計之 AWS 受管理原則的詳細資訊，請參閱AWS 受管理的工作職能政策。

如需 AWS 受管策略的清單，請參閱AWS 受管理的策略參考指南。

將內嵌政策轉換為受管政策

如果您在您的帳戶中擁有內嵌政策，您可以將它們轉換為受管政策。若要執行此操作，請將該政策複製到新的受管政策。接下來，將新政策連接到具有內嵌政策的身分，然後刪除內嵌政策。

將內嵌政策轉換為受管政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在服務導覽窗格中，選擇 User groups (使用者群組)、Users (使用者) 或者 Roles (角色)。
3. 在清單中，選擇包含要刪除的政策的使用者群組、使用者、或角色的名稱。
4. 選擇 Permissions (許可) 標籤。
5. 針對使用者群組，選取要移除之內嵌政策的名稱。針對使用者和角色，選擇顯示 更多項目，並在必要時展開您想要移除之內嵌政策。
6. 選擇複製，以複製該政策的 JSON 政策文件。
7. 在導覽窗格中，選擇政策。
8. 選擇建立政策，然後選擇 JSON 選項。
9. 將現有的文字取代為您的 JSON 政策文字，然後選擇下一步。
10. 輸入您的政策的名稱和選用描述，然後選擇建立政策。
11. 在導覽窗格中選擇 User groups (使用者群組)、Users (使用者) 或 Roles (角色)，並再次選擇擁有您想要移除之政策的使用者群組、使用者或角色的名稱。
12. 選擇許可索引標籤，然後選擇新增許可。
13. 針對使用者群組，選取新政策名稱旁的核取方塊，然後依序選擇 Add permissions (新增許可)、Attach policy (連接政策)。針對使用者或角色，選擇 Add permissions (新增許可)。在下一個頁面上，選擇直接連接現有政策，選取新政策名稱旁的核取方塊，選擇下一步，然後選擇新增許可。

    系統會將您傳回到使用者群組、使用者或角色的 Summary (摘要) 頁面。
14. 選取您要移除之內嵌政策旁的核取方塊，然後選擇移除。

已取代的 AWS 受管理

為了簡化許可的指派，請 AWS 提供受管政策 — 預先定義的政策，可連接到 IAM 使用者、群組和角色。

有時候 AWS 需要將新權限新增至現有原則，例如引入新服務時。將新的許可新增至現有政策，並不會中斷或移除任何功能或能力。

不過，如果所需的變更可 AWS 能會影響客戶套用至現有政策，則可能會選擇建立新政策。例如，從現有政策移除許可可能破壞任何 IAM 實體或應用程式所倚靠的許可，而可能中斷關鍵的操作。

因此，當需要進行此類變更時， AWS 會建立包含所需變更的全新政策，並提供給客戶使用。然後，舊政策會標示為已作廢。已取代的受管政策會在 IAM 主控台的 Policies (政策) 清單中的政策旁顯示警告圖示。

已作廢的政策具有以下特點：

- 它會繼續適用於所有目前連接的使用者、群組和角色。沒有任何破壞。
- 它無法連接到任何新的使用者、群組或角色。如果您將其從目前的實體分開，您無法重新連接。
- 在您將其從所有目前實體分開之後，它不會再顯示，而且無法再以任何方式使用。

如果任何使用者、群組或角色需要政策，您必須改為連接新的政策。當您接收政策已作廢的通知時，我們建議您立即計劃將所有使用者、群組和角色連接到替換的政策，並將其從作廢的政策分開。繼續使用作廢的政策可以帶有風險，唯有切換到替換的政策才會緩解。

使用資料周長建立權限護欄

資料周邊防護的目的是做為永遠在線的界限，以協助保護您在廣泛的帳戶和資源中的 AWS 資料。資料周長遵循 IAM 安全最佳實務，跨多個帳戶建立許可保護。這些全組織的權限護欄不會取代您現有的精細存取控制。相反，它們作為粗略的訪問控制，通過確保用戶，角色和資源遵守一組定義的安全標準，有助於改善安全策略。

資料周邊是您 AWS 環境中的一組權限護欄，有助於確保只有您信任的身分正在存取來自預期網路的受信任資源。

- 受信任的身分：您 AWS 帳戶和 AWS 服務中代表您行事的主體 (IAM 角色或使用者)。
- 受信任的資源：您的 AWS 帳戶或代表您行事的 AWS 服務所擁有的資源。
- 預期網路：您的內部部署資料中心和虛擬私有雲 (VPC)，或代表您行動的 AWS 服務網路。

> **Note**
>
> 在某些情況下，您可能需要擴大資料範圍，以包括您信任的業務合作夥伴的存取權限。當您建立受信任的身分識別、受信任資源和預期網路的定義時，您應該考慮所有預期的資料存取模式，以及貴公司的使用情況 AWS 服務。

數據周邊控制應被視為信息安全和風險管理程序中的任何其他安全控制。這表示您應該執行威脅分析以識別雲端環境中的潛在風險，然後根據您自己的風險接受準則，選取並實作適當的資料周邊控制。為了更好地瞭解資料周邊實作的反覆風險型方法，您需要瞭解資料周邊控制可解決哪些安全風險和威脅媒介，以及您的安全優先順序。

資料周長控制

資料周邊粗粒度控制項可透過實作不同組合政策類型和條件金鑰，協助您在三個資料周圍達成六個截然不同的安全性目標。

周長	控制目標	使用	套用於	全域條件內容索引鍵
Identity	只有信任的身分可以存取我的資源	以資源為基礎政策	資源	AWS：PrincipalOrg身份證 AWS：PrincipalOrgPaths AWS：PrincipalAccount AWS：PrincipalIsAwsService
	我的網路只允許受信任的身分	VPC 端點政策	網路	
資源	您的身分只能存取受信任的資源	SCP	身分	AWS：ResourceOrg身份證 AWS：ResourceOrgPaths
	只有受信任的資源才能從您的網路存取	VPC 端點政策	網路	

周長	控制目標	使用	套用於	全域條件內容索引鍵
				AWS：ResourceAccount
網路	您的身分只能從預期的網路存取資源	SCP	身分	AWS：SourceIp AWS：SourceVpc
	您的資源只能從預期的網絡訪問	以資源為基礎政策	資源	AWS：SourceVpce AWS: 透過 AWSService AWS：PrincipalIsAwsService

您可以將資料周圍視為在資料周圍建立堅實的邊界，以防止意外的存取模式。雖然資料周長可以防止廣泛的意外存取，但您仍然需要做出精細的存取控制決策。建立資料周邊並不會減少透過使用 IAM Access Analyzer 等工具來持續微調許可的需求，作為您獲得最小特權的旅程的一部分。

身份周長

身分周邊是一組粗略的預防性存取控制，有助於確保只有受信任的身分可以存取您的資源，而且只允許來自您的網路的受信任身分識別。信任的身分識別會在您的 AWS 帳戶和代表您行事的 AWS 服務中包括主體 (角色或使用者)。除非授與明確的例外狀況，否則所有其他身分識別都會被視為不受信任，且會被識別周邊防止。

下列全域條件金鑰有助於強制執行身分周邊控制。在以資源為基礎的政策中使用這些金鑰來限制對資源的存取，或在 VPC 端點政策中使用這些金鑰來限制對網路的存取。

- AWS：PrincipalOrg身份證— 您可以使用此條件金鑰來確保提出請求的 IAM 主體屬於中 AWS Organizations的指定組織。
- AWS：PrincipalOrg路徑— 您可以使用此條件金鑰來確保 IAM 使用者、IAM 角色、聯合身分AWS 帳戶根使用者 使用者或提出請求的 A 屬於中 AWS Organizations指定的組織單位 (OU)。

- AWS：PrincipalAccount— 您可以使用此條件金鑰來確保資源只能由您在策略中指定的主體帳戶存取。
- AWS：PrincipalIsAWSService和 AWS：SourceOrg身份證 (交替AWS：SourceOrg路徑和AWS：SourceAccount) — 您可以使用這些條件金鑰來確保當AWS 服務 主參與者存取您的資源時，他們只代表中指定組織、組織單位或帳號中的資源來執行此動作。 AWS Organizations

如需詳細資訊，請參閱建立資料周邊 AWS：僅允許受信任的身分存取公司資料。

資源周邊

資源周邊是一組粗略的預防性存取控制，有助於確保您的身分識別只能存取受信任的資源，而且只能從您的網路存取受信任的資源。受信任的資源包括您的 AWS 帳戶或代表您行事的 AWS 服務所擁有的資源。

下列全域條件索引鍵有助於強制執行資源周邊控制。使用服務控制原則 (SCP) 中的這些金鑰來限制您的身分識別可存取哪些資源，或在 VPC 端點原則中限制哪些資源可從您的網路存取。

- AWS：ResourceOrg身份證— 您可以使用此條件索引鍵來確保要存取的資源屬於中的指定組織 AWS Organizations。
- AWS：ResourceOrg路徑— 您可以使用此條件索引鍵來確保要存取的資源屬於中指定的組織單位 (OU) AWS Organizations。
- AWS：ResourceAccount— 您可以使用此條件金鑰來確保正在存取的資源屬於中的指定帳號 AWS Organizations。

在某些情況下，您可能需要允許存取 AWS 擁有的資源、不屬於您組織的資源，以及由您的主體或代表您行事的 AWS 服務存取的資源。如需有關這些案例的詳細資訊，請參閱建立資料周邊 AWS：僅允許來自組織的受信任資源。

網路周邊

網路周邊是一組粗略的預防性存取控制，有助於確保您的身分識別只能從預期的網路存取資源，而您的資源只能從預期的網路存取。預期網路包括您的內部部署資料中心和虛擬私有雲 (VPC)，以及代表您行事的 AWS 服務網路。

下列全域條件金鑰有助於強制執行網路周邊控制。在服務控制原則 (SCP) 中使用這些金鑰來限制您的身分可通訊的網路，或在以資源為基礎的政策中使用這些金鑰，以限制對預期網路的資源存取。

- AWS：SourceIp— 您可以使用此條件金鑰來確保要求者的 IP 位址在指定的 IP 範圍內。

- AWS：SourceVpc— 您可以使用此條件金鑰來確保要求所經歷的 VPC 端點屬於指定的 VPC。
- AWS：SourceVpce— 您可以使用此條件金鑰來確保要求通過指定的 VPC 端點。
- AWS: 透過 AWSService— 您可以使用此條件金鑰來確保 AWS 服務 可以使用 轉送存取工作階段 (FAS) 代表您的主體提出要求。
- AWS：PrincipalIsAWSService— 您可以使用此條件金鑰來確保 AWS 服務 可以使用存取您的資源AWS 服務主體。

在其他情況下，您需要允許從網路外部存取您的資源的存取權。 AWS 服務 有關詳情，請參閱建立資料周邊 AWS：僅允許從預期網路存取公司資料。

進一步了解資料周長的資源

下列資源可協助您進一步了解資 AWS料周長。

- 上的資料周長 AWS — 瞭解資料周長及其優點和使用案例。
- 白皮書：建立資料周邊 AWS — 本白皮 paper 概述在 AWS中建立身分識別、資源和網路周邊的最佳實務和可用服務。
- 網路研討會：在中建立資料周邊 AWS— 瞭解在何處以及如何根據不同的風險案例實作資料周邊控制。
- 部落格文章系列：建立資料周邊 AWS — 這些部落格文章涵蓋大規模建立資料周邊的規範指引，包括重要的安全性和實作考量。
- 資料周長原則範例 — 此 GitHub 儲存庫包含範例原則，涵蓋一些常見模式，可協助您實作資料周邊 AWS。
- 資料周邊協助程式 — 此工具可透過分析AWS CloudTrail記錄中的存取活動，協助您設計和預測資料周邊控制的影響。

IAM 實體的許可界限

AWS 支援 IAM 實體 (使用者或角色) 的許可界限。許可界限是一種進階功能，可供您使用受管政策來設定以身分為基礎的政策可以授予 IAM 實體的最大許可。實體的許可界限可讓其只執行由身分類型政策和其許可界限同時允許的動作。

如需有關政策類型的詳細資訊，請參閱 政策類型。

> **Important**
>
> 對於已附加許可界限政策的 IAM 使用者或角色，請勿使用所含 NotPrincipal 政策元素具有 Deny 效果的資源型政策陳述式。無論在 NotPrincipal 元素中指定何值，具有 Deny 效果的 NotPrincipal 元素將始終拒絕任何已附加許可界限政策的 IAM 主體。這會造成部分本可存取資源的 IAM 使用者或角色失去存取權。我們建議您變更資源型政策陳述式，將條件運算子 ArnNotEquals 與 aws:PrincipalArn 內容索引鍵搭配使用來限制存取，而不是使用 NotPrincipal 元素。如需有關 NotPrincipal 元素的資訊，請參閱 AWS 政策元素：NotPrincipal。

您可以使用 AWS 受管政策或客戶受管政策來設定 IAM 實體 (使用者或角色) 的界限。該政策限制使用者或角色的許可上限。

例如，假設ShirleyRodriguez應該允許名為的 IAM 使用者只能管理 Amazon S3 CloudWatch、Amazon 和 Amazon EC2。為了強行實施這個規則，您可以使用以下政策設定 ShirleyRodriguez 使用者的許可界限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

當您使用政策來設定使用者的許可界限時，它會限制使用者的許可，但不自行提供許可。在此範例中，政策將 Amazon S3 和 Amazon EC2 中所有操作的最大許可設定ShirleyRodriguez為。CloudWatchShirley 永遠無法在其他任何服務中執行操作，包括 IAM，即使許可政策允許她這麼做。例如，您可以將以下政策加入到 ShirleyRodriguez 使用者：

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

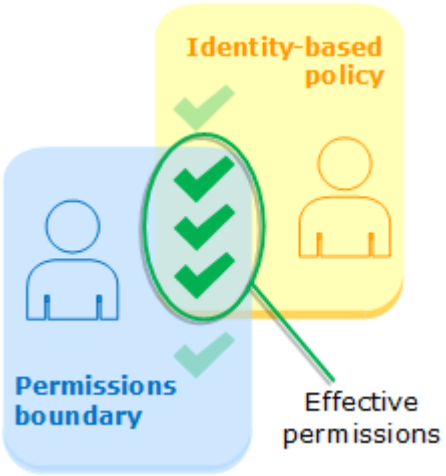
這項政策允許在 IAM 中建立使用者。如果您將此許可政策連接到 ShirleyRodriguez 使用者，當 Shirley 嘗試建立使用者時，操作會失敗。失敗的原因是許可界限不允許 iam:CreateUser 操作。因為這兩項政策，Shirley 沒有許可在 AWS中執行任何操作。您必須新增不同的許可政策，才能允許其他服務 (例如 Amazon S3) 中的動作。或者，您可以更新許可界限，允許她在 IAM 中建立使用者。

評估含界限的有效許可

IAM 實體 (使用者或角色) 的許可界限會設定該實體可以擁有的最大許可。這可能改變該使用者或角色的有效許可。實體的有效許可，是指會影響使用者或角色的所有政策所授予的許可。在帳戶範圍內，以身分為基礎的政策、以資源為基礎的政策、許可界限、Organizations SCP 或工作階段政策，都可能會影響到實體的許可。如需有關不同政策類型的詳細資訊，請參閱 IAM 中的政策和許可。

如果其中任何一種政策會明確拒絕存取操作，則請求將會遭拒。經由多種許可類型授予的許可將更為複雜。如需有關如何 AWS 評估策略的詳細資訊，請參閱政策評估邏輯。

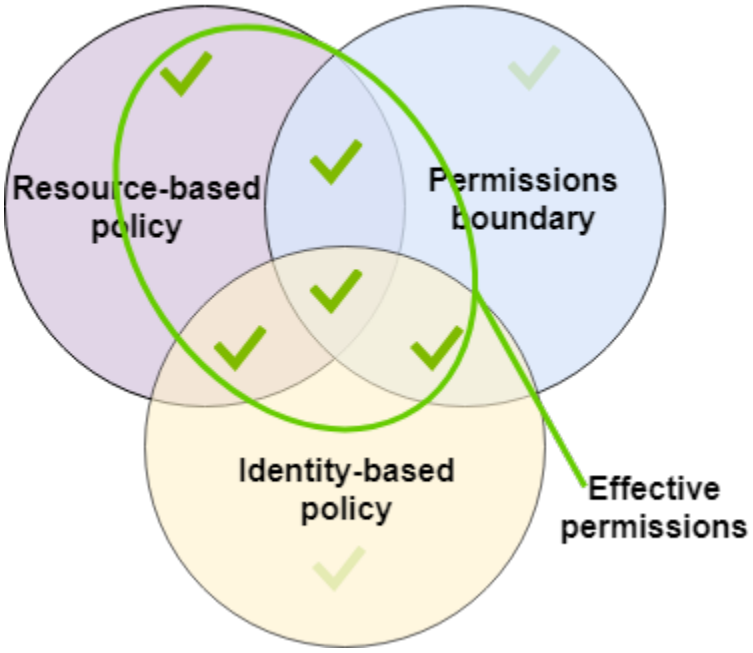
包含界限的以身分為基礎政策 – 以身分為基礎的政策可分為會連接至使用者、使用者群組或角色的內嵌或受管政策。以身分為基礎的政策可為實體授予許可，而許可界限會限制這些許可。有效的許可是兩種政策類型的交集。在這些政策的明確拒絕會覆寫允許。



以資源為基礎的政策 – 以資源為基礎的政策會控制指定主體如何存取資源，即政策連接至其中的資源。

適用於 IAM 使用者的資源型政策

在同一帳戶內，將許可授予 IAM 使用者 ARN (非聯合身分使用者工作階段) 的資源型政策不受身分識別型政策或許可界限中隱含拒絕的限制。



適用於 IAM 角色的資源型政策

IAM 角色 – 將許可授予 IAM 角色 ARN 的資源型政策受限於許可界限或工作階段政策中隱含拒絕的限制。

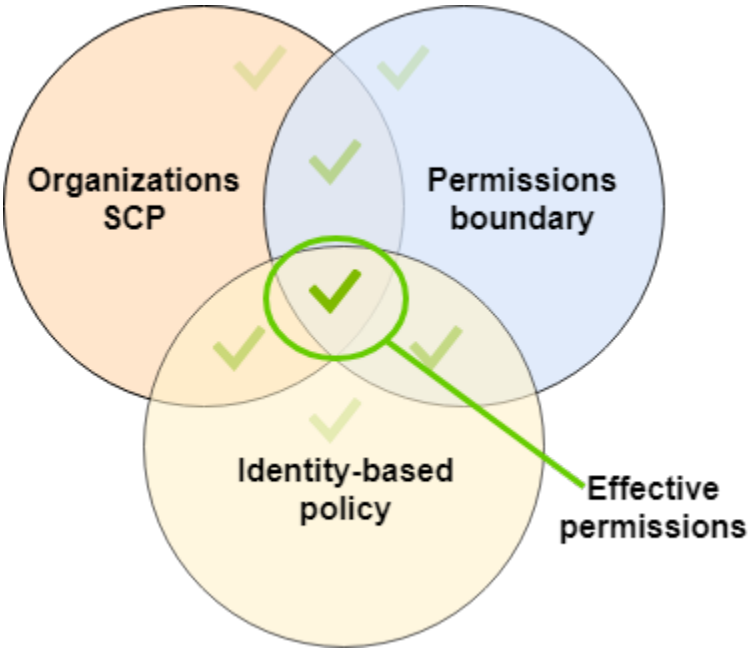
IAM 角色工作階段 – 在同一帳戶內，將許可授予 IAM 角色工作階段 ARN 的資源型政策會直接授予許可給擔任角色工作階段。直接授予工作階段的許可不會受到身分識別型政策、許可界限或工作階段政策中隱含拒絕的限制。當您擔任角色並提出要求時，提出請求的主體是 IAM 角色工作階段 ARN，而不是角色本身的 ARN。

適用於 IAM 聯合身分使用者的資源型政策

IAM 聯合身分使用者工作階段 – IAM 聯合身分使用者工作階段是透過呼叫 `GetFederationToken` 建立的工作階段。當聯合身分使用者提出要求時，提出要求的主體是聯合身分使用者 ARN，而不是聯合身分 IAM 使用者的 ARN。在相同的帳戶中，授予許可給聯合身分使用者 ARN 的資源型政策會直接將許可授予工作階段。直接授予工作階段的許可不會受到身分識別型政策、許可界限或工作階段政策中隱含拒絕的限制。

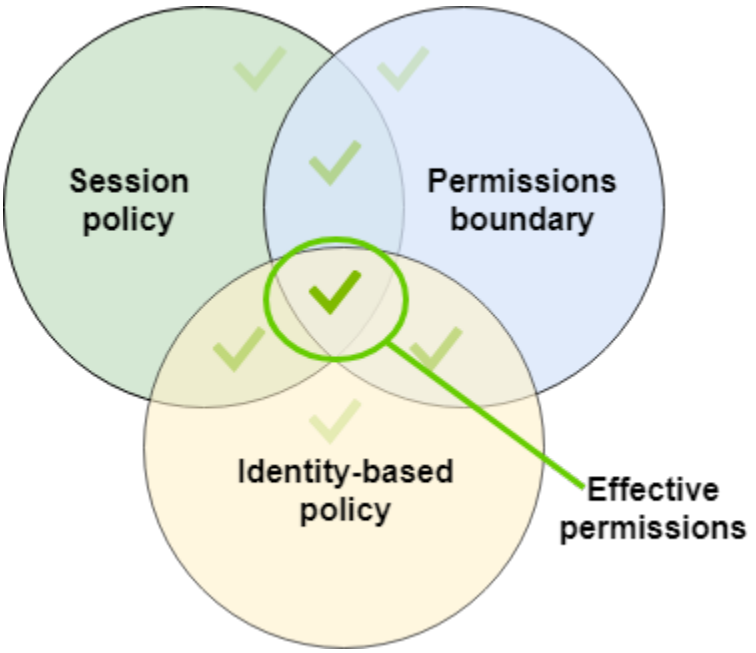
但是，如果資源型政策向聯合身分 IAM 使用者 ARN 授予許可，則聯合身分使用者在工作階段期間提出的要求會受到許可界限或工作階段政策中隱含拒絕的限制。

Organizations SCP – SCP 會套用到整個 AWS 帳戶。它們會限制由該帳戶內主體所提出各個請求的許可。IAM 實體 (使用者或角色) 可以提出一個要求，而這個要求會受到 SCP、許可邊界及身分類型政策的影響。在這種情況下，只有當所有三種政策類型允許時才允許該要求。有效許可是所有三種政策類型的交集。所有這類政策中的明確拒絕都會覆寫該允許。



您可以了解您的帳戶是否為 AWS Organizations中組織的成員。組織成員可能會受到 SCP 影響。若要使用 AWS CLI 命令或 AWS API 作業檢視此資料，您必須擁有 Organizations 實體`organizations:DescribeOrganization`動作的權限。您必須擁有其他許可，才能在 Organizations 主控台中執行操作。若要瞭解 SCP 是否拒絕存取特定要求，或是要變更您的有效權限，請聯絡您 AWS Organizations 的系統管理員。

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身分使用者的暫時工作階段時，做為參數傳遞。工作階段的許可，該工作階段的來源是建立該工作階段時所使用的 IAM 實體 (使用者或角色) 或工作階段政策。實體的以身分為基礎的政策，會受到工作階段政策及許可界限所限制。這組政策類型的有效許可是所有三種政策類型的交集。所有這類政策中的明確拒絕都會覆寫該允許。如需有關工作階段政策的詳細資訊，請參閱工作階段政策。



使用許可界限委派責任給他人

您可以使用許可界限來委派許可管理任務 (像是使用者建立) 給您的帳戶中的 IAM 使用者。這允許其他人在特定許可界限內代替您執行任務。

例如，假設瑪麗亞是 X AWS 帳戶公司的管理員。她想要委派使用者建立責任給 Zhang。不過，她必須確保 Zhang 建立的使用者遵守下列公司規則：

- 使用者不能使用 IAM 建立或管理使用者、群組、角色或政策。
- 使用者被拒絕存取 Amazon S3 `logs` 儲存貯體，且無法存取 `i-1234567890abcdef0` Amazon EC2 執行個體。
- 使用者無法移除自己的界限政策。

為了強制執行這些規則，María 完成以下任務，其包含以下詳細資訊：

1. María 建立 `XCompanyBoundaries` 受管政策以針對帳戶中所有的新的使用者，當成許可界限使用。
2. María 建立 `DelegatedUserBoundary` 受管政策，並將其指派為 Zhang 的許可界限。Maria 記下其管理員 使用者的 ARN，並在防止 Zhang 存取的政策中使用此 ARN。
3. María 建立 `DelegatedUserPermissions` 受管政策，並將其連接為 Zhang 的許可政策。
4. María 告訴 Zhang 有關他的新責任和限制。

任務 1：María 必須先建立受管政策來定義新使用者的界限。María 將允許 Zhang 提供使用者所需的許可政策，但她希望這些使用者受到限制。為了執行此操作，她建立以下客戶受管政策，名稱為 XCompanyBoundaries。此政策會執行下列動作：

- 可讓使用者完整存取數個服務
- 允許在 IAM 主控台中限制自行管理存取。這表示他們可以在登入主控台之後變更密碼。也無法設定其初始密碼。若要允許此操作，請將 "*LoginProfile" 動作加入 AllowManageOwnPasswordAndAccessKeys 陳述式。
- 拒絕對 Amazon S3 日誌儲存貯體或 i-1234567890abcdef0 Amazon EC2 執行個體的存取權限

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceBoundaries",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*",
                "dynamodb:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMConsoleForCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswordAndAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*ServiceSpecificCredential*",
```

```
                "iam:*SigningCertificate*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::logs",
                "arn:aws:s3:::logs/*"
            ]
        },
        {
            "Sid": "DenyEC2Production",
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
        }
    ]
}
```

每個陳述式提供不同的用途：

1. 此原則的ServiceBoundaries陳述式允許完整存取指定的 AWS 服務。這表示新使用者在這些服務的動作，僅受限於連接到使用者的許可政策。
2. AllowIAMConsoleForCredentials 陳述式允許列出所有 IAM 使用者的存取權。這個存取權是導覽 AWS Management Console中 Users (使用者) 頁面的必要條件。它也允許檢視帳戶的密碼要求，而這是變更您自身密碼時的必要程序。
3. AllowManageOwnPasswordAndAccessKeys 陳述式允許使用者僅管理自己的主控台密碼和程式設計存取金鑰。如果 Zhang 或另一個管理員為新使用者指派包含完整 IAM 存取的許可政策，這就顯得非常重要。在這種情況下，該使用者可能會變更自己的或其他使用者的許可。此陳述式可避免此種情況發生。
4. DenyS3Logs 陳述式明確拒絕存取 logs 儲存貯體。
5. DenyEC2Production 陳述式明確拒絕存取 i-1234567890abcdef0 執行個體。

任務 2：María 想要允許 Zhang 建立所有 X-Company 使用者，但只允許 XCompanyBoundaries 許可界限。她建立以下客戶受管政策，名稱為 DelegatedUserBoundary。此政策定義 Zhang 可以擁有的許可上限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateOrChangeOnlyWithBoundary",
            "Effect": "Allow",
            "Action": [
                "iam:AttachUserPolicy",
                "iam:CreateUser",
                "iam:DeleteUserPolicy",
                "iam:DetachUserPolicy",
                "iam:PutUserPermissionsBoundary",
                "iam:PutUserPolicy"
            ],
            "Resource": "*",
            "Condition": {
               "StringEquals": {
                 "iam:PermissionsBoundary": "arn:aws:iam::123456789012:policy/
XCompanyBoundaries"
                }
            }
        },
        {
            "Sid": "CloudWatchAndOtherIAMTasks",
            "Effect": "Allow",
            "Action": [
              "cloudwatch:*",
              "iam:CreateAccessKey",
              "iam:CreateGroup",
              "iam:CreateLoginProfile",
              "iam:CreatePolicy",
              "iam:DeleteGroup",
              "iam:DeletePolicy",
              "iam:DeletePolicyVersion",
              "iam:DeleteUser",
              "iam:GetAccountPasswordPolicy",
              "iam:GetGroup",
              "iam:GetLoginProfile",
              "iam:GetPolicy",
              "iam:GetPolicyVersion",
              "iam:GetRolePolicy",
              "iam:GetUser",
              "iam:GetUserPolicy",
```

```
            "iam:ListAccessKeys",
            "iam:ListAttachedRolePolicies",
            "iam:ListAttachedUserPolicies",
            "iam:ListEntitiesForPolicy",
            "iam:ListGroups",
            "iam:ListGroupsForUser",
            "iam:ListMFADevices",
            "iam:ListPolicies",
            "iam:ListPolicyVersions",
            "iam:ListRolePolicies",
            "iam:ListSSHPublicKeys",
            "iam:ListServiceSpecificCredentials",
            "iam:ListSigningCertificates",
            "iam:ListUserPolicies",
            "iam:ListUsers",
            "iam:SetDefaultPolicyVersion",
            "iam:SimulateCustomPolicy",
            "iam:SimulatePrincipalPolicy",
            "iam:UpdateGroup",
            "iam:UpdateLoginProfile",
            "iam:UpdateUser"
        ],
        "NotResource": "arn:aws:iam::123456789012:user/Maria"
    },
    {
        "Sid": "NoBoundaryPolicyEdit",
        "Effect": "Deny",
        "Action": [
            "iam:CreatePolicyVersion",
            "iam:DeletePolicy",
            "iam:DeletePolicyVersion",
            "iam:SetDefaultPolicyVersion"
        ],
        "Resource": [
            "arn:aws:iam::123456789012:policy/XCompanyBoundaries",
            "arn:aws:iam::123456789012:policy/DelegatedUserBoundary"
        ]
    },
    {
        "Sid": "NoBoundaryUserDelete",
        "Effect": "Deny",
        "Action": "iam:DeleteUserPermissionsBoundary",
        "Resource": "*"
    }
```

```
    ]
}
```

每個陳述式提供不同的用途：

1. CreateOrChangeOnlyWithBoundary 陳述式允許 Zhang 建立 IAM 使用者，但前提是他須使用 XCompanyBoundaries 政策來設定許可界限。此陳述式也可讓他為現有的使用者設定許可界限，但只能使用該相同政策。最後，此陳述式讓 Zhang 管理具此許可界限設定之使用者的許可政策。
2. CloudWatchAndOtherIAMTasks 陳述式允許 Zhang 完成其他使用者、群組和政策管理任務。其有權重設密碼，並為 NotResource 政策元素中未列出的任何 IAM 使用者建立存取金鑰。這可讓 Zhang 協助使用者解決登入問題。
3. NoBoundaryPolicyEdit 陳述式拒絕 Zhang 存取更新 XCompanyBoundaries 政策。他不被允許變更任何用於設定他自己或其他使用者的許可界限的政策。
4. NoBoundaryUserDelete 陳述式拒絕讓 Zhang 存取刪除他自己或其他使用者的許可界限。

然後，María 針對 Zhang 使用者將 DelegatedUserBoundary 政策分派許可界限。

任務 3：由於許可界限限制許可上限，但不自行授予存取權，因此 Maria 必須為 Zhang 建立許可政策。她建立以下政策，名稱為 DelegatedUserPermissions。此政策定義在已定義的界限內，Zhang 可以執行的操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "IAM",
            "Effect": "Allow",
            "Action": "iam:*",
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLimited",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetDashboard",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListDashboards",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "S3BucketContents",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::ZhangBucket"
        }
    ]
}
```

每個陳述式提供不同的用途：

1. 此政策的 IAM 陳述式允許 Zhang 完整存取 IAM。不過，由於他的許可界限僅允許部分的 IAM 操作，他的有效 IAM 許可僅受限於他的許可界限。
2. 該CloudWatchLimited聲明允許張在中執行五個操作 CloudWatch。他的權限邊界允許中的所有操作 CloudWatch，因此他的有效 CloudWatch 權限僅受其權限策略的限制。
3. S3BucketContents 陳述式允許 Zhang 列出 ZhangBucket Amazon S3 儲存貯體。不過，他的許可界限不允許任何 Amazon S3 動作，因此他無法執行任何 S3 操作，無論他是否有許可政策。

> **Note**
>
> Zhang 的政策允許其建立一個使用者，此使用者可存取 Zhang 無法存取的 Amazon S3 資源。透過委派這些管理動作，Maria 利用 Amazon S3 存取權以有效方式信任了 Zhang。

接著，María 將 DelegatedUserPermissions 政策連接為 Zhang 使用者的許可政策。

任務 4：她提供 Zhang 有關建立新使用者的指示。她通知他可以建立具備所需的任何許可的新使用者，但他必須將 XCompanyBoundaries 政策指派給他們做為許可界限。

Zhang 完成下列任務：

1. Zhang 透過 AWS Management Console建立使用者。他輸入使用者名稱 Nikhil，並啟用該使用者的主控台存取權。他清除了 Requires password reset (需要重設密碼) 旁的核取方塊，因為上述政策只允許使用者在登入 IAM 主控台後變更密碼。
2. 在「設定權限」頁面上，張氏會選擇允許 Nikhil 完成工作的 IAM FullAccess 和 AmazonS3 ReadOnlyAccess 許可政策。
3. Zhang 忘了 María 的指示，略過 Set permissions boundary (設定許可界限) 區段。

4. Zhang 檢閱使用者詳細資訊，然後選擇 Create user (建立使用者)。

   操作失敗，存取遭拒。Zhang 的 `DelegatedUserBoundary` 許可界限要求他所建立的任何使用者都將 `XCompanyBoundaries` 政策當做許可界限來使用。

5. Zhang 返回到之前的頁面。在 Set permissions boundary (設定許可界限) 區段中，他選擇了 `XCompanyBoundaries` 政策。
6. Zhang 檢閱使用者詳細資訊，然後選擇 Create user (建立使用者)。

   使用者已建立。

當 Nikhil 登入時，即有權存取 IAM 和 Amazon S3，除了許可界限拒絕的那些操作之外。例如，他可以在 IAM 變更自己的密碼，但無法建立其他使用者或編輯他的政策。Nikhil 只有 Amazon S3 的讀取存取權。

如果有人將資源類型政策新增至 `logs` 儲存貯體，而讓 Nikhil 在儲存貯體中放入物件，則他仍然無法存取該儲存貯體。原因在於對 `logs` 儲存貯體執行的任何動作，均會受到其許可界限的明確拒絕。任何政策類型中的明確拒絕，都會導致請求遭到拒絕。不過，如果 Secrets Manager 秘密連接的以資源為基礎的政策允許 Nikhil 執行 – `secretsmanager:GetSecretValue`動作，則 Nikhil 可以擷取和解密該秘密。原因在於這項許可界限並未明確拒絕上述 Secrets Manager 操作，而且許可界限的隱含拒絕不會限制資源類型政策。

以身分為基礎和以資源為基礎的政策

原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。當您建立許可政策來限制對資源的存取時，您可以選擇以身分為基礎的政策或以資源為基礎的政策。

以身分為基礎的政策會連接至 IAM 使用者、群組或角色。這些政策可讓您指定該身分可以執行哪些動作 (其許可)。例如，您可以將政策連接到名為 John 的 IAM 使用者，指出他已獲允許而可執行 Amazon EC2 `RunInstances` 動作。政策可進一步指出 John 已獲允許而可從名為 `MyCompany` 的 Amazon DynamoDB 資料表中取得項目。您也可以允許 John 管理他自己的 IAM 安全憑證。以身分為基礎的政策，可以受管理或內嵌。

以資源為基礎的政策會連接至資源。例如，您可以將以資源為基礎的政策附加到 Amazon S3 儲存貯體、Amazon SQS 佇列、VPC 端點、 AWS Key Management Service 加密金鑰以及 Amazon DynamoDB 表格和串流。如需有關可支援以資源為基礎之政策的服務清單，請參閱AWS 與 IAM 搭配使用的服務。

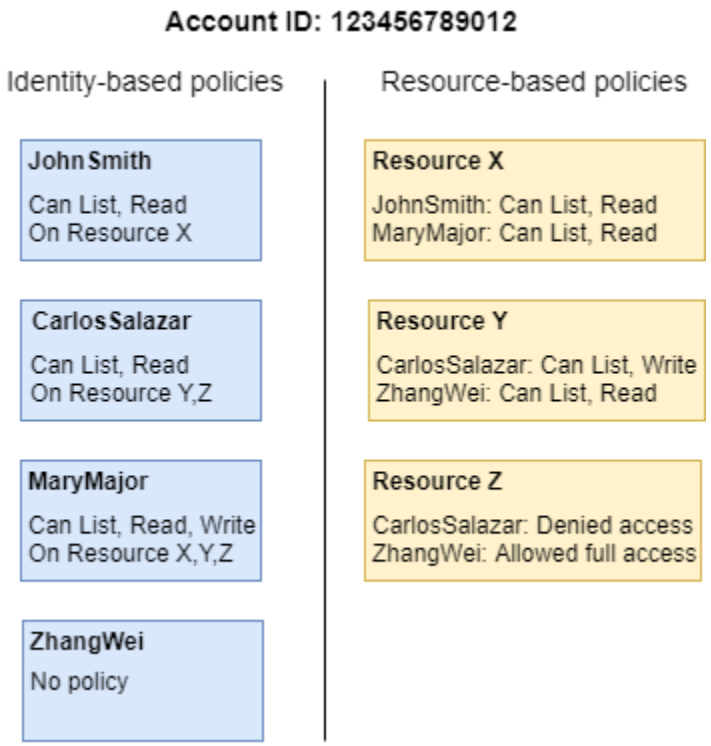
有了以資源為基礎的政策，您可以指定誰可以存取資源，以及他們可以在資源上執行哪些動作。若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。以資源為基礎的政策僅是內嵌，而不是受管理。

> **Note**
>
> 以資源為基礎的政策不同於資源層級的許可。您可以將以資源為基礎的政策直接連接到本主題中所述的資源。資源層級許可代表有能力使用 ARN 在政策中指定個別的資源。僅部分 AWS 服務支援以資源為基礎的政策。如需服務支援以資源為基礎的政策和資源層級許可的清單明細，請參閱AWS 與 IAM 搭配使用的服務。

若要了解以身分為基礎的政策和以資源為基礎的政策如何在相同帳戶內進行互動，請參閱 運用單一帳戶評估政策。

若要了解政策如何跨帳戶互動，請參閱 跨帳戶政策評估邏輯。

為了更了解這些概念，檢視這些圖表。123456789012 帳戶的管理員會將以身分為基礎的政策連接至JohnSmith、CarlosSalazar 和 MaryMajor 使用者。您可以對特定資源執行在這些政策中的一些動作。例如，使用者 JohnSmith 可以對 Resource X 執行一些動作。這是一個在以身分為基礎政策的資源層級許可。管理員也將以資源為基礎的政策新增至 Resource X、Resource Y 和 Resource Z。以資源為基礎的政策允許您指定誰可以存取該資源。例如，在 Resource X 上以資源為基礎的政策允許 JohnSmith 和 MaryMajor 使用者列出和讀取資源的存取權。



123456789012 帳戶範例允許以下使用者執行所列的動作：

- JohnSmith— John 可以在上執行清單和讀取動作Resource X。他是由以其使用者身分為基礎的政策和以資源為基礎的政策 Resource X 獲與此許可。
- CarlosSalazar— Carlos 可以對其執行列表、讀取和寫入操作Resource Y，但被拒絕訪問Resource Z。以 Carlos 之身分為基礎的政策允許可執行對 Resource Y 的列出和讀取動作。以 Resource Y 資源為基礎的政策允許擁有寫入許可。不過，雖然他的以身分為基礎的政策可允許存取 Resource Z，以 Resource Z 資源為基礎的政策會拒絕該存取。明確 Deny拒絕會覆寫 Allow，且他對 Resource Z 的存取會遭拒。如需詳細資訊，請參閱 政策評估邏輯。
- MaryMajor— Mary 可以在Resource X、和上執行清單、讀取和寫入作業Resource Z。Resource Y她的以身分為基礎的政策允許比以資源為基礎的政策對更多動作執行更多動作，但這些以資源為基礎的政策不會拒絕存取。
- ZhangWei— 張有完全的訪問權限Resource Z. Zhang 沒有以身分為基礎的政策，但以 Resource Z 資源為基礎的政策可允許完整存取資源。Zhang 也可以對 Resource Y 執行清單和讀取動作。

以身分為基礎的政策和以資源為基礎的政策都是許可政策，且會同時受到評估。對於只套用權限原則的要求，請 AWS 先檢查Deny. 如果有的話，則該請求會遭拒。然後， AWS 會檢查各個 Allow。如果至少有一個政策陳述式允許在請求中的動作，則該請求會受到允許。在以身分為基礎的政策中或以資源為基礎的政策中，是否有 Allow 並不重要。

> **Important**
>
> 此邏輯僅在單一 AWS 帳戶內做出該請求時適用。對於從一個帳戶對另一個帳戶所做的請求，在 Account A 中的請求者必須擁有以身分為基礎的政策，該政策能允許對 Account B 中的資源做出請求。此外，在 Account B 中以資源為基礎的政策，必須允許在 Account A 中的請求者存取資源。兩個帳戶中均須有允許操作的政策，否則請求會失敗。如需有關使用跨帳戶存取之以資源為基礎的政策詳細資訊，請參閱 IAM 中的跨帳戶資源存取。

具有特定許可的使用者可能請求資源，而該資源也有連接許可政策。在這種情況下，在決定是否授與資源存取權時， AWS 會評估這兩組權限。如需有關如何評估政策的詳細資訊，請參閱政策評估邏輯。

> **Note**
>
> Amazon S3 支援以身分為基礎的政策和以資源為基礎的政策 (稱為儲存貯體政策)。此外，Amazon S3 支援許可機制稱為存取控制清單 (ACL)，其為獨立的 IAM 政策和許可。您可以合併使用 IAM 政策與 Amazon S3 ACL。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的存取控制 。

使用政策控制對 AWS 資源的存取

您可以使用政策來控制 IAM 內或全部資源的存取 AWS。

若要使用原則來控制中的存取 AWS，您必須瞭解如何 AWS 授予存取權。 AWS 由資源集合組成。IAM 使用者即為資源。Amazon S3 儲存貯體即為資源。當您使用 AWS API AWS CLI、或執行作業 (例如建立使用者) 時，您會傳送該作業的要求。 AWS Management Console 您的請求指定動作、資源、主體實體 (使用者或角色)、主體帳戶，以及所需的任何請求資訊。所有這些資訊提供了上下文。

AWS 然後檢查您（主體）是否通過身份驗證（已登錄）並授權（有權限）對指定的資源執行指定的操作。在授權期間， AWS 會檢查適用於請求內容的所有原則。大多數原則會 AWS 以 JSON 文件的形式儲存在中，並指定主體實體的權限。如需有關政策類型及其使用的詳細資訊，請參閱 IAM 中的政策和許可。

AWS 只有在政策允許您的請求的每個部分時，才會授權請求。若要檢視此程序的圖表，請參閱 IAM 的運作方式。如需如何 AWS 判斷是否允許請求的詳細資訊，請參閱政策評估邏輯。

當您建立 IAM 政策時，可以控制對下列項目的存取：

- 主體 – 控制允許發出請求的人員 (主體) 執行哪些操作。
- IAM 身分 – 控制哪些 IAM 身分 (使用者群組、使用者與角色) 可被存取以及存取的方法。
- IAM政策 – 控制哪些使用者可以建立、編輯和刪除客戶受管政策，以及哪些使用者可以連接和分開所有受管政策。
- AWS 資源 – 控制哪些使用者有權使用以身分為基礎的政策或以資源為基礎的政策來存取資源。
- AWS 帳戶 – 控制是否僅允許特定帳戶的成員發出請求。

策略可讓您指定誰可以存取 AWS 資源，以及他們可以對這些資源執行哪些動作。每個 IAM 使用者在開始時都沒有許可。換言之，在預設狀態下，使用者無法執行任何動作，甚至不能查看自己的存取金鑰。若要為使用者授予執行某些操作的許可，您可以為使用者新增許可 (也就是將政策連接到使用者)。或者，您可以將使用者加入到具有預期的許可的使用者群組中。

例如，您可能授予使用者列出自己的存取金鑰的許可。您可能擴展該許可，並讓每位使用者建立、更新和刪除自己的索引鍵。

當您授予一個使用者群組許可時，使用者群組內的全部使用者都會獲得那些許可。例如，您可以授與管理員使用者群組權限，以便對任何 AWS 帳戶 資源執行任何 IAM 動作。另一個範例：您可以授予 Managers 使用者群組描述 AWS 帳戶的 Amazon EC2 執行個體的許可。

有關如何委託基本許可給使用者、使用者群組和角色的資訊，請參閱 存取 IAM 資源所需的許可。有關說明基本許可的其他政策範例，請參閱 管理 IAM 資源的政策範例。

控制主體的存取許可

您可以使用政策，控制允許發出請求的人員 (主體) 執行哪些操作。若要執行此操作，您必須將以身分為基礎的政策連接到此人的身分 (使用者、使用者群組或角色)。您也可以使用許可界限，設定實體 (使用者或角色) 可擁有的最大許可。

例如，假設您希望使用者張偉擁有亞馬遜 DynamoDB CloudWatch、亞馬遜 EC2 和 Amazon S3 的完整存取權。您可以建立兩個不同的政策，以便以後其他使用者需要一組許可時為其分配其中的一個政策。或者，您可以將兩個許可放在單一政策中，然後將該政策連接到名為 Zhang Wei 的 IAM 使用者。您還可以將一個政策連接到 Zhang 所屬的使用者群組，或者連接到 Zhang 可以擔任的角色。因此，在 Zhang 查看 S3 儲存貯體內容時，會允許他的請求。如果他嘗試建立新的 IAM 使用者，他的請求會被拒絕，因為他未獲許可。

您可以在 Zhang 身上使用許可界限，確定他絕對不會被授予 `DOC-EXAMPLE-BUCKET1` S3 儲存貯體存取權。若要執行此操作，請決定您希望 Zhang 擁有的許可上限。在這種情況下，您可以控制他使

用許可政策做哪些事。在這裡，您只在意他不會存取機密儲存貯體。因此，您可以使用下列政策定義 Zhang 的界限，以允許 Amazon S3 和其他一些服務的所有 AWS 動作，但拒絕存取 DOC-EXAMPLE-BUCKET1 S3 儲存貯體。由於許可界限不允許任何 IAM 動作，如此可防止 Zhang IAM 刪除他 (或任何人的) 界限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsBoundarySomeServices",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "dynamodb:*",
                "ec2:*",
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PermissionsBoundaryNoConfidentialBucket",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
            ]
        }
    ]
}
```

當您針對使用者指派像這類許可界限的政策時，請記得，並沒有授予任何許可。它負責設定以身分為基礎的政策可為 IAM 實體授予的最大許可。如需有關許可界限的詳細資訊，請參閱 IAM 實體的許可界限。

如需之前所述的程序詳細資訊，請參考以下資源：

- 若要進一步了解有關建立可連接到主體的 IAM 政策的資訊，請參閱 建立 IAM 政策。
- 若要了解將 IAM 政策連接到主體的方法，請參閱 新增和移除 IAM 身分許可。
- 若要查看授予 EC2 完全存取許可的範例政策，請參閱 Amazon EC2：允許在特定區域內，以程式設計方式和在主控台進行 EC2 完全存取。

- 允許對 S3 儲存貯體進行唯讀存取，請使用以下範例政策的前兩個說明：Amazon S3：允許以程式設計方式和在主控台以讀取和寫入存取 S3 儲存貯體中的物件。
- 查看範例政策，以便讓使用者設定其憑證，例如其主控台密碼、程式設計存取金鑰及 MFA 裝置，請參閱 AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

控制對身分的存取

您可以透過使用者群組來建立連接到使用者的 IAM 政策，以使用該政策控制您的所有使用者可對某個身分執行哪些操作。若要執行此操作，請建立一個政策以限制可對某個身分執行哪些操作，以及哪些使用者可以存取該身分。

例如，您可以建立名為的使用者群組 AllUsers，然後將該使用者群組附加至所有使用者。在建立該使用者群組時，您可以為所有使用者授予存取權限以設定其憑證，如前一節中所述。然後，您可以建立一個政策，以便拒絕存取以禁止更改該使用者群組，除非在政策條件中包含使用者名稱。但是，該政策部分僅拒絕列出的使用者以外的任何人進行存取。您也必須包含許可權，以允許該使用者群組中的所有使用者可執行所有使用者群組管理動作。最後，將該政策連接到該使用者群組，以便將其套用於所有使用者。因此，在政策中未指定的使用者嘗試更改該使用者群組時，將拒絕請求。

若要使用視覺化編輯器來建立此政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。

   如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。
3. 選擇 Create policy (建立政策)。
4. 在政策編輯器區段中，選擇視覺化選項。
5. 在選擇服務中，選擇 IAM。
6. 在允許的動作中，在搜尋方塊中輸入 **`group`**。視覺化編輯器將顯示所有包含 `group` 一詞的 IAM 動作。選取所有核取方塊。
7. 選擇 Resources (資源) 來為您的政策指定資源。根據您所選的動作，應會看到群組和使用者資源類型。

   - 群組 – 選擇新增 ARN。針對資源位於，選取任何帳號選項。選取任何具有路徑的群組名稱核取方塊，然後輸入使用者群組名稱 **`AllUsers`**。接著選擇新增 ARN。
   - 使用者 – 選取此帳戶中的任何旁的核取方塊。

您選擇的操作之一 (`ListGroups`) 不支援使用特定的資源。您不需要為該動作選擇 All resources (所有資源)。在 JSON 編輯器中儲存或查看您的政策時，您可以看到 IAM 自動建立新的許可區塊，以便為所有資源授予該動作的許可。

8. 若要新增其他許可區塊，請選擇新增更多許可。
9. 選擇選取服務，然後選擇 IAM。
10. 選擇允許的動作，然後選擇切換為拒絕許可。在執行該操作時，將使用整個區塊來拒絕許可。
11. 在搜尋方塊中，輸入 **`group`**。視覺化編輯器將顯示所有包含 `group` 一詞的 IAM 動作。選取下列動作旁的核取方塊：

    - CreateGroup
    - DeleteGroup
    - RemoveUserFromGroup
    - AttachGroupPolicy
    - DeleteGroupPolicy
    - DetachGroupPolicy
    - PutGroupPolicy
    - UpdateGroup
12. 選擇 Resources (資源) 來為您的政策指定資源。根據您所選的動作，應會看到群組資源類型。選擇新增 ARN。針對資源位於，選取任何帳號選項。針對任何具有路徑的群組名稱，輸入使用者群組名稱 **`AllUsers`**。接著選擇新增 ARN。
13. 選擇請求條件 - 選用，然後選擇新增另一個條件。以下列的值來完成表單：

    - 條件金鑰 - 選擇 aws：使用者名稱
    - Qualifier (限定詞) – 選擇 Default (預設)
    - 運算子 — 選擇 StringNotEquals
    - 值 – 輸入 **`srodriguez`** 然後選擇新增，以新增另一個值。輸入 **`mjackson`**，然後選擇新增，以新增另一個值。輸入 **`adesai`**，然後選擇新增條件。

    在清單中不包含進行呼叫的使用者時，該條件將確認拒絕存取指定的使用者群組管理動作。由於這會明確拒絕許可權，它將覆寫前面允許這些使用者呼叫這些操作的區塊。不會拒絕清單中的使用者進行存取，並在第一個許可區塊中為他們授予許可，因此，他們可以完全管理該使用者群組。
14. 完成時，選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器選項中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

15. 在檢視與建立頁面上，針對政策名稱輸入 **`LimitAllUserGroupManagement`**。針對 Description (描述)，輸入 **`Allows all users read-only access to a specific user group, and allows only specific users access to make changes to the user group`**。檢視該政策中定義的許可，來確認您已授予想要的許可。然後選擇 Create policy (建立政策) 來儲存您的新政策。
16. 將政策連接到您的使用者群組。如需詳細資訊，請參閱 新增和移除 IAM 身分許可。

或者，您也可以使用該範例 JSON 政策文件建立相同的政策。若要檢視此 JSON 政策，請參閱 IAM：允許特定 IAM 使用者以程式設計方式在主控台中管理群組。如需關於使用 JSON 文件來建立政策的詳細說明，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

控制對政策的存取

您可以控制使用者套用 AWS 受管策略的方式。若要執行此操作，請將該政策連接到您的所有使用者。理想情況下，您可以使用使用者群組執行該操作。

例如，您可以建立一個政策，讓使用者僅將 IAM UserChangePassword 和PowerUserAccess AWS 受管政策附加到新的 IAM 使用者、使用者群組或角色。

對於客戶受管政策，您可以控制哪些使用者可以建立、更新和刪除這些政策。您可以控制哪些使用者可以將政策連接到主體實體 (使用者群組、使用者和角色) 以及將政策從主體實體中分開。您也可以控制使用者可以對哪些實體連接或分開哪些政策。

例如，您可以為帳戶管理員授予許可以建立、更新和刪除政策。然後，您可以為小組負責人或其他有限管理員授予許可，以便將這些政策連接到該有限管理員管理的主體實體以及將這些政策從這些主體實體中分開。

如需詳細資訊，請參閱下列資源：

- 若要進一步了解有關建立可連接到主體的 IAM 政策的資訊，請參閱 建立 IAM 政策。
- 若要了解將 IAM 政策連接到主體的方法，請參閱 新增和移除 IAM 身分許可。

- 若要查看限制使用受管政策的範例政策，請參閱 IAM：限制可套用至 IAM 使用者、群組或角色的受管政策。

控制建立、更新和刪除客戶受管政策的許可

您可以使用 IAM 政策控制誰有權在您的 AWS 帳戶中建立、更新和刪除客戶管理政策。以下清單包含與建立、更新和刪除政策或政策版本直接相關的 API 操作：

- CreatePolicy
- CreatePolicyVersion
- DeletePolicy
- DeletePolicyVersion
- SetDefaultPolicyVersion

上述清單中的 API 操作對應於可使用 IAM 政策允許或拒絕的操作 — 也就是您可以授予的許可。

請考量下列範例政策。允許使用者建立、更新 (即建立新的政策版本)、刪除 AWS 帳戶中的所有客戶管理政策以及設定這些政策的預設版本。該範例政策還允許該使用者列出政策並獲取政策。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

Example 允許建立、更新、刪除、列出、獲取所有政策以及設定這些政策的預設版本的政策範例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:CreatePolicy",
      "iam:CreatePolicyVersion",
      "iam:DeletePolicy",
      "iam:DeletePolicyVersion",
      "iam:GetPolicy",
      "iam:GetPolicyVersion",
      "iam:ListPolicies",
      "iam:ListPolicyVersions",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
```

```
  }
}
```

您可以建立限制使用這些 API 操作的政策以僅影響指定的受管政策。例如，您可能希望允許使用者設定預設版本和刪除政策版本，但是僅允許針對特定客戶受管政策執行這些操作。運用在授予這些許可權的政策的 Resource 元素中指定政策 ARN，可實現此目的。

以下範例顯示的政策，可讓使用者刪除政策版本，並設定預設版本。但是，這些動作只允許用於客戶受管政策，其包含路徑 /TEAM-A/。客戶受管政策 ARN 是在政策的 Resource 元素中指定 (在此範例中，ARN 包含一個路徑和一個萬用字元，因而可配對包含路徑 /TEAM-A/ 的所有客戶受管政策)。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

如需有關如何使用該範例 JSON 政策文件建立政策，請參閱 易用名稱和路徑。

Example 僅允許針對特定政策刪除政策版本和設定預設版本的政策範例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:DeletePolicyVersion",
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "arn:aws:iam::account-id:policy/TEAM-A/*"
  }
}
```

控制連接和分開受管政策的許可

您還可以使用 IAM 政策僅允許使用者使用特定的受管政策。實際上，您可以控制允許使用者為其他主體實體授予哪些許可。

以下清單顯示與將受管政策連接到主體實體以及從中分開直接相關的 API 操作：

- AttachGroupPolicy
- AttachRolePolicy
- AttachUserPolicy
- DetachGroupPolicy

- DetachRolePolicy
- DetachUserPolicy

您可以建立限制使用這些 API 操作的政策以僅影響特定的受管政策與/或您指定的主體實體。例如，您可能希望允許使用者連接受管政策，但是只能連接您指定的受管政策。或者，您可能希望允許使用者連接受管政策，但是只能連接到您指定的主體實體。

以下範例政策僅允許使用者將受管政策連接到包含路徑 /TEAM-A/ 的使用者群組和角色。使用者群組和角色 ARN 是在政策的 Resource 元素中指定的。(在此範例中，ARN 包含一個路徑和一個萬用字元，因而與包含路徑 /TEAM-A/ 的所有使用者群組和角色配對。) 若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

Example 僅允許將受管政策連接到特定使用者群組或角色的政策範例

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::account-id:group/TEAM-A/*",
      "arn:aws:iam::account-id:role/TEAM-A/*"
    ]
  }
}
```

您可以進一步限制前面範例中的動作，讓它僅影響特定的政策。也就是，透過在政策中新增條件，您可以控制允許使用者連接到其他主體實體的許可。

以下範例中的條件可確保只有在連接的政策與指定政策之一配對時才允許 AttachGroupPolicy 與 AttachRolePolicy 許可。條件使用iam:PolicyARN 條件金鑰來決定要連接哪個政策或哪些政策。以下範例政策延伸之前的範例。其允許使用者僅將包含路徑 /TEAM-A/ 的受管政策連接到包含路徑 /TEAM-A/ 的使用者群組和角色。若要了解如何使用此範例 JSON 政策文件來建立政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:AttachGroupPolicy",
      "iam:AttachRolePolicy"
    ],
    "Resource": [
      "arn:aws:iam::account-id:group/TEAM-A/*",
      "arn:aws:iam::account-id:role/TEAM-A/*"
    ],
    "Condition": {"ArnLike":
      {"iam:PolicyARN": "arn:aws:iam::account-id:policy/TEAM-A/*"}
    }
  }
}
```

本政策使用 ArnLike 條件運算子，但也可以使用 ArnEquals 條件運算子，因為這兩個條件運算子執行相同的操作。如需有關 ArnLike 與 ArnEquals 的詳細資訊，請參閱《政策元素參考》條件類型章節中的 Amazon Resource Name (ARN) 條件運算子。

例如，您可以限制使用操作以僅涉及指定的受管政策。運用在授予這些許可權的政策的 Condition 元素中指定政策 ARN，可實現此目的。例如，指定客戶受管政策的 ARN：

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::123456789012:policy/POLICY-NAME"}
}
```

您也可以在策略的Condition元素中指定 AWS 受管理策略的 ARN。 AWS 受管理策略的 ARN 使用策略 ARN aws 中的特殊別名而不是帳戶 ID，如下列範例所示：

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"}
}
```

控制資源的存取許可

您可以控制哪些使用者有權使用以身分為基礎的政策或以資源為基礎的政策來存取資源。在以身分為基礎的政策中，您將政策連接到一個身分並指定該身分可以存取哪些資源。在以資源為基礎的政策中，您將政策連接到要控制的資源。在該政策中，您指定哪些主體可以存取該資源。如需有關兩種政策類型的詳細資訊，請參閱 以身分為基礎和以資源為基礎的政策。

如需詳細資訊，請參閱下列資源：

- 若要進一步了解有關建立可連接到主體的 IAM 政策的資訊，請參閱 建立 IAM 政策。
- 若要了解將 IAM 政策連接到主體的方法，請參閱 新增和移除 IAM 身分許可。
- Amazon S3 支援在儲存貯體上使用以資源為基礎的政策。如需詳細資訊，請參閱儲存貯體政策範例。

資源建立者不會自動擁有許可

如果您使用 AWS 帳戶根使用者 認證登入，則您有權對屬於該帳戶的資源執行任何動作。但是，對 IAM 使用者來說並非如此。IAM 使用者可以獲得建立資源的許可，但即使對於該資源，該使用者許可也僅限於明確授予的內容。這表示您不會只因為建立資源 (如 IAM 角色) 就自動具有編輯或刪除該角色的許可。此外，帳戶擁有者或有權管理您的許可的其他使用者可以隨時撤銷您的許可。

控制對特定帳戶中的主體的存取

您可以直接向自己帳戶中的 IAM 使用者授予對您的資源的存取權限。如果來自另一個帳戶的使用者需要存取您的資源，您可以建立 IAM 角色。角色是一個實體，其中包含許可，但不與特定使用者關聯。然後，其他帳戶中的使用者可以擔任該角色，並根據您為該角色分配的許可存取資源。如需詳細資訊，請參閱 在您擁有的另一個 AWS 帳戶 IAM 使用者中提供存取權。

> **Note**
>
> 部分服務支援以資源為基礎的政策，如 以身分為基礎和以資源為基礎的政策 (例如 Amazon S3、Amazon SNS 和 Amazon SQS) 中所述。對於這些服務，使用角色的替代方法是將政策連接到您要共用的資源 (儲存貯體、主題或佇列)。以資源為基礎的策略可以指定具有資源存取權限的 AWS 帳號。

使用標籤控制對 IAM 使用者和角色的存取

使用以下區段中的資訊來控制可以存取 IAM 使用者和角色的人員，以及使用者和角色可以存取的資源。如需控制其他 AWS 資源存取權的一般資訊和範例，包括其他 IAM 資源，請參閱標記 IAM 資源。

> **Note**
>
> 如需有關標籤索引鍵和標籤索引鍵值區分大小寫的詳細資訊，請參閱 Case sensitivity。

您可以將標籤連接至 IAM 資源、傳遞至請求，或連接至提出請求的主體。IAM 使用者或角色可以同時是資源和主體。例如，您可以編寫政策，此政策會允許使用者列出使用者的群組。只有在發出請求的使用者 (主體) 擁有與其嘗試檢視的使用者相同的 `project=blue` 標籤時，才允許此操作。在這個範例中，只要使用者處理的是同一個專案，他們就可以檢視任何使用者 (包括自己) 的群組成員資格。

若要根據標籤控制存取，則在政策的條件元素中提供標籤資訊。建立 IAM 政策時，您可以使用 IAM 標籤與關聯的標籤條件索引鍵，以控制下列任何項目的存取權：

- 資源 – 根據使用者或角色資源的標籤控制對這些資源的存取權。為此，請使用 aws:ResourceTag/######來指定哪些標籤鍵值對必須附加到資源。如需詳細資訊，請參閱 控制 AWS 資源的存取許可。
- 請求 – 控制您可以在 IAM 請求條件中傳遞哪些標籤。若要這麼做，請使用 aws:RequestTag/金###條件金鑰來指定可從 IAM 使用者或角色新增、變更或移除哪些標籤。此金鑰的使用方式與 IAM 資源和其他 AWS 資源的使用方式相同。如需詳細資訊，請參閱 在 AWS 請求期間控制存取許可。
- 主體 – 根據連接至該人員 IAM 使用者或角色的標籤，控制提出請求的人員 (主體) 可執行的操作。若要這麼做，請使用 aws:PrincipalTag/金###條件金鑰來指定在允許要求之前，必須將哪些標籤附加至 IAM 使用者或角色。
- 授權程序的任何部分 — 使用 aws: TagKeys condition 金鑰控制是否可以在請求中或主體使用特定標籤金鑰。在此情況下，此鍵值無關緊要。此金鑰的行為與 IAM 和其他 AWS 服務類似。不過，當您在 IAM 中標記使用者時，這也會控制主體是否可以對任何服務發出請求。如需詳細資訊，請參閱 根據標籤索引鍵控制存取權限。

您可以使用視覺編輯器、使用 JSON，或匯入現有的受管政策，來建立 IAM 政策。如需詳細資訊，請參閱 建立 IAM 政策。

> **Note**
>
> 您也可以在擔任 IAM 角色或與使用者聯合身分時傳遞工作階段標籤。這些標籤僅在工作階段的時長內有效。

控制 IAM 主體的存取權限

您可以根據在該人員身分中連接的標籤，控制主體可執行的操作。

此範例會示範如何建立身分型政策，允許此帳戶中的任何使用者檢視任何使用者的群組成員資格，包括他們自己，只要他們正在處理同一個專案。只有在使用者的資源標籤和主體的標籤具有相同標籤鍵值 `project` 時，才允許此操作。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:ListGroupsForUser",
            "Resource": "arn:aws:iam::111222333444:user/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/project":
 "${aws:PrincipalTag/project}"}
            }
        }]
}
```

根據標籤索引鍵控制存取權限

您可以在 IAM 政策中使用標籤來控制是否可以在請求中或主體使用特定標籤金鑰。

此範例會示範如何建立身分型政策，僅允許從使用者中刪除具有 `temporary` 索引鍵的標籤。若要使用此政策，請將範例政策中的*########*取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:UntagUser",
        "Resource": "*",
        "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": ["temporary"]}}
    }]
}
```

使用標籤控制 AWS 資源的存取

您可以使用標籤來控制對支援標記 (包括 IAM AWS 資源) 之資源的存取。您可以標記 IAM 使用者和角色，來控制他們可以存取的內容。若要了解如何標記 IAM 使用者和角色，請參閱 標記 IAM 資源。此外，您可以控制對下列 IAM 資源的存取：客戶管理政策、IAM 身分提供者、執行個體設定檔、伺服器憑證和虛擬 MFA 裝置。若要檢視建立和測試允許具有主體標籤的 IAM 角色政策，以相符標籤存取資源的教學，請參閱 IAM 教學課程：根據標籤定義存取 AWS 資源的許可。使用下一節中的資訊來控制對其他 AWS 資源 (包括 IAM 資源) 的存取，而無需標記 IAM 使用者或角色。

在您使用標籤來控制對 AWS 資源的存取之前，您必須瞭解如何 AWS 授予存取權。 AWS 由資源集合組成。Amazon EC2 執行個體是一種資源。Amazon S3 儲存貯體即為資源。您可以使用 AWS API AWS CLI、或執行作業，例如在 Amazon S3 中建立儲存貯體。 AWS Management Console 當您這麼做時，您會為該操作傳送請求。您的請求指定動作、資源、主體實體 (使用者或角色)、主體帳戶，以及所需的任何請求資訊。所有這些資訊提供了上下文。

AWS 然後檢查您（主體實體）是否通過身份驗證（登錄）和授權（有權限）對指定的資源執行指定的操作。在授權期間， AWS 會檢查適用於請求內容的所有原則。大多數原則會 AWS 以 JSON 文件的形式儲存在中，並指定主體實體的權限。如需有關政策類型及其使用的詳細資訊，請參閱 IAM 中的政策和許可。

AWS 只有在政策允許您的請求的每個部分時，才會授權請求。若要查看圖表和進一步了解 IAM 基礎設施的詳細資訊，請參閱 IAM 的運作方式。有關 IAM 如何決定是否允許請求的詳細資訊，請參閱 政策評估邏輯。

標籤是這個程序中的另一項考慮因素，因為您可以將標籤連接到資源或在請求中將標籤傳遞至支援標籤的服務。若要根據標籤控制存取，則在政策的條件元素中提供標籤資訊。若要瞭解 AWS 服務是否支援使用標記控制存取權，請參閱AWS 與 IAM 搭配使用的服務並尋找 ABAC 資料行中具有 [是] 的服務。選擇服務的名稱，以檢視該服務的授權和存取控制文件。

然後，您可以建立 IAM 政策，這類政策可根據該資源的標籤允許或拒絕對某資源的存取。在該政策中，您可以使用標籤條件金鑰，以控制執行下列項目的存取權：

- 資源 — 根據這些資源上的標籤控制對 AWS 服務資源的存取。要執行此操作，請使用 ResourceTag/***##***條件鍵來確定是否允許根據附加到資源的標籤訪問資源。
- 請求 – 控制您可以在請求中傳遞哪些標籤。要做到這一點，請使用 aws:RequestTag/***key name*** 條件鍵來指定可以在請求中傳遞哪些標籤鍵值對來標記資源。 AWS
- 授權程序的任何部分 — 使用 aws: TagKeys condition 金鑰控制特定標籤金鑰是否可以在請求中。

您可以使用 JSON，或匯入現有的受管政策，透過視覺化的方式來建立 IAM 政策。如需詳細資訊，請參閱 建立 IAM 政策。

> **Note**
>
> 如果使用者有權使用建立資源的動作，部分服務允許使用者在建立資源時指定標籤。

控制 AWS 資源的存取許可

您可以使用 IAM 政策中的條件，根據該 AWS 資源上的標籤來控制對資源的存取。作法是使用全域 `aws:ResourceTag/tag-key` 條件金鑰，或服務特定索引鍵。某些服務僅支援此索引鍵的服務特定版本，而不支援全域版本。

> **Warning**
>
> 請勿嘗試透過標記角色，然後使用政策中的 ResourceTag 條件金鑰搭配 iam:PassRole 動作來控制誰可以傳遞角色。這種方法所產生的結果並不可靠。如需有關傳遞角色至服務之必要許可的詳細資訊，請參閱 授予使用者將角色傳遞至 AWS 服務的許可。

此範例會示範如何建立身分型政策，它允許啟動或停止 Amazon EC2 執行個體。只有在 Owner 執行個體標籤具有使用者名稱值時，才允許這些操作。此政策定義了程式設計和主控台存取的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果使用者指定 `richard-roe` 嘗試啟動 Amazon EC2 執行個體，您必須為該執行個體加上標籤 `Owner=richard-roe` 或 `owner=richard-roe`。否

則，他將會收到存取遭拒。標籤鍵 Owner 同時符合 Owner 和 owner，因為條件金鑰名稱不區分大小寫。如需詳細資訊，請參閱 IAM JSON 政策元素：Condition。

此範例會示範如何建立在資源 ARN 中使用 team 主體標籤的身分型政策。該政策會授與刪除 Amazon Simple Queue Service 佇列的許可，但前提是佇列名稱必須以團隊名稱為開頭，並在後面接著 -queue。例如，qa-queue if qa 是 team 主體標籤的團隊名稱。

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Sid": "AllQueueActions",
        "Effect": "Allow",
        "Action": "sqs:DeleteQueue",
        "Resource": "arn:aws:sqs:us-east-2::${aws:PrincipalTag/team}-queue"
      }
}
```

在 AWS 請求期間控制存取許可

您可以使用 IAM 政策中的條件來控制哪些標籤鍵值配對可以在應用標記 AWS 資源的請求中傳遞。

此範例會示範如何建立身分型政策，它允許使用 Amazon EC2 CreateTags 動作以將標籤連接至執行個體。只有在標籤包含 environment 金鑰和 preprod 或 production 值時，您才能連接標籤。您可以視需要將 ForAllValues 修飾詞與 aws:TagKeys 條件金鑰搭配使用，表示只允許在請求中使用 environment 索引鍵。這會使得使用者無法包含其他金鑰，例如意外使用 Environment 而非 environment。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "ec2:CreateTags",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/environment": [
                    "preprod",
                    "production"
                ]
            },
            "ForAllValues:StringEquals": {"aws:TagKeys": "environment"}
```

```
        }
    }
}
```

根據標籤索引鍵控制存取權限

您可以在 IAM 政策中使用條件，以控制是否可在請求中使用特定標籤鍵。

我們建議您使用原則來控制使用標籤的存取時，請使用aws:TagKeys條件索引鍵。 AWS 支援標籤的服務可能允許您建立多個僅因大小寫而不同的標籤金鑰名稱，例如使用stack=production和標記 Amazon EC2 執行個體Stack=test。鍵名稱在政策條件中不區分大小寫。這表示如果您在政策的條件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，則該條件會符合名為 TagKey1 或 tagkey1 的資源標籤鍵 (但不會同時符合兩者)。未免出現鍵大小寫差異導致的重複標籤，請使用 aws:TagKeys 條件來定義使用者可以套用的標籤鍵，或使用 AWS Organizations提供的標籤政策。如需詳細資訊，請參閱《Organizations 使用者指南》中的標籤政策。

此範例會示範如何建立身分型政策，它允許建立和標記 Secrets Manager 密碼，但僅能使用標籤索引鍵 environment 或 cost-center。Null 條件可確保如果請求中沒有標籤，條件將評估為 false。

```
{
        "Effect": "Allow",
        "Action": [
            "secretsmanager:CreateSecret",
            "secretsmanager:TagResource"
        ],
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:TagKeys": "false"
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "environment",
                    "cost-center"
                ]
            }
        }
}
```

IAM 中的跨帳戶資源存取

對於某些 AWS 服務，您可以使用 IAM 授予跨帳戶對資源的存取權。若要執行此操作，您可將直接將資源政策連接到您要分享的資源，或將角色用作代理。

若要直接分享資源，您要分享的資源必須支援資源型政策。與角色的身分型政策不同，資源型政策指定誰 (主體) 可以存取該資源。

如果想要存取不支援資源型政策之其他帳戶中的資源，則請將角色用作代理。

如需有關這些政策類型之間差異的詳細資訊，請參閱 以身分為基礎和以資源為基礎的政策。

> **Note**
>
> IAM 角色和資源型政策只會在單一分割內跨帳戶委派存取許可。例如，您在標準 aws 分割區的美國西部 (加利佛尼亞北部) 中有一個帳戶。您在 aws-cn 分割區的中國也有一個帳戶。您不能在中國的帳戶中使用基於資源的政策來允許標準 AWS 帳戶中的用戶訪問。

使用角色進行跨帳戶存取

並非所有 AWS 服務都支援資源型政策。對於這些服務，在向多個服務提供跨帳户存取權時，可以使用跨帳戶 IAM 角色集中管理許可。跨帳戶 IAM 角色是 IAM 角色，其中包含信任政策，允許其他 AWS 帳戶中的 IAM 主體擔任該角色。簡而言之，您可以在一個 AWS 帳戶中創建一個角色，該角色將特定權限委託給另一個 AWS 帳戶。

如需有關將政策連接至 IAM 身分的資訊，請參閱 管理 IAM 政策。

> **Note**
>
> 當主體切換到某個角色以暫時使用角色的許可時，他們會放棄其原始許可，並接受指派給他們假設之角色的許可。

讓我們看一下整體流程，因為其適用於需要存取客戶帳戶的 APN 合作夥伴軟體。

1. 客戶在自己的帳戶中使用政策建立 IAM 角色，以允許存取 APN 合作夥伴所需的 Amazon S3 資源。在此範例中，角色名稱為 APNPartner。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

2. 然後，客戶透過在角色的信任政策中提供 APN 合作夥伴的 AWS 帳戶 ID，指定該角色可由合作夥伴的 AWS 帳戶擔任該APNPartner角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::APN-account-ID:role/APN-user-name"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

3. 客戶將角色的 Amazon Resource Name (ARN) 提供給 APN 合作夥伴。ARN 是角色的完整名稱。

```
arn:aws:iam::APN-ACCOUNT-ID:role/APNPartner
```

> **Note**
>
> 建議在多租戶情況下使用外部 ID。如需詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。

4. 當 APN 合作夥伴的軟體需要存取客戶的帳戶時，軟體會使 AWS Security Token Service 用客戶帳戶中角色的 ARN 呼叫 AssumeRoleAPI。STS 返回一個臨時 AWS 憑據，允許軟件完成其工作。

如需使用角色授予跨帳户存取權的另一個範例，請參閱 在您擁有的另一個 AWS 帳戶 IAM 使用者中提供存取權。您也可以參照 IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶。

使用資源型政策的跨帳戶存取權

帳戶使用資源型政策透過另一個帳戶存取資源時，主體仍可在受信任帳戶中運作，不需為了接受角色許可而放棄其許可。換言之，主體在存取信任帳戶中的資源時，仍可繼續存取受信任帳戶中的資源。這對於像複製資訊到其他帳戶中的共同資源，或從其複製而來的任務，非常受用。

您可以在以資源為基礎的政策中指定的主體包括帳戶、IAM 使用者、聯合身分使用者、IAM 角色、假定角色工作階段或服務。 AWS 如需詳細資訊，請參閱指定主體。

若要了解在您信任區域外帳戶 (信任組織或帳戶) 中的主體是否可擔任您的角色，請參閱識別與外部實體共用的資源。

下列清單包含一些支援以資源為基礎之政策的 AWS 服務。如需支援將權限原則附加至資源而非主參與者之 AWS 服務的完整清單，請參閱AWS 與 IAM 搭配使用的服務並尋找 [以資源為基礎] 欄中具有 [是] 的服務。

- Amazon S3 儲存貯體 – 政策連接到儲存貯體，但政策控制項同時存取儲存貯體和其中的物件。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的存取控制 。在某些情況下，最好使用角色以跨帳戶存取 Amazon S3。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的範例演練。
- Amazon Simple Notification Service (Amazon SNS) 主題 – 如需詳細資訊，請前往《Amazon Simple Notification Service 開發人員指南》中的 Amazon SNS 存取控制的範例案例。
- Amazon Simple Queue Service (Amazon SQS) 佇列 – 如需詳細資訊，請前往《Amazon Simple Queue Service 開發人員指南》中的附錄：存取原則語言。

在以資源為基 AWS 礎的政策中委派權限

如果資源將許可授予您帳戶中的主體，您可以將這些許可委派給特定的 IAM 身分。身分是您帳戶中的使用者、使用者群組或角色。您可以將政策連接至身分以委派許可。您授予的許可上限為擁有資源的帳戶所允許的最大許可。

> **Important**
>
> 在跨帳戶存取中，主體在身分型政策和資源型政策中需要 `Allow`。

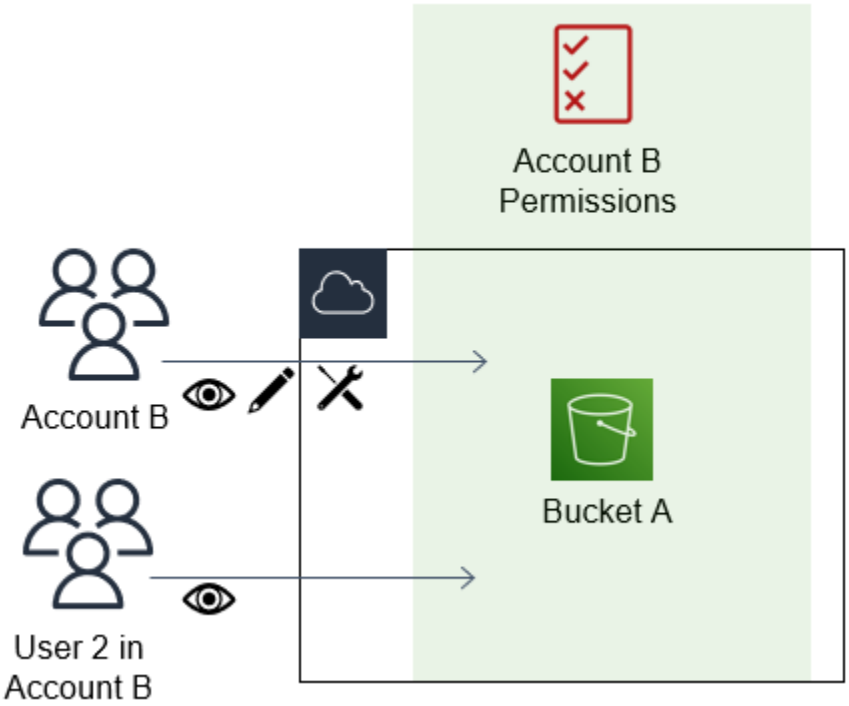
假設以資源為基礎的政策允許您帳戶中的所有主體具有資源的完整管理存取權。然後，您可以將完整存取權、唯讀存取權或任何其他部分存取權委派給您 AWS 帳戶中的主體。或者，如果以資源為基礎的政

策只允許列出許可，則您只能委派列出存取權。如果您嘗試委派比您的帳戶還多的許可，則您的主體仍然只有列出存取權。

如需有關如何做出這些決策的詳細資訊，請參閱確定帳戶內是否允許或拒絕請求。

> **Note**
>
> IAM 角色和資源型政策只會在單一分割內跨帳戶委派存取許可。例如，您無法在標準 aws 分割區的帳戶和 aws-cn 分割區的帳戶之間，新增跨帳戶存取權。

例如，假設您管理 AccountA 和 AccountB。在 AccountA 中，您具有名為 BucketA 的 Amazon S3 儲存貯體。



1. 您可以將資源型政策連接至 BucketA，以允許 AccountB 中的所有主體完整存取儲存貯體中的物件。他們可以建立、讀取或刪除該儲存貯體中的任何物件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalAccess",
            "Effect": "Allow",
```

```
            "Principal": {"AWS": "arn:aws:iam::AccountB:root"},
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::BucketA/*"
            }
        ]
}
```

透過將 AccountB 命名為資源型政策中的主體，AccountA 允許 AccountB 完整存取 BucketA。因此，AccountB 獲得授權可對 BucketA 執行任何動作，而 AccountB 管理員可以將存取權委派給它在 AccountB 的使用者。

AccountB 根使用者具有授予給帳戶的所有許可。因此，根使用者具有 BucketA 的完整存取權。

2. 在 AccountB 中，將政策連接至名為 User2 的 IAM 使用者。該政策允許使用者唯讀存取 BucketA 中的物件。這表示 User2 可以檢視物件，但無法建立、編輯或刪除物件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Action" : [
                "s3:Get*",
                "s3:List*" ],
                "Resource" : "arn:aws:s3:::BucketA/*"
        }
    ]
}
```

AccountB 可委派的最大存取層級是授予帳戶的存取層級。在此情況下，資源型政策將完整存取授予 AccountB，但只將唯讀存取授予 User2。

AccountB 管理員不提供存取權給 User1。依預設，除了明確授予的許可外，使用者不具備任何許可，因此 User1 無權存取 BucketA。

IAM 會在主體提出請求時評估主體的許可。如果您使用萬用字元 (*) 提供使用者對您資源的完整存取權，主參與者可以存取您的 AWS 帳戶可存取的任何資源。即使對於您在建立使用者政策後新增或取得存取權的資源，也是如此。

在上述範例中，如果 AccountB 已將政策連接至 User2，而此政策允許完整存取所有帳戶中的所有資源，則 User2 可自動存取 AccountB 能夠存取的任何資源。這包括 BucketA 存取，以及 AccountA 中資源型政策所授予對任何其他資源的存取。

如需有關角色的複雜使用詳細資訊，例如授予應用程式和服務存取權，請參閱 常見的角色方案：使用者、應用程式和服務。

> **Important**
>
> 只將存取權給予您信任的實體，並提供最低必要存取權。每當受信任的實體是另一個 AWS 帳戶時，都可以授予任何 IAM 主體對您資源的存取權。受信任的 AWS 帳戶只能在授予存取權限的範圍內委派存取權；它無法委派比帳戶本身所授予的更多存取權限。

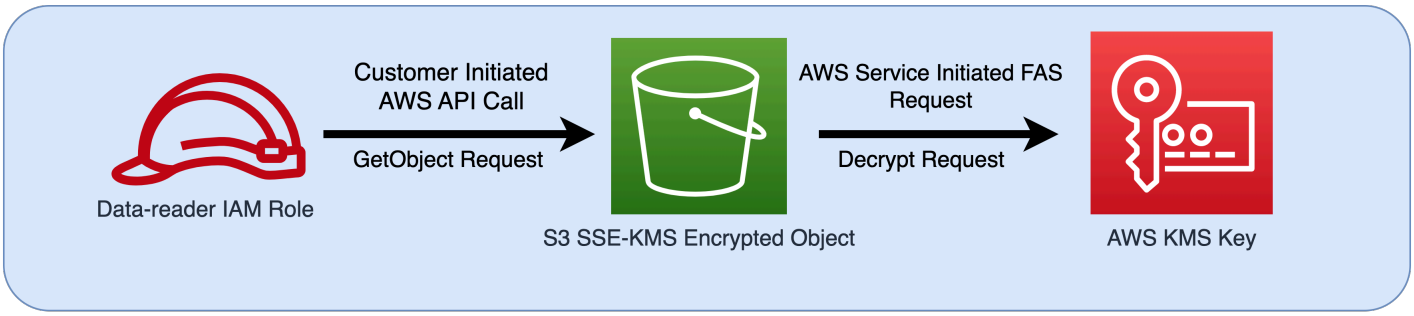
如需有關許可、政策以及用於撰寫政策的許可政策語言的詳細資訊，請參閱AWS 資源存取管理。

轉送存取工作階段

轉寄存取工作階段 (FAS) 是一種 IAM 技術， AWS 服務會在服 AWS 務代表您提出要求時，傳遞您的身分、許可和工作階段屬性。FAS 會使用呼叫 AWS 服務的身分識別權限，並結合 AWS 服務的識別，向下游服務發出要求。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，FAS 請求才會代表 IAM 主體向 AWS 服務提出。發出 FAS 請求時：

- 從 IAM 主體接收初始請求的服務會檢查 IAM 主體的許可。
- 接收後續 FAS 請求的服務也會檢查相同 IAM 主體的許可。

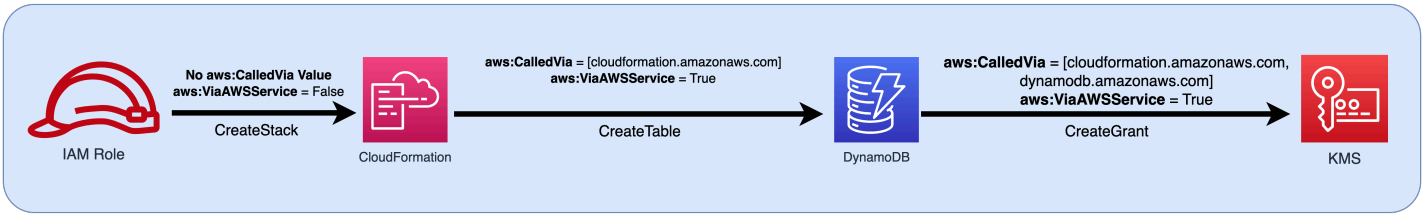
例如，當使用 SSE-KMS 加密物件時，Amazon S3 會使用 FAS AWS Key Management Service 來進行呼叫以解密物件。下載 SSE-KMS 加密物件時，名為資料讀取器的角色會針對 Amazon S3 呼叫 GetObject 物件，而不會直接呼叫。 AWS KMS 在收到 GetObject 請求並授權資料讀取器之後，Amazon S3 會向其發出 FAS 請求，以 AWS KMS 解密 Amazon S3 物件。當 KMS 收到 FAS 請求時，其會檢查角色的許可，並且只有在 data-reader 具有 KMS 金鑰的正確許可時才授權解密請求。對 Amazon S3 和 AWS KMS Amazon S3 的請求均使用角色的許可進行授權，並且只有在資料讀取器同時具有 Amazon S3 物件和 AWS KMS 金鑰的許可時，才會成功。



> **Note**
>
> 收到 FAS 請求的服務可以發出其他 FAS 請求。在這種情況下，請求主體必須具有 FAS 呼叫之所有服務的許可。

FAS 請求和 IAM 政策條件

發出 FAS 請求時，AWS：CalledVia、AWS：CalledVia第一 和 AWS：CalledVia最後 條件索引鍵會填入啟動 FAS 呼叫之服務的服務主體。每當發出 FAS 請求時，都會將 AWS: 透過 AWSService 條件索引鍵值設定為 `true`。在下圖中， CloudFormation 直接請求沒有設置任何`aws:CalledVia`或`aws:ViaAWSService`條件鍵。當 CloudFormation 和 DynamoDB 代表角色提出下游 FAS 請求時，會填入這些條件索引鍵的值。



若要允許發出 FAS 請求，以免遭具有測試來源 IP 地址或來源 VPC 之條件索引鍵的拒絕政策陳述式拒絕，您必須在您的拒絕政策中使用條件索引鍵，為 FAS 請求提供例外狀況。這可以透過使用 `aws:ViaAWSService` 條件索引鍵，為所有 FAS 請求完成此操作。若要僅允許特定 AWS 服務提出 FAS 要求，請使用`aws:CalledVia`。

> **Important**
>
> 當透過 VPC 端點發出初始請求後發出 FAS 請求時，不會在 FAS 請求中使用來自初始請求中 AWS#SourceVpce、AWS#SourceVpc 和 AWS#VpcSourceIP 的條件索引鍵值。當寫入政策使用 `aws:VPCSourceIP` 或 `aws:SourceVPCE` 有條件地授予存取權時，您也必須使用

aws:ViaAWSService 或 aws:CalledVia 以允許 FAS 請求。在公用 AWS 服務端點接收到初始要求之後發出 FAS 要求時，後續 FAS 要求將會使用相同的aws:SourceIP條件索引鍵值進行。

範例：允許從 VPC 或使用 FAS 存取 Amazon S3

在以下 IAM 政策範例中，只有 Amazon S3 GetObject 和 Athena 請求來自連接至 *example_* vpc 的 VPC 端點，或者請求是 Athena 提出的 FAS 請求時，才允許這些請求。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowMyIPs",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "athena:StartQueryExecution",
        "athena:GetQueryResults",
        "athena:GetWorkGroup",
        "athena:StopQueryExecution",
        "athena:GetQueryExecution"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceVPC": [
            "example_vpc"
          ]
        }
      }
    },
    {
      "Sid": "OnlyAllowFAS",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
```

```
          "aws:CalledVia": "athena.amazonaws.com"
        }
      }
    }
  ]
}
```

如需使用條件索引鍵允許 FAS 存取的其他範例，請參閱 data perimeter 範例政策儲存庫。

以身分為基礎的 IAM 政策範例

原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當 IAM 主體 (使用者或角色) 提出要求時，評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策都 AWS 以 JSON 文件的形式儲存在中，這些文件會附加至 IAM 身分 (使用者、使用者群組或角色)。以身分為基礎的政策包括 AWS 受管政策、客戶受管政策以及內嵌政策。若要了解如何使用這些範例 JSON 政策文件來建立 IAM 政策，請參閱 the section called “正在使用 JSON 編輯器建立政策”。

在預設情況下，所有請求會遭拒絕，所以您必須提供對要存取身分的服務、動作和資源的存取。如果您還要允許存取在 IAM 主控台中完成指定的動作，則需要提供額外的許可。

以下政策程式庫可協助您定義 IAM 身分的許可。在您找到所需的政策後，選擇 View this policy (查看此政策) 以查看 JSON 的政策。您可以將 JSON 政策文件用作自己政策的範本。

> **Note**
>
> 如果您想提交要包含在本參考指南中的政策，請使用此頁面底部的 Feedback (意見回饋) 按鈕。

政策範例： AWS

- 在特定日期範圍期間允許存取。查看此政策。
- 允許啟用和停用 AWS 區域。查看此政策。
- 允許 MFA 驗證的使用者在「安全性認證」頁面上管理自己的認證。查看此政策。
- 在特定日期範圍內使用 MFA 時允許特定存取。查看此政策。
- 允許使用者在 [安全認證] 頁面上管理自己的認證。查看此政策。
- 允許使用者在 [安全性認證] 頁面上管理自己的 MFA 裝置。查看此政策。

- 允許使用者在 [安全認證] 頁面上管理自己的密碼。查看此政策。
- 允許使用者在安全登入資料頁面上管理自己的密碼、存取金鑰和 SSH 公開金鑰。查看此政策。
- AWS 根據要求的區域拒絕存取。查看此政策。
- AWS 根據來源 IP 位址拒絕存取。查看此政策。

範例政策： AWS Data Exchange

- 拒絕存取您帳戶以外的 Amazon S3 資源，但 AWS Data Exchange除外。查看此政策。

政策範例： AWS Data Pipeline

- 拒絕存取使用者沒有建立的管道 (查看此政策。)

範例政策：Amazon DynamoDB

- 允許存取特定 Amazon DynamoDB 資料表 (查看此政策。)
- 允許存取特定 Amazon DynamoDB 屬性 (查看此政策。)
- 允許根據 Amazon Cognito ID 對 Amazon DynamoDB 進行項目層級存取 (查看此政策。)

範例政策：Amazon EC2

- 允許根據標籤將 Amazon EBS 磁碟區與 Amazon EC2 執行個體連接或分開 (查看此政策。)
- 允許以程式設計方式並在主控台的特定子網路中啟動 Amazon EC2 執行個體 (查看此政策。)
- 允許以程式設計方式及在主控台中管理與特定 VPC 相關聯的 Amazon EC2 安全群組 (查看此政策。)
- 允許以程式設計方式及在主控台中啟動或停用使用者已加上標籤的 Amazon EC2 執行個體 (查看此政策。)
- 允許以程式設計方式及在主控台中根據資源和主體標籤來啟用或停用 Amazon EC2 執行個體 (查看此政策。)
- 當資源與主體標籤相符時，允許啟用或停用 Amazon EC2 執行個體 (查看此政策。)
- 允許在特定區域內，以程式設計方式和在主控台進行 Amazon EC2 完全存取。查看此政策。
- 允許以程式設計方式和在主控台中啟動或停用特定 Amazon EC2 執行個體並修改特定安全群組 (查看此政策。)
- 拒絕在沒有 MFA 的情況下，存取特定 Amazon EC2 操作 (查看此政策。)

- 限制將 Amazon EC2 執行個體終止到特定的 IP 地址範圍 (查看此政策。)

政策範例： AWS Identity and Access Management (IAM)

- 允許存取政策模擬器 API (查看此政策。)
- 允許存取政策模擬器主控台 (查看此政策。)
- 能以程式設計方式和主控台中擔任擁有特定標籤的任何角色 (查看此政策。)
- 能以程式設計方式和主控台中允許和拒絕存取多個服務 (查看此政策。)
- 允許以程式設計方式和在主控台中使用不同的特定標籤，將特定標籤新增到 IAM 使用者 (查看此政策。)
- 允許以程式設計方式和在主控台中將特定標籤新增到任何 IAM 使用者或角色 (查看此政策。)
- 允許只使用特定標籤建立新使用者 (查看此政策。)
- 允許產生和擷取 IAM 憑證報告 (查看此政策。)
- 允許以程式設計方式及在主控台中管理群組的成員資格 (查看此政策。)
- 允許管理特定標籤 (查看此政策。)
- 允許將 IAM 角色傳遞至特定服務 (查看此政策。)
- 允許對 IAM 主控台的唯讀存取，而不需要報告 (查看此政策。)
- 允許對 IAM 主控台的唯讀存取 (查看此政策。)
- 允許特定使用者以程式設計方式及在主控台中管理群組 (查看此政策。)
- 允許以程式設計方式在主控台中設定帳戶密碼要求 (查看此政策。)
- 允許對具有特定路徑的使用者使用政策模擬器 API (查看此政策。)
- 允許對具有特定路徑的使用者使用政策模擬器主控台 (查看此政策。)
- 允許 IAM 使用者自行管理 MFA 裝置。查看此政策。
- 允許 IAM 使用者以程式設計方式和在主控台設定自己的憑證。查看此政策。
- 允許在 IAM 主控台中檢視 AWS Organizations 政策上次存取的服務資訊。查看此政策。
- 限制可以套用到新的 IAM 使用者、群組或角色的受管政策 (查看此政策。)
- 僅允許存取您帳戶中的 IAM 政策 (檢視此政策。)

政策範例： AWS Lambda

- 允許 AWS Lambda 函數存取 Amazon DynamoDB 表格 (檢視此政策)。

範例政策：Amazon RDS

- 允許在特定區域內的完整 Amazon RDS 資料庫存取。查看此政策。
- 允許以程式設計方式和在主控台中復原 Amazon RDS 資料庫 (查看此政策。)
- 允許標籤持有者擁有對已加上標籤的 Amazon RDS 資源的完整存取許可 (查看此政策)

範例政策：Amazon S3

- 允許 Amazon Cognito 使用者存取自己的 Amazon S3 儲存貯體中的物件 (查看此政策。)
- 允許聯合身分使用者以程式設計方式及在主控台中存取自己 Amazon S3 中的主目錄 (查看此政策。)
- 允許完整的 S3 存取，但如果管理員並未在最後 30 分鐘內使用 MFA 登入，則會明確拒絕存取生產儲存貯體。(查看此政策。)
- 允許 IAM 使用者以程式設計方式或在主控台中存取 Amazon S3 中自己的主目錄 (查看此政策。)
- 允許使用者管理單一 Amazon S3 儲存貯體，並拒絕其他所有 AWS 動作和資源 (檢視此政策)。
- 允許 `Read` 和 `Write` 存取特定的 Amazon S3 儲存貯體 (查看此政策。)
- 允許 `Read` 和`Write` 以程式設計方式和在主控台存取特定的 Amazon S3 儲存貯體 (查看此政策。)

AWS：允許根據日期和時間進行存取

此範例會示範如何建立身分型政策，允許根據日期和時間存取動作。此政策會限制只能在 2020 年 4 月 1 日和 2020 年 6 月 30 日 (UTC) (含) 間存取動作。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的*########*取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

若要了解使用 IAM 政策中 `Condition` 區塊內的多重條件的相關資訊，請參閱 條件中的多個值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "service-prefix:action-name",
            "Resource": "*",
            "Condition": {
                "DateGreaterThan": {"aws:CurrentTime": "2020-04-01T00:00:00Z"},
                "DateLessThan": {"aws:CurrentTime": "2020-06-30T23:59:59Z"}
            }
```

```
        }
    ]
}
```

> **Note**
>
> 您不可搭配使用政策變數與 Date 條件運算子。如需進一步了解，請參閱 條件元素

AWS：允許啟用和停用 AWS 區域

此範例會示範如何建立身分型政策，允許管理員啟用和停用亞太區域 (香港) 區域 (ap-east-1)。此政策定義了程式設計和主控台存取的許可。此設定會顯示在 AWS Management Console頁面中的 Account settings (帳戶設定)。本頁包含只有帳戶管理員能夠檢視和管理的敏感帳戶層級資訊。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

> **Important**
>
> 您無法啟用或停用預設啟用的區域。您只能包含預設停用的區域。如需詳細資訊，請參閱《AWS 一般參考》中的管理 AWS 區域。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnableDisableHongKong",
            "Effect": "Allow",
            "Action": [
                "account:EnableRegion",
                "account:DisableRegion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"account:TargetRegion": "ap-east-1"}
            }
        },
        {
            "Sid": "ViewConsole",
            "Effect": "Allow",
```

```
            "Action": [
                "account:ListRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料

此範例說明如何建立身分型政策，允許使用多重要素驗證 (MFA) 驗證的 IAM 使用者在 [安全登入資料] 頁面上管理自己的登入資料。此 AWS Management Console 頁面會顯示如帳戶 ID 和正式使用者 ID 的帳戶資訊。使用者也可以檢視和編輯其密碼、存取金鑰、MFA 裝置、X.509 憑證及 SSH 金鑰和 Git 憑證。此政策範例包括檢視和編輯頁面上所有資訊所需的許可。它還要求使用者在中 AWS執行任何其他作業之前，先使用 MFA 進行設定和驗證。若要允許使用者在不使用 MFA 的情況下管理自己的憑證，請參閱AWS：允許 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

若要瞭解使用者如何存取安全認證頁面，請參閱IAM 使用者如何變更他們自己的密碼 (主控台)。

> **Note**
>
> - 此範例原則不允許使用者在第一次登入時重設密碼。 AWS Management Console 在新使用者登入之前，建議您不要將許可授予給他們。如需詳細資訊，請參閱 如何安全地建立 IAM 使用者？。這也可防止密碼到期的使用者在登入期間重設密碼。您可以透過將 `iam:ChangePassword` 和 `iam:GetAccountPasswordPolicy` 新增至 `DenyAllExceptListedIfNoMFA` 陳述式，來允許這項作業。不過，我們不建議您這麼做，因為若允許使用者在不使用 MFA 的情況下變更密碼，則會帶來安全性風險。
> - 如果您打算使用此政策進行程式設計存取，則必須叫用 `GetSessionToken` 以使用 MFA 進行驗證。如需詳細資訊，請參閱 設定受 MFA 保護的 API 存取。

此政策的功能為何？

- `AllowViewAccountInfo` 陳述式允許使用者檢視帳戶層級資訊。這些許可必須位於自己的陳述式中，因為它們不支援或不需要指定資源 ARN。而是許可指定 `"Resource" : "*"`。此陳述式包括下列動作，可讓使用者檢視特定的資訊：
  - `GetAccountPasswordPolicy` – 檢視帳戶密碼需求，同時變更自己的 IAM 使用者密碼。
  - `ListVirtualMFADevices` – 檢視為使用者啟用之虛擬 MFA 裝置的詳細資訊。

- `AllowManageOwnPasswords` 陳述式可讓使用者變更自己的密碼。此陳述式也包括檢視 My Security Credentials (我的安全憑證) 頁面上大部分資訊所需的 `GetUser` 動作。
- `AllowManageOwnAccessKeys` 陳述式可讓使用者建立、更新及刪除自己的存取金鑰。使用者也能擷取關於指定之存取金鑰最後一次使用時間的資訊。
- `AllowManageOwnSigningCertificates` 陳述式可讓使用者上傳、更新及刪除自己的簽章憑證。
- 該`AllowManageOwnSSHPublicKeys`語句允許用戶上傳，更新和刪除自己的 SSH 公鑰 CodeCommit。
- 該`AllowManageOwnGitCredentials`語句允許用戶創建，更新和刪除自己的 Git 憑據 CodeCommit。
- `AllowManageOwnVirtualMFADevice` 陳述式可讓使用者建立自己的虛擬 MFA 裝置。此陳述式中的資源 ARN 允許使用者以任何名稱建立 MFA 裝置，但政策中的其他陳述式僅允許使用者連接裝置到目前登入的使用者。
- `AllowManageOwnUserMFA` 陳述式可讓使用者檢視或管理其使用者的虛擬、U2F 或硬體 MFA 裝置。此陳述式中的資源 ARN 僅允許存取使用者自己的 IAM 使用者。使用者不能檢視或管理其他使用者的 MFA 裝置。
- 該`DenyAllExceptListedIfNoMFA`聲明拒絕訪問所有 AWS 服務中的每個操作，除了一些列出的操作，但前提是用戶未使用 MFA 登錄。陳述式使用 "Deny" 和 "NotAction" 的組合，來明確拒絕存取未列出的每項動作。此陳述式不會拒絕或允許列出的項目。然而，政策中的其他陳述式會允許多項動作。如需有關此陳述式邏輯的詳細資訊，請參閱「拒絕」(NotAction Deny)。如果使用者使用 MFA 登入，則不會通過 `Condition` 測試，且此陳述式不會拒絕任何動作。在此情況下，使用者的其他政策或陳述式決定使用者的許可。

  此陳述式確保在使用者未使用 MFA 登入時，他們只能執行列出的動作。此外，只有在另一條陳述式或政策允許存取這些動作時，他們才能執行列出的動作。這不允許使用者在登入時建立密碼，因為在未進行 MFA 身分驗證的情況下，並不允許 `iam:ChangePassword` 動作。

  `...IfExists` 運算子的 `Bool` 版本會確認，如果 `AWS#MultiFactorAuthPresent` 索引鍵遺失，條件將返回 true。這就表示，拒絕使用長期憑證 (如存取金鑰) 存取 API 的使用者存取為非 IAM API 操作。

此政策不允許使用者檢視 IAM 主控台中的 Users (使用者) 頁面，或使用該頁面存取自己的使用者資訊。若要允許此操作，請將 `iam:ListUsers` 動作加入到 `AllowViewAccountInfo` 陳述式和 `DenyAllExceptListedIfNoMFA` 陳述式。它也不允許使用者在自己的使用者頁面上變更密碼。若要允許此操作，請將 `iam:GetLoginProfile` 及 `iam:UpdateLoginProfile`

動作加入 AllowManageOwnPasswords 陳述式。若也要允許使用者在未使用 MFA 登入的情況下，在自己的使用者頁面上變更其密碼，請將 iam:UpdateLoginProfile 動作加入 DenyAllExceptListedIfNoMFA 陳述式。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSigningCertificates",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSigningCertificate",
                "iam:ListSigningCertificates",
```

```
            "iam:UpdateSigningCertificate",
            "iam:UploadSigningCertificate"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnSSHPublicKeys",
        "Effect": "Allow",
        "Action": [
            "iam:DeleteSSHPublicKey",
            "iam:GetSSHPublicKey",
            "iam:ListSSHPublicKeys",
            "iam:UpdateSSHPublicKey",
            "iam:UploadSSHPublicKey"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnGitCredentials",
        "Effect": "Allow",
        "Action": [
            "iam:CreateServiceSpecificCredential",
            "iam:DeleteServiceSpecificCredential",
            "iam:ListServiceSpecificCredentials",
            "iam:ResetServiceSpecificCredential",
            "iam:UpdateServiceSpecificCredential"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnVirtualMFADevice",
        "Effect": "Allow",
        "Action": [
            "iam:CreateVirtualMFADevice"
        ],
        "Resource": "arn:aws:iam::*:mfa/*"
    },
    {
        "Sid": "AllowManageOwnUserMFA",
        "Effect": "Allow",
        "Action": [
            "iam:DeactivateMFADevice",
            "iam:EnableMFADevice",
            "iam:ListMFADevices",
```

```
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

AWS：允許在特定日期內使用 MFA 進行特定存取

此範例會示範如何建立身分型政策，以使用多個條件，且這些條件使用邏輯 AND 進行評估。它允許完整存取名為 SERVICE-NAME-1 的服務，以及存取名為 ACTION-NAME-A 服務內的 ACTION-NAME-B 和 SERVICE-NAME-2 動作。這些動作只有在使用多重驗證 (MFA) 驗證使用者時才獲允許進行。存取會限制於在 2017 年 7 月 1 日到 2017 年 12 月 31 日 (UTC)(包含在內) 之間發生的動作。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

若要了解使用 IAM 政策中 Condition 區塊內的多重條件的相關資訊，請參閱 條件中的多個值。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Action": [
            "service-prefix-1:*",
            "service-prefix-2:action-name-a",
            "service-prefix-2:action-name-b"
        ],
        "Resource": "*",
        "Condition": {
            "Bool": {"aws:MultiFactorAuthPresent": true},
            "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
            "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
        }
    }
}
```

AWS：允許 IAM 使用者在安全登入資料頁面上管理自己的登入資料

此範例顯示如何建立身分型政策，讓 IAM 使用者在「安全登入資料」頁面上管理自己的所有登入資料。此 AWS Management Console 頁面顯示帳戶資訊，例如帳號 ID 和規範使用者 ID。使用者也可以檢視和編輯其密碼、存取金鑰、X.509 憑證、SSH 金鑰和 Git 憑證。除了使用者的 MFA 裝置以外，此政策範例包括檢視和編輯頁面上所有資訊所需的許可。若要允許使用者使用 MFA 管理自己的憑證，請參閱AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

若要瞭解使用者如何存取安全認證頁面，請參閱IAM 使用者如何變更他們自己的密碼 (主控台)。

此政策的功能為何？

- `AllowViewAccountInfo` 陳述式允許使用者檢視帳戶層級資訊。這些許可必須位於自己的陳述式中，因為它們不支援或不需要指定資源 ARN。而是許可指定 `"Resource" : "*"`。此陳述式包括下列動作，可讓使用者檢視特定的資訊：
  - `GetAccountPasswordPolicy` – 檢視帳戶密碼需求，同時變更自己的 IAM 使用者密碼。
  - `GetAccountSummary` – 檢視帳戶 ID 和帳戶正式使用者 ID。
- `AllowManageOwnPasswords` 陳述式可讓使用者變更自己的密碼。此陳述式也包括檢視 My Security Credentials (我的安全憑證) 頁面上大部分資訊所需的 `GetUser` 動作。
- `AllowManageOwnAccessKeys` 陳述式可讓使用者建立、更新及刪除自己的存取金鑰。使用者也能擷取關於指定之存取金鑰最後一次使用時間的資訊。
- `AllowManageOwnSigningCertificates` 陳述式可讓使用者上傳、更新及刪除自己的簽章憑證。
- 該`AllowManageOwnSSHPublicKeys`語句允許用戶上傳，更新和刪除自己的 SSH 公鑰 CodeCommit。

- 該AllowManageOwnGitCredentials語句使用戶能夠創建，更新和刪除自己的 Git 憑據 CodeCommit。

此政策不允許使用者檢視或管理自己的 MFA 裝置。他們也不能檢視 IAM 主控台中的 Users (使用者) 頁面，或使用該頁面存取自己的使用者資訊。若要允許此操作，請將 iam:ListUsers 動作加入 AllowViewAccountInfo 陳述式。它也不允許使用者在自己的使用者頁面上變更密碼。若要允許此操作，請將 iam:CreateLoginProfile、iam:DeleteLoginProfile、iam:GetLoginProfile 及 iam:UpdateLoginProfile 動作加入 AllowManageOwnPasswords 陳述式。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
```

```
        },
        {
            "Sid": "AllowManageOwnSigningCertificates",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSigningCertificate",
                "iam:ListSigningCertificates",
                "iam:UpdateSigningCertificate",
                "iam:UploadSigningCertificate"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnGitCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceSpecificCredential",
                "iam:DeleteServiceSpecificCredential",
                "iam:ListServiceSpecificCredentials",
                "iam:ResetServiceSpecificCredential",
                "iam:UpdateServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的 MFA 裝置

此範例說明如何建立身分型政策，允許透過多重要素驗證 (MFA) 驗證的 IAM 使用者在 [安全登入資料] 頁面上管理自己的 MFA 裝置。此 AWS Management Console 頁面顯示帳戶和使用者資訊，但使用者

只能檢視和編輯自己的 MFA 裝置。若要允許使用者使用 MFA 管理自己的所有憑證，請參閱AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

> **Note**
>
> 如果具有此政策的 IAM 使用者未經過 MFA 驗證，則此政策會拒絕存取所有動作，但使用 MFA 進行驗證所需的 AWS 動作除外。若要使用 AWS CLI 和 AWS API，IAM 使用者必須先使用 AWS STS GetSessionToken作業擷取其 MFA 權杖，然後使用該權杖來驗證所需的作業。其他政策 (例如資源型政策或其他身分型政策) 可允許其他服務中的動作。如果 IAM 使用者並未通過 MFA 身分驗證，此政策將會拒絕該存取。

若要瞭解使用者如何存取安全認證頁面，請參閱IAM 使用者如何變更他們自己的密碼 (主控台)。

此政策的功能為何？

- `AllowViewAccountInfo` 陳述式可讓使用者檢視為使用者啟用之虛擬 MFA 裝置的詳細資訊。由於此許可不支援指定資源 ARN，因此此許可必須在其陳述式中。您反而必須指定 `"Resource" : "*"`。
- `AllowManageOwnVirtualMFADevice` 陳述式可讓使用者建立自己的虛擬 MFA 裝置。此陳述式中的資源 ARN 允許使用者以任何名稱建立 MFA 裝置，但政策中的其他陳述式僅允許使用者連接裝置到目前登入的使用者。
- `AllowManageOwnUserMFA` 陳述式可讓使用者檢視或管理自己的虛擬、U2F 或硬體 MFA 裝置。此陳述式中的資源 ARN 僅允許存取使用者自己的 IAM 使用者。使用者不能檢視或管理其他使用者的 MFA 裝置。
- 該`DenyAllExceptListedIfNoMFA`聲明拒絕訪問所有 AWS 服務中的每個操作，除了一些列出的操作，但前提是用戶未使用 MFA 登錄。陳述式使用 `"Deny"` 和 `"NotAction"` 的組合，來明確拒絕存取未列出的每項動作。此陳述式不會拒絕或允許列出的項目。然而，政策中的其他陳述式會允許多項動作。如需有關此陳述式邏輯的詳細資訊，請參閱「拒絕」(NotAction Deny)。如果使用者使用 MFA 登入，則不會通過 `Condition` 測試，且此陳述式不會拒絕任何動作。在此情況下，使用者的其他政策或陳述式決定使用者的許可。

  此陳述式確保在使用者未使用 MFA 登入時，他們只能執行列出的動作。此外，只有在另一條陳述式或政策允許存取這些動作時，他們才能執行列出的動作。

  `...IfExists` 運算子的 Bool 版本會確認，如果 `aws:MultiFactorAuthPresent` 索引鍵遺失，條件將返回 true。這就表示，拒絕使用長期憑證 (如存取金鑰) 存取 API 操作的使用者存取為非 IAM API 操作。

此政策不允許使用者檢視 IAM 主控台中的 Users (使用者) 頁面，或使用該頁面存取自己的使用者資訊。若要允許此操作，請將 `iam:ListUsers` 動作加入到 `AllowViewAccountInfo` 陳述式和 `DenyAllExceptListedIfNoMFA` 陳述式。

> **Warning**
>
> 在沒通過 MFA 身分驗證的情況下不新增刪除 MFA 裝置的許可。使用此政策的使用者可能會嘗試為本身指派虛擬 MFA 裝置，但收到未獲授權執行 `iam:DeleteVirtualMFADevice` 的錯誤。如果發生這種情況，請勿將該許可新增到 `DenyAllExceptListedIfNoMFA` 陳述式。未通過 MFA 身分驗證的使用者一律不得刪除其 MFA 裝置。如果使用者先前開始將虛擬 MFA 裝置指派到其使用者並取消該程序，使用者可能就會看到此錯誤。若要解決此問題，您或其他系統管理員必須使用 AWS CLI 或 AWS API 刪除使用者現有的虛擬 MFA 裝置。如需更多詳細資訊，請參閱 我沒有授權執行：IAM：DeleteVirtualMFA 設備。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": "iam:ListVirtualMFADevices",
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
```

```
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}
            }
        }
    ]
}
```

AWS：允許 IAM 使用者在安全登入資料頁面上變更自己的主控台密碼

此範例顯示如何建立身分型政策，讓 IAM 使用者在安全登入資料頁面上變更自己的 AWS Management Console 密碼。此 AWS Management Console 頁面顯示帳戶和使用者資訊，但使用者只能存取自己的密碼。若要允許使用者使用 MFA 管理自己的所有憑證，請參閱AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。若要允許使用者在不使用 MFA 的情況下管理自己的憑證，請參閱AWS：允許 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

若要瞭解使用者如何存取安全認證頁面，請參閱IAM 使用者如何變更他們自己的密碼 (主控台)。

此政策的功能為何？

- `ViewAccountPasswordRequirements` 陳述式可讓使用者檢視帳戶密碼需求，同時變更自己的 IAM 使用者密碼。
- `ChangeOwnPassword` 陳述式可讓使用者變更自己的密碼。此陳述式也包括檢視 My Security Credentials (我的安全憑證) 頁面上大部分資訊所需的 `GetUser` 動作。

此政策不允許使用者檢視 IAM 主控台中的 Users (使用者) 頁面，或使用該頁面存取自己的使用者資訊。若要允許此操作，請將 `iam:ListUsers` 動作加入 `ViewAccountPasswordRequirements` 陳述式。它也不允許使用者在自己的使用者頁面上變更密碼。若要允許此操作，請將 `iam:GetLoginProfile` 及 `iam:UpdateLoginProfile` 動作加入 `ChangeOwnPasswords` 陳述式。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewAccountPasswordRequirements",
            "Effect": "Allow",
            "Action": "iam:GetAccountPasswordPolicy",
            "Resource": "*"
        },
        {
            "Sid": "ChangeOwnPassword",
            "Effect": "Allow",
            "Action": [
                "iam:GetUser",
                "iam:ChangePassword"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

AWS：允許 IAM 使用者在安全登入資料頁面上管理自己的密碼、存取金鑰和 SSH 公開金鑰

此範例顯示如何建立身分型政策，讓 IAM 使用者在安全登入資料頁面上管理自己的密碼、存取金鑰和 X.509 憑證。此 AWS Management Console 頁面會顯示如帳戶 ID 和正式使用者 ID 的帳戶資訊。使用者也可以檢視和編輯其密碼、存取金鑰、MFA 裝置、X.509 憑證、SSH 金鑰和 Git 憑證。此範例政策包括僅檢視和編輯其密碼、存取金鑰和 X.509 憑證所需的許可。若要允許使用者使用 MFA 管理自己的所有憑證，請參閱AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。若要允許使用者在不使用 MFA 的情況下管理自己的憑證，請參閱AWS：允許 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

若要瞭解使用者如何存取安全認證頁面，請參閱IAM 使用者如何變更他們自己的密碼 (主控台)。

此政策的功能為何？

- AllowViewAccountInfo 陳述式允許使用者檢視帳戶層級資訊。這些許可必須位於自己的陳述式中，因為它們不支援或不需要指定資源 ARN。而是許可指定 "Resource" : "*"。此陳述式包括下列動作，可讓使用者檢視特定的資訊：
  - GetAccountPasswordPolicy – 檢視帳戶密碼需求，同時變更自己的 IAM 使用者密碼。
  - GetAccountSummary – 檢視帳戶 ID 和帳戶正式使用者 ID。
- AllowManageOwnPasswords 陳述式可讓使用者變更自己的密碼。此陳述式也包括檢視 My Security Credentials (我的安全憑證) 頁面上大部分資訊所需的 GetUser 動作。
- AllowManageOwnAccessKeys 陳述式可讓使用者建立、更新及刪除自己的存取金鑰。使用者也能擷取關於指定之存取金鑰最後一次使用時間的資訊。
- 該AllowManageOwnSSHPublicKeys語句允許用戶上傳，更新和刪除自己的 SSH 公鑰 CodeCommit。

此政策不允許使用者檢視或管理自己的 MFA 裝置。他們也不能檢視 IAM 主控台中的 Users (使用者) 頁面，或使用該頁面存取自己的使用者資訊。若要允許此操作，請將 iam:ListUsers 動作加入 AllowViewAccountInfo 陳述式。它也不允許使用者在自己的使用者頁面上變更密碼。若要允許此操作，請將 iam:GetLoginProfile 及 iam:UpdateLoginProfile 動作加入 AllowManageOwnPasswords 陳述式。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
```

```
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

AWS: AWS 根據請求的區域拒絕訪問

此範例會示範如何建立身分型政策，拒絕以 aws:RequestedRegion 條件索引鍵所指定區域以外位置任何操作的存取權，但使用 NotAction 所指定服務的操作除外。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

此政策使用含 Deny 效果的 NotAction 元素，此效果會明確拒絕存取「未」列在陳述式中的所有動作。不應拒絕 IAM CloudFront、Route 53 和 AWS Support 服務中的動作，因為這些是具有實際位於us-east-1區域中的單一端點的熱門 AWS 全球服務。這些服務的所有請求是對 us-east-1 區域提出，因此在沒有使用 NotAction 元素的情況下請求會遭拒。編輯此元素來為您使用的其他 AWS 全域服務 (例如，budgets、globalaccelerator、importexport、organizations 或 waf) 包含動作。其他一些全球服務，例如 AWS Chatbot 和 AWS Device Farm，是具有實際位於該us-west-2地區的端點的全球服務。要了解具有單一全域端點的所有服務，請參閱 AWS 一般參考 中的

AWS 區域和端點。如需有關使用含 NotAction 效果之 Deny 元素的詳細資訊，請參閱 IAM JSON 政策元素：NotAction。

> **Important**
>
> 此政策不允許任何動作。將此政策與允許特定動作的其他政策結合使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideRequestedRegions",
            "Effect": "Deny",
            "NotAction": [
                "cloudfront:*",
                "iam:*",
                "route53:*",
                "support:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

AWS： AWS 根據來源 IP 拒絕存取

此範例顯示如何建立以身分識別為基礎的原則，當要求來自指定 IP 範圍之外的主參與者時，拒絕存取帳戶中的所有 AWS 動作。當您公司的 IP 地址在指定範圍內時，該政策很有用。在此範例中，請求需要源自 CIDR 範圍 192.0.2.0/24 或 203.0.113.0/24 ，否則會遭到拒絕。此原則不會拒絕 AWS 服務使用的要求，轉送存取工作階段因為原始要求者的 IP 位址會保留。

請小心使用相同政策陳述式中的負面條件做為 "Effect": "Deny"。當您這麼做時，除了指定的委託人之外，政策陳述式中指定的動作在所有條件下都會明確拒絕授予。

> **Important**
>
> 此政策不允許任何動作。將此政策與允許特定動作的其他政策結合使用。

當其他政策允許動作時，主體可以從 IP 地址範圍內提出請求。 AWS 服務也可以使用主體的認證提出要求。當主體從 IP 範圍外提出請求時，即會拒絕該請求。

如需有關使用 aws:SourceIp 條件鍵 (包括何時 aws:SourceIp 可能無法在政策中運作) 的詳細資訊，請參閱 AWS 全域條件內容索引鍵。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            }
        }
    }
}
```

AWS：拒絕存取您帳戶外部的 Amazon S3 資源，除非 AWS Data Exchange

此範例顯示如何建立以身分識別為基礎的原則，拒絕存取不屬於您帳戶 AWS 的所有資源，但正常作業所 AWS Data Exchange 需的資源除外。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

您可以建立類似的策略來限制對組織或組織單位內資源的存取，同時使用條件鍵aws:ResourceOrgPaths和來計算 AWS Data Exchange 擁有的資源aws:ResourceOrgID。

如果您 AWS Data Exchange 在環境中使用，服務會建立服務帳戶所擁有的 Amazon S3 儲存貯體等資源並與之互動。例如，代表叫用 AWS Data Exchange API 的 IAM 主體 (使用者或角色)， AWS Data

Exchange 將請求傳送到 AWS Data Exchange 服務擁有的 Amazon S3 儲存貯體。在這種情況下，在政策aws:ResourceOrgID中使用aws:ResourceAccountaws:ResourceOrgPaths、或而不計算 AWS Data Exchange 擁有資源的情況下，會拒絕存取服務帳戶所擁有的值區。

- 陳述式 DenyAllAwsResourcesOutsideAccountExceptS3 會搭配使用 NotAction 元素與拒絕效果，明確拒絕存取陳述式中未列出的且不屬於所列帳戶的每個動作。NotAction 元素表示此陳述式的例外狀況。這些動作是此陳述式的例外狀況，因為如果動作是針對由建立的資源執行 AWS Data Exchange，則原則會拒絕這些動作。
- DenyAllS3ResoucesOutsideAccountExceptDataExchange 陳述式會結合使用 ResourceAccount 和 CalledVia 條件，拒絕存取先前的陳述式中排除的 Amazon S3 動作。如果資源不屬於列出的帳戶並且如果呼叫服務不是 AWS Data Exchange，陳述式會拒絕執行動作。如果資源屬於列出的帳戶，或者列出的服務主體 dataexchange.amazonaws.com 執行操作，則陳述式不會拒絕執行動作。

> **Important**
>
> 此政策不允許任何動作。它使用 Deny 效果，明確拒絕存取陳述式中列出的不屬於所列帳戶的所有資源。將此政策與允許存取特定資源的其他政策結合使用。

以下範例示範如何設定政策，以允許存取所需的 Amazon S3 儲存貯體。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllAwsReourcesOutsideAccountExceptAmazonS3",
      "Effect": "Deny",
      "NotAction": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
```

```
        }
      }
    },
    {
      "Sid": "DenyAllS3ResourcesOutsideAccountExceptDataExchange",
      "Effect": "Deny",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        },
        "ForAllValues:StringNotEquals": {
          "aws:CalledVia": [
            "dataexchange.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

AWS Data Pipeline：拒絕存取使用者未建立的 DataPipeline 管道

此範例會示範如何建立身分型政策，拒絕存取使用者未建立的管道。如果 `PipelineCreator` 欄位的值符合 IAM 使用者名稱，則不會拒絕指定的動作。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。

> **Important**
>
> 此政策不允許任何動作。將此政策與允許特定動作的其他政策結合使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "ExplicitDenyIfNotTheOwner",
            "Effect": "Deny",
            "Action": [
                "datapipeline:ActivatePipeline",
                "datapipeline:AddTags",
                "datapipeline:DeactivatePipeline",
                "datapipeline:DeletePipeline",
                "datapipeline:DescribeObjects",
                "datapipeline:EvaluateExpression",
                "datapipeline:GetPipelineDefinition",
                "datapipeline:PollForTask",
                "datapipeline:PutPipelineDefinition",
                "datapipeline:QueryObjects",
                "datapipeline:RemoveTags",
                "datapipeline:ReportTaskProgress",
                "datapipeline:ReportTaskRunnerHeartbeat",
                "datapipeline:SetStatus",
                "datapipeline:SetTaskStatus",
                "datapipeline:ValidatePipelineDefinition"
            ],
            "Resource": ["*"],
            "Condition": {
                "StringNotEquals": {"datapipeline:PipelineCreator": "${aws:userid}"}
            }
        }
    ]
}
```

Amazon DynamoDB：允許存取特定資料表

此範例會示範如何建立身分型政策，允許完整存取 MyTable DynamoDB 資料表。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

> **Important**
>
> 此政策允許可以在 DynamoDB 資料表上執行的所有動作。若要查看這些動作的詳細資訊，請參閱《Amazon DynamoDB 開發人員指南》中的 DynamoDB API 許可：動作、資源和條件參考。您可以透過列出每個個別的動作來提供相同的許可。不過，如果您在 Action 元素使用萬

用字元 (*) (例如 "dynamodb:List*")，則在 DynamoDB 新增新的清單動作時，您就不需要更新您的政策。

此政策僅允許對具有指定名稱的 DynamoDB 資料表執行動作。若要允許使用者Read存取 DynamoDB 中的所有內容，您也可以附加AmazonDynamo資料庫ReadOnlyAccess AWS 管理的原則。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAndDescribe",
            "Effect": "Allow",
            "Action": [
                "dynamodb:List*",
                "dynamodb:DescribeReservedCapacity*",
                "dynamodb:DescribeLimits",
                "dynamodb:DescribeTimeToLive"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGet*",
                "dynamodb:DescribeStream",
                "dynamodb:DescribeTable",
                "dynamodb:Get*",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWrite*",
                "dynamodb:CreateTable",
                "dynamodb:Delete*",
                "dynamodb:Update*",
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
        }
    ]
}
```

Amazon DynamoDB：允許存取特定屬性

此範例會示範如何建立身身分型政策，允許存取特定 DynamoDB 屬性。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

dynamodb:Select 要求防止 API 動作傳回任何不允許的屬性，例如來自索引投影的屬性。若要進一步了解 DynamoDB 條件金鑰，請參閱《Amazon DynamoDB 開發人員指南》中的指定條件：使用條件金鑰。若要了解使用 IAM 政策中的 Condition 區塊內之多重條件或多重條件索引鍵的相關資訊，請參閱 條件中的多個值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem"
            ],
            "Resource": ["arn:aws:dynamodb:*:*:table/table-name"],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:Attributes": [
                        "column-name-1",
                        "column-name-2",
                        "column-name-3"
                    ]
                },
                "StringEqualsIfExists": {"dynamodb:Select": "SPECIFIC_ATTRIBUTES"}
            }
        }
    ]
}
```

Amazon DynamoDB：允許根據 Amazon Cognito ID 對 DynamoDB 進行項目層級存取

此範例會示範如何建立身分型政策，允許根據 Amazon Cognito 身分池使用者 ID 對 MyTable DynamoDB 資料表進行項目層級存取。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

若要使用此政策，您必須建構 DynamoDB 資料表，使 Amazon Cognito 身分池使用者 ID 是分割區索引鍵。如需詳細資訊，請參閱《Amazon DynamoDB 開發人員指南》中的建立資料表。

若要進一步了解 DynamoDB 條件金鑰，請參閱《Amazon DynamoDB 開發人員指南》中的指定條件：使用條件金鑰。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": ["arn:aws:dynamodb:*:*:table/MyTable"],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]
                }
            }
        }
    ]
}
```

Amazon EC2：根據標籤將 Amazon EBS 磁碟區與 EC2 執行個體連接或分開

此範例會示範如何建立身分型政策，允許 EBS 磁碟區擁有者將使用標籤 VolumeUser 定義的 EBS 磁碟區與標記為開發執行個體的 EC2 執行個體 (Department=Development) 連接或分開。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

如需建立 IAM 政策以控制 Amazon EC2 資源存取權的詳細資訊，請參閱 Amazon EC2 使用者指南中的《控制對 Amazon EC2 資源的存取權限》。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Department": "Development"}
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:*:*:volume/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/VolumeUser": "${aws:username}"}
            }
        }
    ]
}
```

Amazon EC2：允許以程式設計方式在特定子網路中，以及主控台中啟動 EC2 執行個體

此範例會示範如何建立身分型政策，允許列出所有 EC2 物件的資訊並在特定子網路中啟動 EC2 執行個體。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:Describe*",
                "ec2:GetConsole*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/subnet-subnet-id",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*::image/ami-*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        }
    ]
}
```

Amazon EC2：允許以程式設計方式和在主控台中管理具備特定標籤鍵值對的 EC2 安全群組

此範例顯示如何建立身分型政策，授予使用者對具有相同標籤的安全群組採取特定動作的許可。此政策會授予許可以在 Amazon EC2 主控台中檢視安全群組，並針對具有 Department=Test 標籤的現有安全群組新增和移除傳入與傳出規則，以及列出和修改規則說明。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
         "ec2:DescribeSecurityGroups",
         "ec2:DescribeSecurityGroupRules",
         "ec2:DescribeTags"
      ],
      "Resource": "*"
    },
```

```
    {
      "Effect": "Allow",
      "Action": [
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:RevokeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:ModifySecurityGroupRules",
         "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
         "ec2:UpdateSecurityGroupRuleDescriptionsEgress"
      ],
      "Resource": [
         "arn:aws:ec2:region:111122223333:security-group/*"
      ],
      "Condition": {
         "StringEquals": {
            "aws:ResourceTag/Department": "Test"
         }
      }
     },
     {
      "Effect": "Allow",
      "Action": [
         "ec2:ModifySecurityGroupRules"
      ],
      "Resource": [
         "arn:aws:ec2:region:111122223333:security-group-rule/*"
      ]
     }
   ]
}
```

Amazon EC2：允許以程式設計方式和在主控台中開始或停止使用者已標記的 EC2 執行個體

此範例會示範如何建立身分型政策，允許 IAM 使用者啟動或停止 EC2 執行個體，但前提是執行個體標籤 Owner 的值是該使用者的使用者名稱。此政策定義了程式設計和主控台存取的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

EC2：依據標籤啟動或停止執行個體

此範例會示範如何建立身分型政策，允許使用標籤鍵值對 `Project = DataAnalytics` 來啟動或停止執行個體，但僅可由具有標籤鍵值對 `Department = Data` 的主體進行。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的######取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

如果條件的兩個部分都傳回 true，政策的條件就會傳回 true。執行個體必須擁有 `Project=DataAnalytics` 標籤。此外，發出請求的 IAM 主體 (使用者或角色) 都必須擁有 `Department=Data` 標籤。

> **Note**
>
> 對於某些使用者可能有而某些使用者可能沒有指定的標籤的案例而言，最佳實務是將具有 `aws:PrincipalTag` 條件索引鍵的政策連接至 IAM 群組。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "StartStopIfTags",
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:region:account-id:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "DataAnalytics",
                    "aws:PrincipalTag/Department": "Data"
                }
            }
        }
    ]
}
```

EC2：依據比對主體和資源的標籤啟動或停止執行個體

此範例會示範如何建立身分型政策，允許主體在執行個體的資源標籤與主體的標籤之標籤鍵 `CostCenter` 值相同時，啟動或停止 Amazon EC2 執行個體。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

> **Note**
>
> 對於某些使用者可能有而某些使用者可能沒有指定的標籤的案例而言，最佳實務是將具有 `aws:PrincipalTag` 條件索引鍵的政策連接至 IAM 群組。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "ec2:startInstances",
            "ec2:stopInstances"
        ],
        "Resource": "*",
        "Condition": {"StringEquals":
            {"aws:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"}}
    }
```

```
}
```

Amazon EC2：允許在特定區域內，以程式設計方式和在主控台進行 EC2 完全存取

此範例會示範如何建立身分型政策，允許在特定區域內完整存取 EC2。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。如需區域代碼的清單，請參閱《Amazon EC2 使用者指南》中的可用區域。

或者，您可以使用全域條件金鑰 aws:RequestedRegion，它受所有 Amazon EC2 API 動作支援。如需詳細資訊，請參閱《Amazon EC2 使用者指南》中的範例：限制特定區域的存取。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "ec2:*",
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "us-east-2"
                }
            }
        }
    ]
}
```

Amazon EC2：允許以程式設計方式和在主控台中啟動或停用特 EC2 執行個體並修改安全群組

此範例會示範如何建立身分型政策，允許啟動或停止特定的 EC2 執行個體並修改特定的安全群組。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
```

```
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeStaleSecurityGroups"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/i-instance-id",
        "arn:aws:ec2:*:*:security-group/sg-security-group-id"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Amazon EC2：特定 EC2 操作需要 MFA (GetSessionToken)

此範例說明如何建立以身分識別為基礎的政策，以便完整存取 Amazon EC2 中的所有 AWS API 操作。不過，如果未透過多重要素驗證 (MFA) 來對使用者進行驗證，此政策會明確拒絕對 StopInstances 和 TerminateInstances API 操作的存取。若要以程式設計的方式執行，使用者必須在呼叫 SerialNumber 操作時包含選用 TokenCode 和 GetSessionToken 值。這個操作會傳回透過使用 MFA 驗證的臨時憑證。若要進一步瞭解 GetSessionToken，請參閱— GetSessionToken 不信任環境中使用者的暫時憑證。

此政策的功能為何？

- AllowAllActionsForEC2 陳述式會允許所有 Amazon EC2 動作。
- MFA 內容遺失時，DenyStopAndTerminateWhenMFAIsNotPresent 陳述式會拒絕 StopInstances 和 TerminateInstances 動作。這表示當多重驗證內容遺失 (表示未使用 MFA) 時，動作遭拒。拒絕會覆寫允許。

> **Note**
>
> MultiFactorAuthPresent 陳述式中 Deny 的條件檢查不應該是 {"Bool": {"aws:MultiFactorAuthPresent":false}}，因為該索引鍵不存在，並且在不使用 MFA 時無法評估。因此，在檢查值之前，使用 BoolIfExists 檢查來查看索引鍵是否存在。如需更多詳細資訊，請參閱 ... IfExists 條件運算子。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllActionsForEC2",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
            "Effect": "Deny",
            "Action": [
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {"aws:MultiFactorAuthPresent": false}
            }
        }
    ]
}
```

Amazon EC2：限制終止 EC2 執行個體到 IP 地址範圍

此範例會示範如何建立身分型政策，透過允許動作限制 EC2 執行個體，但當請求來自指定 IP 範圍之外時明確拒絕存取。當您公司的 IP 地址在指定範圍內時，該政策很有用。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

如果將此政策與其他允許執`ec2:TerminateInstances`行動作的政策 (例如 AmazonEC2 FullAccess AWS 受管政策) 結合使用，則存取將被拒絕。這是因為明確拒絕陳述式的優先順序會在允許陳述式。如需詳細資訊，請參閱 the section called “決定是否允許或拒絕帳戶中的請求”。

> **Important**
>
> `aws:SourceIp`條件金鑰會拒絕存取代表您撥打電話的 AWS 服務 AWS CloudFormation，例如。如需有關使用 `aws:SourceIp` 條件索引鍵的詳細資訊，請參閱 AWS 全域條件內容索引鍵。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ec2:TerminateInstances"],
            "Resource": ["*"]
        },
        {
            "Effect": "Deny",
            "Action": ["ec2:TerminateInstances"],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            },
            "Resource": ["*"]
        }
    ]
}
```

IAM：存取政策模擬器 API

此範例會示範如何建立身分型政策，允許將政策模擬器 API 用於連接到目前 AWS 帳戶中的使用者、群組或角色的政策。此政策還允許存取模擬做為字串傳送到 API 的不太敏感的政策。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForCustomPolicy",
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulateCustomPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 若要允許使用者存取政策模擬器主控台，以模擬連接到目前使用者、群組或角色的策略 AWS 帳戶，請參閱IAM：存取政策模擬器主控台。

IAM：存取政策模擬器主控台

此範例會示範如何建立身分型政策，允許將政策模擬器主控台用於連接到目前 AWS 帳戶中的使用者、群組或角色的政策。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。

您可前往 https://policysim.aws.amazon.com/ 存取 IAM 政策模擬器主控台

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetGroup",
                "iam:GetGroupPolicy",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:GetUser",
                "iam:GetUserPolicy",
```

```
                "iam:ListAttachedGroupPolicies",
                "iam:ListAttachedRolePolicies",
                "iam:ListAttachedUserPolicies",
                "iam:ListGroups",
                "iam:ListGroupPolicies",
                "iam:ListGroupsForUser",
                "iam:ListRolePolicies",
                "iam:ListRoles",
                "iam:ListUserPolicies",
                "iam:ListUsers"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

IAM：擔任具有特定標籤的角色

此範例會示範如何建立身分型政策，允許 IAM 使用者擔任具標籤鍵值對 `Project = ExampleCorpABC` 的角色。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的*########*取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

如果具此標籤的角色存在於與使用者相同的帳戶中，則該使用者可擔任該角色。如果具此標籤的角色存在於與使用者不同的帳戶中，則需要額外的許可。跨帳戶角色的信任政策還必須允許使用者或使用者帳戶的所有成員擔任該角色。如需使用跨帳戶存取之角色的詳細資訊，請參閱 在您擁有的另一個 AWS 帳戶 IAM 使用者中提供存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeTaggedRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:ResourceTag/Project": "ExampleCorpABC"}
            }
        }
    ]
```

```
}
```

IAM：以程式設計方式和在主控台中允許和拒絕對多個服務的存取

此範例會示範如何建立身分型政策，允許完整存取 IAM 中的多項服務和有限的自我管理存取權限。此政策也拒絕對 Amazon S3 `logs` 儲存貯體或 Amazon EC2 `i-1234567890abcdef0` 執行個體的存取權限。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的`########`取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

> **Warning**
>
> 此政策允許在多個服務中對每個動作和資源的完整存取。此政策應該僅適用於信任的管理員。

您可以將此政策用作為許可界線，來定義以身分為基礎的政策可以授予 IAM 使用者的許可上限。如需詳細資訊，請參閱 使用許可界限委派責任給他人。將此政策用作為使用者的許可界線時，此陳述式會定義以下界線：

- 此 `AllowServices` 陳述式會允許對指定 AWS 服務的完整存取。這表示使用者在這些服務的動作，僅受限於連接到使用者的許可政策。
- `AllowIAMConsoleForCredentials` 陳述式允許列出所有 IAM 使用者的存取權。這個存取權是導覽 AWS Management Console中 Users (使用者) 頁面的必要條件。它也允許檢視帳戶的密碼要求，對於要變更自身密碼的使用者來說這是必要的程序。
- `AllowManageOwnPasswordAndAccessKeys` 陳述式允許使用者僅管理自己的主控台密碼和程式設計存取金鑰。這很重要，因為如果另一個政策提供使用者完整的 IAM 存取權，則該使用者便可以變更自己或其他使用者的許可。此陳述式可避免此種情況發生。
- `DenyS3Logs` 陳述式明確拒絕存取 `logs` 儲存貯體。此政策會向該使用者強制執行公司限制。
- `DenyEC2Production` 陳述式明確拒絕存取 `i-1234567890abcdef0` 執行個體。

此政策不允許存取其他服務或動作。當原則作為使用者的權限界限使用時，即使附加至使用者的其他策略允許這些動作，也會 AWS 拒絕要求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServices",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMConsoleForCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswordAndAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*LoginProfile*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::logs",
                "arn:aws:s3:::logs/*"
            ]
        },
        {
            "Sid": "DenyEC2Production",
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
        }
    ]
```

```
}
```

IAM：將特定標籤新增至具有特定標籤的使用者

此範例會示範如何建立身分型政策，允許對 IAM 使用者新增具有標籤值 Marketing、Development 或 QualityAssurance 的標籤索引鍵 Department。該使用者必須已包含標籤鍵值對 JobFunction = manager。您可以使用此政策要求經理僅隸屬於下列三個部門之一。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

ListTagsForAllUsers 陳述式可讓您檢視帳戶中的所有使用者。

TagManagerWithSpecificDepartment 陳述式中的第一個條件是使用 StringEquals 條件運算子。如果條件的兩個部分都傳回 true，該條件就會傳回 true。即將標記的使用者必須已經擁有 JobFunction=Manager 標籤。該請求必須包含 Department 標籤鍵，以及其中一個列出的標籤值。

第二個條件會使用 ForAllValues:StringEquals 條件運算子。如果請求中的所有標籤鍵與政策中的鍵相符合，條件會傳回 true。這表示請求中的唯一標籤鍵一定是 Department。如需使用 ForAllValues 的詳細資訊，請參閱 多值內容索引鍵。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListTagsForAllUsers",
            "Effect": "Allow",
            "Action": [
                "iam:ListUserTags",
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "TagManagerWithSpecificDepartment",
            "Effect": "Allow",
            "Action": "iam:TagUser",
            "Resource": "*",
            "Condition": {"StringEquals": {
                "iam:ResourceTag/JobFunction": "Manager",
                "aws:RequestTag/Department": [
                    "Marketing",
```

```
                    "Development",
                    "QualityAssurance"
                    ]
                },
                "ForAllValues:StringEquals": {"aws:TagKeys": "Department"}
            }
        }
    ]
}
```

IAM：新增特定標籤與特定值

此範例會示範如何建立身分型政策，允許對任何 IAM 使用者或角色僅新增標籤索引鍵 CostCenter 以及標籤值 A-123 或標籤值 B-456。您可以使用這個政策，限制標記至特定標籤鍵與一組標籤值。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

ConsoleDisplay 陳述式可讓您檢視帳戶中的所有使用者與角色。

AddTag 陳述式中的第一個條件是使用 StringEquals 條件運算子。如果請求的 CostCenter 標籤鍵包含其中一個列出的標籤值，該條件就會傳回 true。

第二個條件會使用 ForAllValues:StringEquals 條件運算子。如果請求中的所有標籤鍵與政策中的鍵相符合，條件會傳回 true。這表示請求中的唯一標籤鍵一定是 CostCenter。如需使用 ForAllValues 的詳細資訊，請參閱 多值內容索引鍵。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConsoleDisplay",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetUser",
                "iam:ListRoles",
                "iam:ListRoleTags",
                "iam:ListUsers",
                "iam:ListUserTags"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "AddTag",
            "Effect": "Allow",
            "Action": [
                "iam:TagUser",
                "iam:TagRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CostCenter": [
                        "A-123",
                        "B-456"
                    ]
                },
                "ForAllValues:StringEquals": {"aws:TagKeys": "CostCenter"}
            }
        }
    ]
}
```

IAM：建立僅具有特定標籤的新使用者

此範例會示範如何建立身分型政策，允許建立 IAM 使用者，但僅能使用 Department 和 JobFunction 標籤鍵 (其中之一或兩者)。Department 標籤鍵必須擁有 Development 或 QualityAssurance 標籤值。JobFunction 標籤鍵必須擁有 Employee 標籤值。您可以使用此政策要求新使用者具有特定任務函數與部門。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

陳述式中的第一個條件是使用 StringEqualsIfExists 條件運算子。如果請求中有採用 Department 或 JobFunction 鍵的標籤，則該標籤必須擁有指定值。如果上述鍵都不存在，則這個條件會評估為 true。只有當請求中出現其中一個指定的條件金鑰，且包含的值不同於允許的值時，條件才會評估為 false。如需使用 IfExists 的詳細資訊，請參閱 ... IfExists 條件運算子。

第二個條件會使用 ForAllValues:StringEquals 條件運算子。如果請求中的每個指定的標籤鍵值，至少與政策中的一個值之間相符，條件就會傳回 true。這表示請求中的所有標籤都一定會屬於這份清單。不過，請求可以在清單中僅包含其中一個標籤。例如，您可以建立僅包含 Department=QualityAssurance 標籤的 IAM 使用者。不過，您無法建立同時包含 JobFunction=employee 標籤和 Project=core 標籤的 IAM 使用者。如需使用 ForAllValues 的詳細資訊，請參閱 多值內容索引鍵。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "TagUsersWithOnlyTheseTags",
            "Effect": "Allow",
            "Action": [
                "iam:CreateUser",
                "iam:TagUser"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "aws:RequestTag/Department": [
                        "Development",
                        "QualityAssurance"
                    ],
                    "aws:RequestTag/JobFunction": "Employee"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Department",
                        "JobFunction"
                    ]
                }
            }
        }
    ]
}
```

IAM：產生和擷取 IAM 憑證報告

此範例說明如何建立身分型政策，讓使用者產生並下載列出其中所有 IAM 使用者的報告。 AWS 帳戶此報告包含使用者憑證的狀態，包括密碼、存取金鑰、MFA 裝置和簽章憑證。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。

如需有關憑證報告的詳細資訊，請參閱 取得 AWS 帳戶的憑證報告。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Action": [
            "iam:GenerateCredentialReport",
            "iam:GetCredentialReport"
        ],
        "Resource": "*"
    }
}
```

IAM：允許以程式設計方式及在主控台中管理群組的成員資格

此範例會示範如何建立身分型政策，允許更新名為 MarketingTeam 的群組成員。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

此政策的功能為何？

- ViewGroups 陳述式可讓使用者在 AWS Management Console中列出所有使用者和群組。它還可讓使用者檢視帳戶中使用者的相關基本資訊。這些許可必須位於自己的陳述式中，因為它們不支援或不需要指定資源 ARN。而是許可指定 "Resource" : "*"。
- ViewEditThisGroup 陳述式可讓使用者檢視 MarketingTeam 群組的相關資訊，以及從該群組新增和移除使用者。

此政策不允許使用者檢視或編輯使用者或 MarketingTeam 群組的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewGroups",
            "Effect": "Allow",
            "Action": [
                "iam:ListGroups",
                "iam:ListUsers",
                "iam:GetUser",
                "iam:ListGroupsForUser"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ViewEditThisGroup",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:AddUserToGroup",
                "iam:RemoveUserFromGroup",
                "iam:GetGroup"
            ],
            "Resource": "arn:aws:iam::*:group/MarketingTeam"
        }
    ]
}
```

IAM：管理特定標籤

此範例會示範如何建立身分型政策，允許使用標籤索引鍵 Department 從 IAM 實體 (使用者和角色) 新增和移除 IAM 標籤。此政策不會限制 Department 標籤的值。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:TagUser",
            "iam:TagRole",
            "iam:UntagUser",
            "iam:UntagRole"

        ],
        "Resource": "*",
        "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": "Department"}}
    }
}
```

IAM：將 IAM 角色傳遞給特定 AWS 服務

此範例顯示如何建立以身分為基礎的政策，以允許將任何 IAM 服務角色傳遞給 Amazon CloudWatch 服務。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

服務角色是 IAM 角色，可將 AWS 服務指定為可擔任該角色的主體。此角色可讓服務擔任角色並代表您存取其他服務中的資源。若要允許 Amazon CloudWatch 擔任您傳遞的角色，您必須在角色的信任

政策中指定cloudwatch.amazonaws.com服務主體做為主體。服務主體是由服務定義。若要了解服務的服務主體，請參閱該服務的文件。對於某些服務，請參閱 AWS 與 IAM 搭配使用的服務 並尋找 Service-Linked Role (服務連結角色) 資料欄具有 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。搜尋 amazonaws.com 來檢視服務主體。

若要進一步了解將服務角色傳遞到服務的詳細資訊，請參閱 授予使用者將角色傳遞至 AWS 服務的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:PassedToService": "cloudwatch.amazonaws.com"}
            }
        }
    ]
}
```

IAM：允許對 IAM 主控台的唯讀存取，而不需要報告

此範例會示範如何建立身分型政策，允許 IAM 使用者執行以字串 Get 或 List 開頭的任何 IAM 動作。當使用者使用主控台時，主控台會向 IAM 發出請求，以列出群組、使用者、角色和政策，並產生與這些資源相關的報告。

星號會做為萬用字元。當您在政策中使用 iam:Get* 時，產生的許可包括以 Get 開頭的所有 IAM 動作，例如 GetUser 和 GetRole。如果在未來將新實體類型新增到 IAM 時萬用字元會很有用。在這種情況下，此政策授予的許可會自動允許使用者列出和取得這些新實體的詳細資訊。

此政策無法用於產生報告或服務上次存取的詳細資訊。如需了解允許此操作的其他政策，請參閱IAM：允許對 IAM 主控台的唯讀存取權。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:Get*",
```

```
            "iam:List*"
        ],
        "Resource": "*"
    }
}
```

IAM：允許對 IAM 主控台的唯讀存取權

此範例會示範如何建立身分型政策，允許 IAM 使用者執行以字串 Get、List 或 Generate 開頭的任何 IAM 動作。當使用者使用 IAM 主控台時，該主控台會發出請求，以列出群組、使用者、角色和政策，並產生與這些資源相關的報告。

星號會做為萬用字元。當您在政策中使用 iam:Get* 時，產生的許可包括以 Get 開頭的所有 IAM 動作，例如 GetUser 和 GetRole。使用萬用字元是有幫助的，特別是在未來將新的實體類型新增到 IAM 的情況。在這種情況下，此政策授予的許可會自動允許使用者列出和取得這些新實體的詳細資訊。

使用此政策進行主控台存取，其中包含產生報告或服務上次存取詳細資訊的許可。如需不允許產生動作的其他政策，請參閱 IAM：允許對 IAM 主控台的唯讀存取，而不需要報告。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:Get*",
            "iam:List*",
            "iam:Generate*"
        ],
        "Resource": "*"
    }
}
```

IAM：允許特定 IAM 使用者以程式設計方式在主控台中管理群組

此範例會示範如何建立身分型政策，允許特定 IAM 使用者管理 AllUsers 群組。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

此政策的功能為何？

- 此 AllowAllUsersToListAllGroups 陳述式允許列出所有群組。主控台存取需要此功能。由於此許可不支援資源 ARN，因此此許可必須在其陳述式中。而是許可指定 "Resource" : "*"。
- 此 AllowAllUsersToViewAndManageThisGroup 陳述式允許可在群組資源類型上執行的所有群組動作。它不允許 ListGroupsForUser 動作，此動作可在使用者資源類型而不是群組資源類型上執行。如需有關可以為 IAM 動作指定的資源類型的詳細資訊，請參閱 AWS Identity and Access Management的動作、資源和條件金鑰。
- LimitGroupManagementAccessToSpecificUsers 陳述式會拒絕具指定名稱的使用者的寫入存取和許可管理群組動作。當政策中指定的使用者嘗試對群組進行變更，此陳述式不會拒絕請求。AllowAllUsersToViewAndManageThisGroup 陳述式會允許該請求。如果其他使用者嘗試執行這些操作，請求會被拒絕。您可以在 IAM 主控台建立此政策時，檢視透過 Write (寫入) 或 Permissions management (許可管理) 存取層級拒絕的 IAM 動作。若要執行此操作，請從 JSON 索引標籤切換到 Visual editor (視覺編輯器) 索引標籤。如需有關存取層級的詳細資訊，請參閱 AWS Identity and Access Management的動作、資源和條件金鑰。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllUsersToListAllGroups",
            "Effect": "Allow",
            "Action": "iam:ListGroups",
            "Resource": "*"
        },
        {
            "Sid": "AllowAllUsersToViewAndManageThisGroup",
            "Effect": "Allow",
            "Action": "iam:*Group*",
            "Resource": "arn:aws:iam::*:group/AllUsers"
        },
        {
            "Sid": "LimitGroupManagementAccessToSpecificUsers",
            "Effect": "Deny",
            "Action": [
                "iam:AddUserToGroup",
                "iam:CreateGroup",
                "iam:RemoveUserFromGroup",
                "iam:DeleteGroup",
                "iam:AttachGroupPolicy",
                "iam:UpdateGroup",
                "iam:DetachGroupPolicy",
```

```
                "iam:DeleteGroupPolicy",
                "iam:PutGroupPolicy"
            ],
            "Resource": "arn:aws:iam::*:group/AllUsers",
            "Condition": {
                "StringNotEquals": {
                    "aws:username": [
                        "srodriguez",
                        "mjackson",
                        "adesai"
                    ]
                }
            }
        }
    ]
}
```

IAM：允許以程式設計方式在主控台中設定帳戶密碼要求

此範例會示範如何建立身分型政策，允許使用者檢視及更新其帳戶的密碼要求。密碼要求會指定帳戶成員密碼的複雜性要求和強制輪換期間。此政策定義了程式設計和主控台存取的許可。

若要了解如何為您的帳戶設定帳戶密碼要求政策，請參閱 設定 IAM 使用者的帳戶密碼政策。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GetAccountPasswordPolicy",
            "iam:UpdateAccountPasswordPolicy"
        ],
        "Resource": "*"
    }
}
```

IAM：根據使用者路徑存取政策模擬器 API

此範例會示範如何建立身分型政策，允許僅對具有路徑 Department/Development 的使用者使用政策模擬器 API。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:user/Department/Development/*"
        }
    ]
}
```

> **Note**
>
> 若要建立允許有路徑 Department/Development 的使用者使用政策模擬器主控台的政策，請參閱 IAM：根據使用者路徑存取政策模擬器主控台。

IAM：根據使用者路徑存取政策模擬器主控台

此範例會示範如何建立身分型政策，允許僅對具有路徑 Department/Development 的使用者使用政策模擬器主控台。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

您可前往 https://policysim.aws.amazon.com/ 存取 IAM 政策模擬器。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetPolicy",
                "iam:GetUserPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
```

```
            "Action": [
                "iam:GetUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListGroupsForUser",
                "iam:ListUserPolicies",
                "iam:ListUsers"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:user/Department/Development/*"
        }
    ]
}
```

IAM：可讓 IAM 使用者自行管理 MFA 裝置

此範例會示範如何建立身分型政策，允許 IAM 使用者自行管理其多重要素驗證 (MFA) 裝置。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。

> **Note**
>
> 如果具有此政策的 IAM 使用者未經過 MFA 驗證，則此政策會拒絕存取所有動作，但使用 MFA 進行驗證所需的 AWS 動作除外。如果您為已登入的使用者新增這些權限 AWS，他們可能需要先登出再重新登入才能看到這些變更。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToCreateVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
```

```
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowUserToManageTheirOwnMFA",
            "Effect": "Allow",
            "Action": [
                "iam:EnableMFADevice",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowUserToDeactivateTheirOwnMFAOnlyWhenUsingMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ],
            "Condition": {
                "Bool": {
                    "aws:MultiFactorAuthPresent": "true"
                }
            }
        },
        {
            "Sid": "BlockMostAccessUnlessSignedInWithMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:ListMFADevices",
                "iam:ListUsers",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
```

```
                }
            }
        }
    ]
}
```

IAM：允許 IAM 使用者以程式設計方式和在主控台更新自己的憑證

此範例會示範如何建立身分型政策，允許 IAM 使用者更新自己的存取金鑰、簽署憑證、服務專用憑證和密碼。此政策定義了程式設計和主控台存取的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*ServiceSpecificCredential*",
                "iam:*SigningCertificate*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        }
    ]
}
```

若要了解使用者可在主控台中變更密碼的詳細資訊，請參閱 the section called “IAM 使用者如何變更自己的密碼”。

IAM：檢視 Organizations 政策上次存取的服務資訊

此範例會示範如何建立身分型政策，允許檢視特定組織政策的服務上次存取資訊。此政策允許使用 p-policy123 ID 來擷取服務控制政策 (SCP) 的資料。產生及檢視報表的人員必須使用 AWS Organizations 管理帳戶認證進行驗證。此政策可讓請求者在其組織中擷取任何 Organizations 實體的資料。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

如需上次存取資訊的重要資訊，包含必要的許可、疑難排解及支援區域，請參閱AWS 使用上次存取的資訊精簡權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOrgsReadOnlyAndIamGetReport",
            "Effect": "Allow",
            "Action": [
                "iam:GetOrganizationsAccessReport",
                "organizations:Describe*",
                "organizations:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowGenerateReportOnlyForThePolicy",
            "Effect": "Allow",
            "Action": "iam:GenerateOrganizationsAccessReport",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:OrganizationsPolicyId": "p-policy123"}
            }
        }
    ]
}
```

IAM：限制可套用至 IAM 使用者、群組或角色的受管政策

此範例說明如何建立以身分識別為基礎的政策，以限制可套用至 IAM 使用者、群組或角色的客戶 AWS 受管政策。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:AttachUserPolicy",
            "iam:DetachUserPolicy"
        ],
        "Resource": "*",
        "Condition": {
            "ArnEquals": {
                "iam:PolicyARN": [
                    "arn:aws:iam::*:policy/policy-name-1",
                    "arn:aws:iam::*:policy/policy-name-2"
                ]
            }
        }
    }
}
```

AWS：拒絕存取帳戶外的資源，但 AWS 受管 IAM 政策除外

在身分型政策中使用 aws:ResourceAccount 可能會影響使用者或角色利用某些服務，這些服務需要與某項服務所擁有帳戶中的資源互動。

您可以建立具有例外情況的政策，以允許 AWS 受管 IAM 政策。Organ AWS izations 外的服務管理帳戶擁有受管 IAM 政策。有四個 IAM 動作可列出並擷取 AWS受管政策。在陳述式的 NotAction 元素中使用這些動作。政策中的 AllowAccessToS3ResourcesInSpecificAccountsAndSpecificService1。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToResourcesInSpecificAccountsAndSpecificService1",
      "Effect": "Deny",
      "NotAction": [
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListEntitiesForPolicy",
        "iam:ListPolicies"
      ],
```

```
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    }
  ]
}
```

AWS Lambda：允許 Lambda 函數存取 Amazon DynamoDB 資料表

此範例會示範如何建立身分型政策，允許對特定 Amazon DynamoDB 資料表進行讀寫存取。此策略也允許將記錄檔寫入 CloudWatch 記錄檔。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

若要使用此政策，請將政策連接到 Lambda 服務角色。服務角色是您在帳戶中建立的角色，以允許服務來代表您執行動作。該服務角色必須包含 AWS Lambda 為信任原則中的主體。如需如何使用此政策的詳細資訊，請參閱 AWS 安全部落格中的如何建立 AWS IAM 政策以授與 Amazon DynamoDB 表的 AWS Lambda 存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/SampleTable"
        },
        {
            "Sid": "GetStreamRecords",
```

```
            "Effect": "Allow",
            "Action": "dynamodb:GetRecords",
            "Resource": "arn:aws:dynamodb:*:*:table/SampleTable/stream/* "
        },
        {
            "Sid": "WriteLogStreamsAndGroups",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateLogGroup",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        }
    ]
}
```

Amazon RDS：允許在特定區域內的完整 RDS 資料庫存取

此範例會示範如何建立身分型政策，允許在特定區域內完整存取 RDS 資料庫。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的*######*取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:*",
            "Resource": ["arn:aws:rds:region:*:*"]
        },
        {
            "Effect": "Allow",
            "Action": ["rds:Describe*"],
            "Resource": ["*"]
        }
    ]
```

```
}
```

Amazon RDS：允許以程式設計方式以及在主控台之中還原 RDS 資料庫

此範例會示範如何建立身分型政策，允許還原 RDS 資料庫。此政策定義了程式設計和主控台存取的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSnapshot",
                "rds:DeleteDBSnapshot",
                "rds:Describe*",
                "rds:DownloadDBLogFilePortion",
                "rds:List*",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyOptionGroup",
                "rds:RebootDBInstance",
                "rds:RestoreDBInstanceFromDBSnapshot",
                "rds:RestoreDBInstanceToPointInTime"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon RDS：允許標籤持有者擁有對已標記的 RDS 資源的完整存取權限

此範例會示範如何建立身分型政策，允許標籤擁有者完整存取他們已加上標籤的 RDS 資源。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
```

```
                "rds:Describe*",
                "rds:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "rds:DeleteDBInstance",
                "rds:RebootDBInstance",
                "rds:ModifyDBInstance"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:db-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyOptionGroup",
                "rds:DeleteOptionGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:og-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyDBParameterGroup",
                "rds:ResetDBParameterGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:pg-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:AuthorizeDBSecurityGroupIngress",
                "rds:RevokeDBSecurityGroupIngress",
```

```
                "rds:DeleteDBSecurityGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:secgrp-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:DeleteDBSnapshot",
                "rds:RestoreDBInstanceFromDBSnapshot"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:snapshot-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyDBSubnetGroup",
                "rds:DeleteDBSubnetGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:subgrp-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyEventSubscription",
                "rds:AddSourceIdentifierToSubscription",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:DeleteEventSubscription"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:es-tag/Owner": "${aws:username}"}
            }
        }
    ]
```

```
}
```

Amazon S3：可讓 Amazon Cognito 使用者存取其儲存貯體中的物件

此範例會示範如何建立身分型政策，允許 Amazon Cognito 使用者存取特定 S3 儲存貯體中的物件。此政策僅允許存取名稱包含 cognito 的物件、應用程式的名稱和聯合身分使用者的 ID，由 ${cognito-identity.amazonaws.com:sub} 變數呈現。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

> **Note**
>
> 物件索引鍵中使用的 'sub' 值不是使用者集區中的使用者的 sub 值，而是身分集區中與使用者相關聯的身分 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListYourObjects",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
          ]
        }
      }
    },
    {
      "Sid": "ReadWriteDeleteYourObjects",
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
```

```
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/cognito/application-name/${cognito-
identity.amazonaws.com:sub}/*"
      ]
    }
  ]
}
```

Amazon Cognito 為您的 Web 和行動應用程式提供身分驗證、授權和使用者管理。您的使用者可透過使用者名稱和密碼直接登入，或透過第三方 (例如 Facebook、Amazon 或 Google) 登入。

Amazon Cognito 的兩個主要元件是使用者集區和身分集區。使用者集區是一種使用者目錄，能為應用程式使用者提供註冊和登入的選項。身分集區可讓您授與使用者存取其他 AWS 服務。您可以單獨或一併使用身分集區和使用者集區。

如需有關 Amazon Cognito 的詳細資訊，請參閱 Amazon Cognito 使用者指南。

Amazon S3：可讓聯合身分使用者以程式設計方式和在主控台存取其 S3 主目錄

此範例會示範如何建立身分型政策，允許聯合身分使用者在 S3 中存取自己的主目錄儲存貯體物件。主目錄是一個儲存貯體，其中包含 home 資料夾和個別聯合身分使用者的資料夾。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

此政策中的 ${aws:userid} 變數會解析成 role-id:specified-name。聯合身分使用者 ID 的 role-id 部分是在建立期間指派給聯合身分使用者的唯一識別符。如需詳細資訊，請參閱 唯一識別碼。當同盟使用者擔任其角色時，會傳送至AssumeRoleWithWebIdentity要求的RoleSessionName 參數。specified-name

您可以使用 AWS CLI 命令檢視角色 ID aws iam get-role --role-name *specified-name*。例如，假設您指定易讀的名稱 John 且 CLI 傳回該角色 ID AROAXXT2NJT7D3SIQN7Z6。在這種情況下，聯合身分使用者 ID 會是 AROAXXT2NJT7D3SIQN7Z6:John。此政策然後就會允許聯合身分使用者 John 使用字首 AROAXXT2NJT7D3SIQN7Z6:John 存取 Amazon S3 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::bucket-name",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "",
                        "home/",
                        "home/${aws:userid}/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name/home/${aws:userid}",
                "arn:aws:s3:::bucket-name/home/${aws:userid}/*"
            ]
        }
    ]
}
```

Amazon S3：S3 儲存貯體的存取，但生產儲存貯體在沒有最新 MFA 的情況下遭拒

此範例會示範如何建立身分型政策，允許 Amazon S3 管理員存取任何儲存貯體，包括更新、新增和刪除物件。不過，如果使用者沒有在最近三十分鐘內使用多重要素驗證 (MFA) 登入，則會明確拒絕 Production 儲存貯體的存取權。此原則會授與在主控台中或以程式設計方式使用 AWS CLI 或 AWS API 執行此動作所需的權限。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

此政策永遠不會使用長期使用者存取金鑰來允許以程式設計方式存取 Production 儲存貯體。這是使用 aws:MultiFactorAuthAge 條件索引鍵搭配 NumericGreaterThanIfExists 條件運算子一起完成。如果 MFA 不存在或 MFA 的存在時間大於 30 分鐘，則此政策條件會傳回 true。在這些情況下會拒絕存取。若要以程式設計方式存取儲存Production貯體，S3 管理員必須使用過去 30 分鐘內使用 GetSessionTokenAPI 作業產生的臨時登入資料。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAllS3Buckets",
            "Effect": "Allow",
            "Action": ["s3:ListAllMyBuckets"],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketLevelActions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketObjectActions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::*/*"
        },
        {
            "Sid": "RequireMFAForProductionBucket",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::Production/*",
                "arn:aws:s3:::Production"
```

```
            ],
            "Condition": {
                "NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge": "1800"}
            }
        }
    ]
}
```

Amazon S3：可讓 IAM 使用者以程式設計方式和在主控台存取其 S3 主目錄

此範例會示範如何建立身分型政策，允許 IAM 使用者在 S3 中存取自己的主目錄儲存貯體物件。主目錄是一個儲存貯體，其中包含 home 資料夾和個別使用者的資料夾。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

使用 IAM 角色時，此政策將無法運作，因為使用 IAM 角色時無法使用 aws:username 變數。如需有關主要鍵值的詳細資訊，請參閱 主體索引鍵值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::bucket-name",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
```

```
                        "",
                        "home/",
                        "home/${aws:username}/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name/home/${aws:username}",
                "arn:aws:s3:::bucket-name/home/${aws:username}/*"
            ]
        }
    ]
}
```

Amazon S3：限制管理特定的 S3 儲存貯體

此範例會示範如何建立身分型政策，將 Amazon S3 儲存貯體的管理限制到特定儲存貯體。此政策授予執行所有 Amazon S3 動作的許可，但拒絕存取每個 AWS 服務 ，但 Amazon S3 除外。請參閱以下範例。根據此政策，您只能存取可對 S3 儲存貯體或 S3 物件資源執行的 Amazon S3 動作。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

如果此政策與允許此政策拒絕動作的其他政策 (例如 AmazonS3 FullAccess 或 AmazonEC2 FullAccess AWS 受管政策) 結合使用，則存取將被拒絕。這是因為明確拒絕陳述式的優先順序會在允許陳述式。如需詳細資訊，請參閱 the section called “決定是否允許或拒絕帳戶中的請求”。

> **Warning**
>
> NotAction 和 NotResource 是必須謹慎使用的進階政策元素。此政策拒絕存取除 Amazon S3 以外的每個 AWS 服務。如果您將此政策連接到使用者，則會忽略向其他服務授予許可的任何其他政策，並拒絕存取。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
            "Effect": "Deny",
            "NotAction": "s3:*",
            "NotResource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

Amazon S3：允許對 S3 儲存貯體中的物件讀取和寫入存取權

此範例會示範如何建立身分型政策，允許 Read 和 Write 存取特定 S3 儲存貯體中的物件。此原則會授與從 AWS API 或以程式設計方式完成此動作所需的權限 AWS CLI。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

s3:*Object 動作在動作名稱中使用萬用字元。AllObjectActions 陳述式允許 GetObject、DeleteObject、PutObject 以及任何其他以字元「Object」結尾的 Amazon S3 動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::bucket-name"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::bucket-name/*"]
```

```
        }
    ]
}
```

> **Note**
>
> 若要允許 Read 和 Write 存取 Amazon S3 儲存貯體中的物件，並包含對主控台的其他存取權限，請參閱 Amazon S3：允許以程式設計方式和在主控台以讀取和寫入存取 S3 儲存貯體中的物件。

Amazon S3：允許以程式設計方式和在主控台以讀取和寫入存取 S3 儲存貯體中的物件

此範例會示範如何建立身分型政策，允許 Read 和 Write 存取特定 S3 儲存貯體中的物件。此政策定義了程式設計和主控台存取的許可。若要使用此政策，請將範例政策中的########取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

s3:*Object 動作在動作名稱中使用萬用字元。AllObjectActions 陳述式允許 GetObject、DeleteObject、PutObject 以及任何其他以字元「Object」結尾的 Amazon S3 動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
```

```
            "Action": "s3:ListBucket",
            "Resource": ["arn:aws:s3:::bucket-name"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::bucket-name/*"]
        }
    ]
}
```

管理 IAM 政策

IAM 提供您建立並管理所有 IAM 政策類型的工具 (受管政策與內嵌政策)。若要新增許可到 IAM 身分 (IAM 使用者、群組或角色)，您可以建立政策、驗證政策，然後將政策連接至身分。您可以將多個政策連接到身分，而每個政策可以包含多個許可。

請參閱這些資源了解詳細資訊：

- 如需有關不同 IAM 類型的詳細資訊，請參閱 IAM 中的政策和許可。
- 如需在 IAM 中使用政策的一般資訊，請參閱 AWS 資源存取管理。
- 如需了解當指定 IAM 身分有多個有效政策時的許可評估方式，請參閱 政策評估邏輯。
- AWS 帳戶中 IAM 資源的數量和大小有限。如需更多詳細資訊，請參閱 IAM 和 AWS STS 配額。

主題

- 建立 IAM 政策
- 驗證 IAM 政策
- 根據存取活動產生政策
- 使用 IAM 政策模擬器測試 IAM 政策
- 新增和移除 IAM 身分許可
- 版本控制 IAM 政策
- 編輯 IAM 政策
- 刪除 IAM 政策
- AWS 使用上次存取的資訊精簡權限

建立 IAM 政策

政策為一個實體，可定義其所連接的身分或資源的許可。您可以使用 AWS Management Console AWS CLI、或 AWS API 在 IAM 中建立客戶受管政策。客戶管理政策是獨立的政策，在您自己的 AWS 帳戶進行管理。然後，您可以將原則附加到 AWS 帳戶.

IAM 中連接到身分的政策又稱為身分型政策。以身分識別為基礎的政策可以包括 AWS 受管政策、客戶管理的策略和內嵌政策。 AWS 受管理的策略是由建立和管理 AWS。您可以使用這些政策，但無法管理。內嵌政策是您建立並接內嵌至 IAM 群組、使用者或角色的政策。內嵌政策無法在其他身分上重複使用，或在其存在的身分之外進行管理。如需詳細資訊，請參閱 新增和移除 IAM 身分許可。

使用客戶管理政策而不是內嵌政策。同時，最好使用客戶管理的政策，而不是 AWS 受管政策。 AWS 受管理的原則通常提供廣泛的管理或唯讀權限。為了達到最高安全性，授予最低權限只會授予執行特定任務工作的許可。

建立或編輯 IAM 政策時， AWS 可以自動執行政策驗證，以協助您建立有效的政策，並考慮到最低權限。在中 AWS Management Console，IAM 可識別 JSON 語法錯誤，而 IAM 存取分析器則提供其他政策檢查以及建議，以協助您進一步調整政策。若要進一步了解政策驗證的資訊，請參閱 驗證 IAM 政策。若要進一步了解 IAM Access Analyzer 政策檢查和可動作的建議，請參閱 IAM Access Analyzer 政策驗證。

您可以使用 AWS Management Console AWS CLI、或 AWS API 在 IAM 中建立客戶受管政策。如需有關使用 AWS CloudFormation 範本新增或更新策略的詳細資訊，請參閱《使AWS CloudFormation 用指南》中的AWS Identity and Access Management 資源類型參考。

主題

- 建立 IAM 政策 (主控台)
- 建立 IAM 政策 (AWS CLI)
- 建立身分與存取權管理政策AWS

建立 IAM 政策 (主控台)

政策為一個實體，可定義其所連接的身分或資源的許可。您可以使用在 AWS Management Console IAM 中建立客戶受管政策。客戶管理政策是獨立的政策，在您自己的 AWS 帳戶進行管理。然後，您可以將原則附加到 AWS 帳戶.

主題

- 建立 IAM 政策

- 正在使用 JSON 編輯器建立政策
- 使用視覺化編輯器來建立政策
- 匯入現有的受管政策

建立 IAM 政策

您可以使 AWS Management Console 用下列其中一種方法在中建立客戶管理政策：

- JSON — 貼上並自訂已發布的以身分為基礎的政策範例。
- 視覺編輯工具 — 在視覺編輯工具中從零開始建構一個新的政策。若您使用視覺化編輯器，您便無需了解 JSON 語法。
- 匯入 — 從帳戶中匯入並自訂受管政策。您可以匯入 AWS 受管政策或先前建立的客戶管理策略。

AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

正在使用 JSON 編輯器建立政策

您可以選擇 JSON 選項以在 JSON 中輸入或貼上政策。此方法對於複製要在帳戶中使用的範例政策很有幫助。或者，您可以在 JSON 編輯器中輸入自己的 JSON 政策文件。您也可以使用 JSON 選項，來在視覺化編輯器與 JSON 之間切換，以比較視圖。

當您建立或編輯 JSON 編輯器中的政策時，IAM 會執行政策驗證以協助您建立有效的政策。IAM 識別 JSON 語法錯誤，而 IAM Access Analyzer 會提供額外的政策檢查及可操作的建議，協助您進一步改良政策。

JSON 政策 文件為包含一或多個陳述式。每個陳述式應包含具有相同效果 (`Allow` 或 `Deny`) 並支援相同資源和條件的所有操作。如果一個動作要求指定所有資源 ("`*`")，而另一個動作支援特定資源的 Amazon Resource Name (ARN)，則它們必須位於兩個單獨的 JSON 陳述式中。如需關於 ARN 格式的詳細資料，請參閱 AWS 一般參考 指南中的 Amazon Resource Name (ARN)。如需有關 IAM 政策的一般資訊，請參閱 IAM 中的政策和許可。如需有關 IAM 政策語言的資訊，請參閱 IAM JSON 政策參考。

若要使用 JSON 政策編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。

4. 在政策編輯器中，選擇 JSON 選項。
5. 輸入或貼上 JSON 政策文件。如需有關 IAM 政策語言的詳細資訊，請參閱 IAM JSON 政策參考。
6. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

7. (選擇性) 在中建立或編輯原則時 AWS Management Console，您可以產生可在 AWS CloudFormation 範本中使用的 JSON 或 YAML 原則範本。

   若要這麼做，請在 [原則編輯器] 中選擇 [動作]，然後選擇 [產生 CloudFormation範本]。若要進一步了解，請參 AWS CloudFormation 閱《 AWS CloudFormation 使用指南》中的AWS Identity and Access Management 資源類型參考。
8. 將許可新增至政策後，請選擇下一步。
9. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，來查看您的政策所授予的許可。
10. (選用) 藉由連接標籤作為鍵值組，將中繼資料新增至政策。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
11. 選擇 Create policy (建立政策) 儲存您的新政策。

在建立政策之後，即可將它連接至您的群組、使用者或角色。如需詳細資訊，請參閱 新增和移除 IAM 身分許可。

使用視覺化編輯器來建立政策

IAM 主控台中的視覺化編輯器將引導您建立政策，而無需編寫 JSON 語法。若要檢視使用視覺化編輯器建立政策的範例，請參閱 the section called “控制對身分的存取”。

若要使用視覺化編輯器來建立政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。

4. 在 [原則編輯器] 區段中，找到 [選取服務] 區段，然後選擇 AWS 服務。您可用上方的搜尋框來限制服務清單中的結果。您僅可以選擇一項視覺化編輯器許可區塊中的服務。若要授予存取一個以上服務的許可，請選擇新增更多許可，來新增多個許可區塊。
5. 在動作中，選擇要新增至政策的動作。您可採用以下方式來選擇動作：

   - 選取所有動作的核取方塊。
   - 選擇 add actions (新增動作) 來輸入特定動作的名稱。您可以使用萬用字元 (*) 來指定多個動作。
   - 選取其中一個 Access level (存取層級) 群組，以選擇存取層級的所有動作 (例如，Read (讀取)、Write (寫入) 或 List (列出))。
   - 展開各個 Access level (存取級別) 群組來選擇個別動作。

   預設情況下，您建立的政策允許執行選擇的操作。若要拒絕選擇的動作，請選擇 Switch to deny permissions (切換為拒絕許可)。由於 IAM 會根據預設拒絕，作為安全最佳實務，我們建議您僅允許使用者所需的操作和資源的許可。只有在要覆蓋其他語句或政策單獨允許的許可時，才應建立 JSON 陳述式來拒絕許可。我們建議您將拒絕許可數限制為最低，因為它們可能會增加解決許可問題的難度。
6. 對於 Resources (資源)，如果您在先前步驟中選取的服務和動作不支援選擇特定資源，則會允許所有資源，而且您無法編輯此區段。

   如果選擇一或多個支援資源等級許可的動作，視覺化編輯器將列出這些資源。然後，您可以展開 Resources (資源) 來為您的政策指定資源。

   您可採用以下方式來指定資源：

   - 選擇新增 ARN，可根據它們的 Amazon Resource Name (ARN) 來指定資源。您可以使用視覺化 ARN 編輯器或手動列出 ARN。如需 ARN 語法的詳細資訊，請參閱 AWS 一般參考 指南中的 Amazon Resource Name (ARN)。如需使用政策之 `Resource` 元素中 ARN 的詳細資訊，請參閱IAM JSON 政策元素：Resource。
   - 選擇資源旁的此帳戶中的任何，將許可授予該類型的任何資源。
   - 選擇所有，可為服務選擇所有資源。
7. (選用) 選擇請求條件 - (選用)，為您正在建立的政策新增條件。條件可限制 JSON 政策陳述式的效果。例如，您可以指定只有在使用者的請求於特定時間範圍內發生時，使用者才能對資源執行動作。您也可以使用常用的條件，限制使用者必須使用多重要素驗證 (MFA) 裝置進行身分驗證。或者，您可以要求請求必須源自於特定 IP 地址範圍。如需可在原則條件中使用之所有內容索引鍵的清單，請參閱服務授權參考中服AWS 務的動作、資源和條件金鑰。

您可採用以下方式來選擇條件：

- 使用核取方塊來選擇常用條件。
- 選擇新增另一個條件，可指定其他條件。選擇條件的 Condition Key (條件金鑰)、Qualifier (函式) 以及 Operator (運算子)，然後輸入一個 Value (值)。若要新增超過一個值，請選擇新增。您可以將這些值視為藉由邏輯「OR」運算子相連。完成時，請選擇新增條件。

若要新增超過一個條件，請再次選擇新增另一個條件。視需要重複執行。每項條件僅適用於這一個視覺化編輯器許可區塊。所有條件的許可區塊皆須為 true 才會被視為符合。換句話說，可以將這些條件視為藉由邏輯「AND」運算子相連。

如需有關 Condition (條件) 元素的詳細資訊，請參閱 IAM JSON 政策參考 中的 IAM JSON 政策元素：Condition。

8. 若要新增更多許可區塊，請選擇新增更多許可。針對每個區塊皆重複步驟 2 到 5。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

9. (選擇性) 在中建立或編輯原則時 AWS Management Console，您可以產生可在 AWS CloudFormation 範本中使用的 JSON 或 YAML 原則範本。

   若要這麼做，請在 [原則編輯器] 中選擇 [動作]，然後選擇 [產生 CloudFormation範本]。若要進一步了解，請參 AWS CloudFormation 閱《 AWS CloudFormation 使用指南》中的AWS Identity and Access Management 資源類型參考。

10. 將許可新增至政策後，請選擇下一步。
11. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢視此政策中定義的許可，可確認您已授予想要的許可。
12. (選用) 藉由連接標籤作為鍵值組，將中繼資料新增至政策。如需有關在 IAM 中使用標籤的詳細資訊，請參閱 標記 IAM 資源。
13. 選擇 Create policy (建立政策) 儲存您的新政策。

在建立政策之後，即可將它連接至您的群組、使用者或角色。如需詳細資訊，請參閱 新增和移除 IAM 身分許可。

匯入現有的受管政策

若要建立新的政策，有一種簡單的方法是在您的帳戶中導入至少具有一部分所需許可權的現有受管政策。接著便可以自訂該政策，使其符合您的新要求。

您無法匯入內嵌政策。若要了解受管與內嵌政策之間的差異，請參閱 受管政策與內嵌政策。

若要在視覺化編輯器中匯入現有的受管政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。
4. 在政策編輯器中，選擇視覺化，然後在頁面右側選擇動作，再選擇匯入政策。
5. 在匯入政策視窗中，選擇最符合您想要放入新政策之政策內容的受管政策。您可用上方的搜尋框來限制政策清單中的結果。
6. 選擇匯入政策。

   匯入的政策新增於政策底部的新許可區塊中。
7. 使用 Visual editor (視覺編輯工具) 或選擇 JSON 來自訂您的政策。然後選擇下一步。

   > **Note**
   >
   > 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。
8. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。您之後便無法編輯這些設定。檢視此政策中定義的許可，然後選擇建立政策來儲存您的工作。

若要在 JSON 編輯器中匯入現有的受管政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 選擇 Create policy (建立政策)。
4. 在政策編輯器區段中，選擇 JSON 選項，然後在頁面右側選擇動作，再選擇匯入政策。
5. 在匯入政策視窗中，選擇最符合您想要放入新政策之政策內容的受管政策。您可用上方的搜尋框來限制政策清單中的結果。
6. 選擇匯入政策。

   來自匯入政策的陳述式新增於 JSON 政策底部。
7. 在 JSON 中自訂您的政策。解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。或者在 Visual editor (視覺編輯工具) 中自訂您的政策。然後選擇下一步。

   > **Note**
   >
   > 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。
8. 在檢視與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。您之後便無法編輯這些內容。檢視政策此政策中定義的許可，然後選擇建立政策來儲存您的工作。

在建立政策之後，即可將它連接至您的群組、使用者或角色。如需更多詳細資訊，請參閱 新增和移除 IAM 身分許可。

建立 IAM 政策 (AWS CLI)

政策為一個實體，可定義其所連接的身分或資源的許可。您可以使用在 AWS CLI IAM 中建立客戶受管政策。客戶管理政策是獨立的政策，在您自己的 AWS 帳戶進行管理。作為最佳實務，我們會建議使用您 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作。透過驗證政策，您可以先處理任何錯誤或建議，再將政策連接到您 AWS 帳戶的身分 (使用者、群組及角色)。

AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

建立 IAM 政策 (AWS CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 來建立 IAM 客戶受管政策或者內嵌政策。

若要建立客戶受管政策 (AWS CLI)

使用下列命令：

- : create-policy

為 IAM 身分 (群組、使用者或角色) 建立內嵌政策 (AWS CLI)

請使用以下其中一個命令：

- put-group-policy
- put-role-policy
- put-user-policy

> **Note**
>
> 您無法使用 IAM 為服務連結角色嵌入內嵌政策。

驗證客戶受管政策 (AWS CLI)

使用下列 IAM Access Analyzer 指令：

- validate-policy

建立身分與存取權管理政策AWS

政策為一個實體，可定義其所連接的身分或資源的許可。您可以使用 AWS API 在 IAM 中建立客戶受管政策。客戶管理政策是獨立的政策，在您自己的 AWS 帳戶進行管理。作為最佳實務，我們會建議使用您 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作。透過驗證政策，您可以先處理任何錯誤或建議，再將政策連接到您 AWS 帳戶的身分 (使用者、群組及角色)。

AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

建立身分與存取權管理政策AWS

您可以使用 AWS API 來建立 IAM 客戶受管政策或者內嵌政策。

建立客戶管理政策 (AWS API)

呼叫以下操作：

- CreatePolicy

為 IAM 身分 (群組、使用者或角色) 建立內嵌政策 (AWS API)

呼叫以下其中一項操作：

- PutGroupPolicy
- PutRolePolicy
- PutUserPolicy

> **Note**
>
> 您無法使用 IAM 為服務連結角色嵌入內嵌政策。

驗證客戶管理政策 (AWS API)

呼叫下列 IAM Access Analyzer 操作：

- ValidatePolicy

驗證 IAM 政策

政策是一種使用 IAM 政策文法的 JSON 文件。當您將政策連接到 IAM 實體 (例如使用者、群組或角色) 時，該政策會對該實體授予許可。

當您使用建立或編輯 IAM 存取控制政策時 AWS Management Console， AWS 會自動檢查這些政策，以確保它們符合 IAM 政策文法。如果 AWS 判斷政策不符合文法，則會提示您修復政策。

IAM Access Analyzer 會提供額外的政策檢查及建議，協助您進一步改良政策。若要進一步了解 IAM Access Analyzer 政策檢查和可動作的建議，請參閱 IAM Access Analyzer 政策驗證。若要檢視 IAM Access Analyzer 傳回的警告、錯誤和建議清單，請參閱 IAM Access Analyzer 政策檢查參考。

驗證範圍

AWS 檢查 JSON 政策語法和語法。同時還會驗證您的 ARN 格式是否適當，以及動作名稱和條件索引鍵是否正確。

存取政策驗證

在 AWS Management Console中建立 JSON 政策或編輯現有政策時，會自動驗證政策。如果政策語法無效，您將收到通知，並且必須先解決問題，然後才能繼續。 AWS Management Console 如果您具有的許可，則會在中自動傳回 IAM 存取分析器政策驗證中的發現項目`access-analyzer:ValidatePolicy`。您也可以使用 AWS API 或 AWS CLI.

現有政策

您可能有無效的現有政策，因為這些政策是在政策引擎的最新更新之前建立或上次儲存的政策。作為最佳實務，我們會建議使用您 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作。建議您開啟現有政策，並檢閱所產生的政策驗證結果。如果不修復任何政策語法錯誤，則無法編輯和儲存現有政策。

根據存取活動產生政策

身為管理員或開發人員，您可能會將超出其所需範圍的許可授予 IAM 實體 (使用者或角色)。IAM 提供數個選項，協助您精簡您授予的許可。其中一種選擇是產生以實體存取活動基礎的 IAM 政策。IAM Access Analyzer 會檢閱您的 AWS CloudTrail 記錄檔，並產生政策範本，其中包含實體在指定日期範圍內使用的許可。您可以使用範本來建立具有精細許可的政策，這些許可只會授予支援特定使用案例所需的許可。

例如，假設您是開發人員，而您的工程團隊一直在開發一個專案來建立新的應用程式。為了鼓勵實驗並讓您的團隊快速行動，您已在應用程式開發期間設定了具有廣泛許可的角色。現在該應用程序已準備就緒可用於生產。應用程式可以在生產帳戶中啟動之前，您只想要識別應用程式運作所需的角色許可。這有助於您符合授予最低許可的最佳實務。您可以根據您在開發帳戶中用於應用程式之角色的存取活動來產生政策。您可以進一步精簡產生的政策，然後將政策連接至生產帳戶中的實體。

若要進一步了解 IAM Access Analyzer 政策產生程序，請參閱產生 IAM Access Analyzer 政策。

使用 IAM 政策模擬器測試 IAM 政策

如需有關如何以及為什麼要使用 IAM 政策的詳細資訊，請參閱 IAM 中的政策和許可。

您可前往 https://policysim.aws.amazon.com/ 存取 IAM 政策模擬器主控台

> **Important**
>
> 政策模擬器的結果可能與您的實際 AWS 環境不同。我們建議您在使用政策模擬器進行測試後，根據您的實際 AWS 環境檢查政策，以確認您獲得所需的結果。如需詳細資訊，請參閱 IAM 政策模擬器的運作方式。

IAM 政策模擬器入門

使用 IAM 政策模擬器，您可以測試和疑難排解身分型政策和 IAM 許可界限。以下是您可以使用政策模擬器執行的一些常見作業：

- 測試連接到您 AWS 帳戶帳戶中 IAM 使用者、使用者群組或角色的身分型政策。如果將多個政策連接到使用者、使用者群組或角色，則可以測試所有政策，或選擇個別政策來測試。您可以測試特定資源選擇的政策所允許或拒絕的動作。
- 測試和疑難排解許可界限對 IAM 實體的效果。您一次只能模擬一個許可界限。
- 測試連接至 AWS 資源的 IAM 使用者資源型政策的效果，例如 Amazon S3 儲存貯體、Amazon SQS 佇列、Amazon SNS 主題或 Amazon S3 Glacier 保存庫。若要在政策模擬器中為 IAM 使用者使用資源型政策，您必須將資源併入模擬中。您還必須選取核取方塊，以將該資源的政策併入模擬中。

> **Note**
>
> IAM 角色不支援資源型政策的模擬。

- 如果您 AWS 帳戶 是中組織的成員 AWS Organizations，則可以測試服務控制原則 (SCP) 對身分型原則的影響。

> **Note**
>
> 政策模擬器不會評估有任何條件的 SCP。

- 透過輸入或將其複製到政策模擬器中來測試尚未連接到使用者、使用者群組或角色的新身分型政策。這些只用於模擬，不會儲存。您無法將資源型政策輸入或複製到政策模擬器中。
- 使用選取的服務、動作和資源來測試身分型政策。例如，您可以測試以確保您的政策可讓實體在特定儲存貯體上執行 Amazon S3 服務中的 `ListAllMyBuckets`、`CreateBucket` 和 `DeleteBucket` 動作。
- 透過提供內容索引鍵來模擬真實世界，這些內容索引鍵包含在測試政策中的 `Condition` 元素，例如 IP 地址或日期。

> **Note**
>
> 若模擬中的身分型政策沒有會明確檢查標籤的 `Condition` 元素，則政策模擬器不會模擬提供做為輸入的標籤。

- 識別身分型政策中的特定陳述式導致允許或拒絕存取特定資源或動作。

主題

IAM 政策模擬器的運作方式

政策模擬器會評估身分型政策中的陳述式，以及您在模擬期間提供的輸入。政策模擬器的結果可能與您的即時 AWS 環境不同。我們建議您在使用政策模擬器進行測試後，根據您的實際 AWS 環境檢查政策，以確認您獲得所需的結果。

政策模擬器與現場 AWS 環境的不同之處在於以下幾個方面：

- 政策模擬器不會提出實際的 AWS 服務請求，因此您可以安全地測試可能對實際 AWS 環境進行不必要更改的請求。政策模擬器不會考慮生產中的真實內容索引鍵值。
- 由於政策模擬器不會模擬執行中所選動作，因此無法向模擬請求報告任何回應。所請求的動作無論是被允許或是拒絕，只會傳回的唯一結果。
- 如果您在政策模擬器內編輯政策，這些變更只會影響政策模擬器。您中的對應政策保 AWS 帳戶 持不變。
- 您無法測試具有任何條件的服務控制政策 (SCP)。
- 政策模擬器不支援 IAM 角色和跨帳户存取權使用者的模擬。

> **Note**
>
> IAM 政策模擬器不會判定哪些服務支援用於授權的全域條件索引鍵。例如，政策模擬器不會辨別不支援 `aws:TagKeys` 的服務。

使用 IAM 政策模擬器所需的許可

您可以使用政策模擬器主控台或政策模擬器 API 來測試政策。根據預設，主控台使用者可以透過在政策模擬器中輸入或複製這些政策來測試尚未連接到使用者、使用者群組或角色的政策。這些政策僅用於模擬，不會揭露敏感資訊。API 使用者必須具有測試未連接政策的許可。您可以允許主控台或 API 使用者測試連接到 AWS 帳戶中 IAM 使用者、使用者群組或角色的政策。若要執行此動作，您必須提供擷取這些政策的許可。為了測試以資源為基礎的政策，使用者必須有擷取資源政策的許可。

有關允許使用者模擬政策的主控台和 API 政策的範例，請參閱the section called “政策範例： AWS Identity and Access Management (IAM)”。

使用政策模擬器主控台所需的許可

您可以允許使用者測試連接到 AWS 帳戶中 IAM 使用者、使用者群組或角色的政策。若要執行此動作，您必須為使用者提供擷取這些政策的許可。為了測試以資源為基礎的政策，使用者必須有擷取資源政策的許可。

若要查看允許將政策模擬器主控台用於連接至使用者、使用者群組或角色的政策的範例政策，請參閱 IAM：存取政策模擬器主控台。

若要查看僅允許具有特定路徑的使用者使用政策模擬器主控台的範例政策，請參閱 IAM：根據使用者路徑存取政策模擬器主控台。

若要建立一個政策，只允許一個類型的實體使用政策模擬器主控台，請使用下列程序。

若要允許主控台使用者模擬使用者的政策

在您的政策中包含以下動作：

- `iam:GetGroupPolicy`
- `iam:GetPolicy`
- `iam:GetPolicyVersion`
- `iam:GetUser`
- `iam:GetUserPolicy`
- `iam:ListAttachedUserPolicies`
- `iam:ListGroupsForUser`
- `iam:ListGroupPolicies`
- `iam:ListUserPolicies`
- `iam:ListUsers`

若要允許主控台使用者模擬使用者群組的政策

在您的政策中包含以下動作：

- `iam:GetGroup`

- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:ListAttachedGroupPolicies
- iam:ListGroupPolicies
- iam:ListGroups

若要允許主控台使用者模擬角色的政策

在您的政策中包含以下動作：

- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:GetRolePolicy
- iam:ListAttachedRolePolicies
- iam:ListRolePolicies
- iam:ListRoles

若要測試以資源為基礎的政策，使用者必須有擷取資源政策的許可。

若要允許主控台使用者在 Amazon S3 儲存貯體中測試以資源為基礎的政策

在您的政策中包含以下動作：

- s3:GetBucketPolicy

例如，以下政策使用這些動作，以允許主控台使用者在特定 Amazon S3 儲存貯體中模擬以資源為基礎的政策。

```
{
        "Version": "2012-10-17",
        "Statement": [
          {
```

```
            "Effect": "Allow",
            "Action": "s3:GetBucketPolicy",
            "Resource":"arn:aws:s3:::bucket-name/*"
          }
        ]
      }
```

使用政策模擬器 API 所需的許可

政策模擬器 API 作業 GetContextKeyForCustomPolicy，可SimulateCustomPolicy讓您測試尚未附加至使用者、使用者群組或角色的政策。若要測試這類政策，您可以將政策當作字串傳遞至 API。這些政策僅用於模擬，不會揭露敏感資訊。您也可以使用 API，來測試連接到 AWS 帳戶中 IAM 使用者、使用者群組或角色的政策。若要這麼做，您必須提供使用者呼叫GetContextKeyForPrincipalPolicy和的權限SimulatePrincipalPolicy。

若要檢視範例政策，允許將原則模擬器 API 用於目前的附加和未附加政策 AWS 帳戶，請參閱IAM：存取政策模擬器 API。

若要建立一個政策，只允許一個類型的政策使用政策模擬器 API，請使用下列程序。

若要允許 API 使用者去模擬直接以字串傳遞給 API 的政策

在您的政策中包含以下動作：

- `iam:GetContextKeysForCustomPolicy`
- `iam:SimulateCustomPolicy`

若要允許 API 使用者模擬連接到 IAM 使用者、使用者群組、角色、或資源的政策

在您的政策中包含以下動作：

- `iam:GetContextKeysForPrincipalPolicy`
- `iam:SimulatePrincipalPolicy`

例如，要授予名為 Bob 的使用者模擬指派給名為 Alice 的使用者的政策的許可，請授予 Bob 存取以下資源的許可：`arn:aws:iam::777788889999:user/alice`。

若要查看僅允許具有特定路徑的使用者使用政策模擬器 API 的範例政策，請參閱 IAM：根據使用者路徑存取政策模擬器 API。

使用 IAM 政策模擬器 (主控台)

根據預設，使用者可以透過在政策模擬器主控台中輸入或複製這些政策來測試尚未連接到使用者、使用者群組或角色的政策。這些政策僅用於模擬，不會揭露敏感資訊。

測試未連接至使用者、使用者群組或角色的政策 (主控台)

1. 請前往 https://policysim.aws.amazon.com/ 開啟 IAM 政策模擬器主控台。
2. 在頁面頂端的 Mode: (模式：) 選單中，選擇 New Policy (新政策)。
3. 在 Policy Sandbox (政策沙盒) 中，選擇 Create New Policy (建立新的政策)。
4. 將政策輸入或複製到政策模擬器，接著使用政策模擬器，如以下步驟所述。

在您擁有使用 IAM 政策模擬器主控台的許可之後，您可以使用政策模擬器來測試 IAM 使用者、使用者群組、角色或資源政策。

測試連接至使用者、使用者群組或角色的政策 (主控台)

1. 請前往 https://policysim.aws.amazon.com/ 開啟 IAM 政策模擬器主控台。

   > **Note**
   >
   > 若要以 IAM 使用者的身分登入政策模擬器，請使用唯一的登入 URL 來登入 AWS Management Console。然後前往 https://policysim.aws.amazon.com/。如需有關以 IAM 使用者身分登入的詳細資訊，請參閱 IAM 使用者如何登入 AWS。

   政策模擬器以 Existing Policies (現有政策) 模式開啟，並在 Users, Groups, and Roles (使用者、群組和角色) 下列出您帳戶中的 IAM 使用者。
2. 選擇適合您的任務的選項：

若要測試這個：	執行此作業：
連接至使用者的政策	在 Users, Groups, and Roles (使用者、群組和角色) 清單中選擇 Users (使用者)。然後選擇使用者。
連接至使用者群組的政策	在 Users, Groups, and Roles (使用者、群組和角色) 清單中選擇 Groups (群組)。然後選擇使用者群組。

若要測試這個：	執行此作業：
連接至角色的政策	在 Users, Groups, and Roles (使用者、群組和角色) 清單中選擇 Roles (角色)。然後選擇角色。
連接至資源的政策	請參閱 Step 9。
使用者、使用者群組或角色的自訂政策	選擇 Create new policy (建立新政策)。在新的 Policies (政策) 窗格中，輸入或貼上政策，然後選擇 Apply (套用)。

> **秘訣**
>
> 若要測試連接到使用者群組的政策，可以直接從 IAM 主控台啟動 IAM 政策模擬器：在導覽窗格中選擇 User groups (使用者群組)。選擇您想要測試政策的群組的名稱，然後選擇 Permissions (許可) 標籤。選擇 Simulate (模擬)。
> 若要測試連接到使用者的客戶受管政策：在導覽窗格中，選擇 Users (使用者)。選擇要測試政策的使用者的名稱。然後選擇 Permissions (許可) 標籤並展開您想要測試的政策。在最右側，選擇 Simulate policy (模擬政策)。IAM Policy Simulator (IAM 政策模擬器) 將在新視窗中開啟，並在 Policies (政策) 窗格中顯示選取的政策。

3. (選用) 如果您的帳戶是 AWS Organizations 中組織的成員，則請勾選 AWS Organizations SCPs 旁的核取方塊，以在您的模擬評估中包含 SCP。SCP 是一種 JSON 政策，負責指定組織或組織單位 (OU) 的最大許可。SCP 會限制成員帳戶中實體的許可。如果 SCP 封鎖服務或動作，則該帳戶中的任何實體都不能存取該服務或執行該動作。即使管理員透過 IAM 或資源政策明確授予對該服務或動作的許可，也是如此。

   如果您的帳戶不是組織的成員，則不會出現核取方塊。

4. (選用) 您可以測試設定為許可界限的政策，適用於 IAM 實體 (使用者或角色)，但不適用於使用者群組。如果目前針對實體設定了許可界限政策，則它會出現在 Policies (政策) 窗格中。您只能針對一個實體設定一個許可界限。若要測試不同的許可界限，您可以建立自訂許可界限。若要執行此操作，請選擇 Create New Policy (建立新政策)。新的 Policies (政策) 窗格隨即開啟。在選單中，選擇 Custom IAM Permissions Boundary Policy (自訂 IAM 許可界限政策)。輸入新政策的名稱，然後在下方空格中輸入或複製政策。選擇 Apply (套用) 以儲存政策。接下來，選擇 Back (上一步) 返回原始 Policies (政策) 窗格。然後，選取您要用於模擬之許可邊界旁邊的核取方塊。

5. (選用) 您只能測試連接至使用者、使用者群組或角色的政策子集。若要這樣做，請在 Policies (政策) 窗格中清除您要排除之每個政策旁邊的核取方塊。

6. 在 Policy Simulator (政策模擬器) 中，選擇 Select service (選取服務)，然後選擇該服務來測試。然後選擇 Select actions (選取動作) 並選取一或多個動作來測試。雖然選單一次只顯示一項服務的可用選項，但您選取的所有服務和動作都會出現在 Action Settings and Results (動作設定和結果) 中。

7. (選用) 如果您在 Step 2 和 Step 5 中選擇的任何政策包含帶有 AWS 全域條件索引鍵的條件，則為這些索引鍵提供值。您可以透過展開 Global Settings (全域設定) 部分並為其中顯示的金鑰名稱輸入值來完成此動作。

   > **Warning**
   >
   > 如果將條件索引鍵的值保留為空，則在模擬期間將忽略該索引鍵。在某些情況下，這會產生錯誤，並且無法執行模擬。在其他情況下，模擬會執行，但結果可能不可靠。在這些情況下，模擬不符合包含條件索引鍵或變數的真實世界條件。

8. (選用) 在您實際執行模擬之前，每個選取的動作都會顯示在 Action Settings and Results (動作設定和結果) 清單中，Not simulated (不模擬) 顯示在 Permission (許可) 欄位中。在執行模擬之前，您可以使用資源設定每個動作。若要為特定案例設定個別動作，請選擇箭頭以展開動作的資料列。如果動作支援資源級許可，則可以輸入要測試其存取權的特定資源 Amazon Resource Name (ARN)。在預設情況下，每個資源設為萬用字元 (*)。您也可以為任何條件內容金鑰指定值。如前所述，具有空白值的索引鍵被忽略，這可能會導致模擬失敗或不可靠的結果。

   a. 選擇動作名稱旁的箭頭可展開每一列，並設定在您的案例中準確模擬動作所需的任何其他資訊。如果動作需要任何資源級許可，則可以輸入要模擬存取的特定資源 Amazon Resource Name (ARN)。在預設情況下，每個資源設為萬用字元 (*)。

   b. 如果動作支援資源級許可，但不需要它們，則可以選擇 Add Resource (新增資源) 來選取要新增至模擬中的資源類型。

   c. 如果任何選取的政策包含參考此動作服務的內容索引鍵的 `Condition` 元素，則該索引鍵名稱將顯示在該動作下。您可以指定適用於模擬指定資源動作時所需使用的值。

   需要不同資源類型群組的動作

   在不同的情況下，某些動作需要不同的資源類型。每一組資源類型都與一個案例相關聯。如果其中一個適用於您的模擬，請選擇它，並且政策模擬器需要適用於該案例的資源類型。下表顯示每個支援案例選項，以及執行模擬時必須定義的資源。

   下列每個 Amazon EC2 案例都需要您指定 `instance`、`image` 和 `security-group` 資源。如果您的案例包含 EBS 磁碟區，則必須將該 `volume` 指定為資源。如果 Amazon EC2 案例包含

Virtual Private Cloud (VPC)，則必須提供 `network-interface` 資源。如果包含 IP 子網路，則必須指定 `subnet` 資源。如需有關 Amazon EC2 案例選項的詳細資訊，請參閱《Amazon C2 使用者指南》中的支援的平台。

- EC2-VPC-InstanceStore

  執行個體、影像、安全群組、網路介面

- EC2-VPC 子InstanceStore网

  執行個體、影像、安全群組、網路介面、子網路

- EC2-VPC-EBS

  執行個體、影像、安全群組、網路介面、磁碟區

- EC2-VPC-EBS-Subnet

  執行個體、影像、安全群組、網路介面、子網路、磁碟區

9. (選用) 如果要在模擬中包含以資源為基礎的政策，則必須先在 Step 6 中選擇要在該資源上模擬的動作。展開所選動作的資料列，然後使用您想要模擬的政策來輸入資源 ARN。然後選擇 ARN 文字方塊旁的 Include Resource Policy (包括資源政策)。IAM 政策模擬器目前僅支援下列服務的以資源為基礎的政策：Amazon S3 (僅限於以資源為基礎的政策；目前不支援 ACL)、Amazon SQS、Amazon SNS 和未鎖定的 S3 Glacier 文件庫 (目前不支援鎖定的文件庫)。
10. 選擇右上角的 Run Simulation (執行模擬)。

    Action Settings and Results (動作設定和結果) 每行中的 Permission (許可) 欄顯示在指定資源上模擬該動作的結果。
11. 若要查看政策中哪些陳述式明確允許或拒絕動作，請選擇 Permissions (許可) 欄中的 ***N*** matching statement(s) (相符陳述式) 連結以展開該資料列。然後選擇 Show statement (顯示陳述式) 連結。Policies (政策) 窗格顯示了相關的政策，並凸顯影響模擬結果的陳述式。

> **Note**
>
> 如果動作是隱含的拒絕 (動作因為沒有明確允許而被拒絕)，則不會顯示 List (清單) 和 Show statement (顯示陳述式) 選項。

IAM 政策模擬器主控台訊息故障診斷

下表列出使用 IAM 政策模擬器時可能遇到的資訊和警告訊息。該表也提供了解決問題的步驟。

Message	解決的步驟
已編輯此政策。變更將不會儲存到您的帳戶。	不需要採取行動。 此訊息是資訊性的。如果您在 IAM 政策模擬器中編輯現有政策，則變更不會影響您的 AWS 帳戶。政策模擬器允許您變更僅用於測試的政策。
無法取得資源政策。原因：#######	政策模擬器無法存取所請求的資源型政策。確保指定的資源 ARN 是正確的，並且執行模擬的使用者擁有讀取資源政策的許可。
一個或多個政策需要模擬設定中的值。如果沒有這些值，模擬可能會失敗。	如果您正在測試的政策包含條件金鑰或變數，但您尚未在 Simulation Settings (模擬設定) 中為這些金鑰或變數提供任何值，則會顯示此訊息。 若要關閉此訊息，請選擇 Simulation Settings (模擬設定)，然後為每個條件金鑰或變數輸入一個值。
您已變更政策。這些結果不再有效。	如果在 Results (結果) 窗格中顯示結果時變更了所選政策，則會顯示此訊息。在 Results (結果) 窗格中顯示的結果不會動態更新。 若要關閉此訊息，請再次選擇 Run Simulation (執行模擬) 以根據在 Policies (政策) 窗格中所做的變更顯示新的模擬結果。
您為此模擬輸入的資源不符合這項服務。	如果您在 Simulation Settings (模擬設定) 窗格中輸入的 Amazon Resource Name (ARN) 不符合您為目前模擬選擇的服務，則會顯示此訊息。例如，如果您為 Amazon DynamoDB 資源指定了 ARN，但您選擇了 Amazon Redshift 作為要模擬的服務，則會顯示此訊息。 若要關閉此訊息，請執行下列項目之一： • 從 Simulation Settings (模擬設定) 窗格的方塊中移除 ARN。

Message	解決的步驟
	• 選擇符合您在 Simulation Settings (模擬設定) 中所指定的 ARN 的服務。
除了以資源為基礎的政策之外，這個動作屬於支援特殊存取控制機制的服務，例如 Amazon S3 ACL 或 S3 Glacier 文件庫鎖定政策。政策模擬器不支援這些機制，因此結果可能與您的生產環境不同。	不需要採取行動。 此訊息是資訊性的。在目前版本中，政策模擬器評估連接到使用者和使用者群組的政策，並且可以評估 Amazon S3、Amazon SQS、Amazon SNS 和 S3 Glacier 的資源型政策。政策模擬器不支援其他 AWS 服務所支援的所有存取控制機制。
目前不支援 DynamoDB FGAC。	不需要採取行動。 此資訊訊息是指精細存取控制。精細存取控制可讓您使用 IAM 政策條件，決定誰可以存取 DynamoDB 資料表和索引中的個別資料項目和屬性。它也指可對這些資料表和索引執行的動作。目前版本的 IAM 政策模擬器不支援這種類型的政策條件。如需有關 DynamoDB 精細存取控制的詳細資訊，請參閱適用於 DynamoDB 的精細存取控制。
您的政策不遵守政策的語法。您可以使用政策驗證程式來檢閱對政策的建議更新。	如果您的政策不遵守 IAM 政策語法，則此訊息顯示在政策清單的上方。為了模擬這些政策，請檢閱位於 驗證 IAM 政策 的政策驗證選項來識別並修正這些政策。
此政策必須更新，以遵守最新的政策語法規則。	如果您的政策不遵守 IAM 政策語法，則會顯示此訊息。為了模擬這些政策，請檢閱位於 驗證 IAM 政策 的政策驗證選項來識別並修正這些政策。

使用 IAM 政策模擬器 (AWS CLI 和 AWS API)

政策模擬器命令通常需要呼叫 API 操作來執行兩個項目：

1. 評估政策並傳回他們所參考的內容索引鍵清單。您需要知道會參考哪些內容索引鍵，以便在下一步驟中將其提供值。
2. 模擬政策，提供在模擬過程中使用的動作、資源和內容索引鍵的清單。

基於安全考量，API 操作已分成兩組：

- 只模擬政策的API 操作會被直接以字串傳遞給 API。此套裝包括GetContextKeysForCustomPolicy和SimulateCustomPolicy。
- API 操作模擬連接到指定 IAM 使用者、使用者群組、角色或資源的政策。由於這些 API 操作可以顯示指派給其他 IAM 實體的許可的詳細資訊，因此您應該考慮限制對這些 API 操作的存取。此套裝包括GetContextKeysForPrincipalPolicy和SimulatePrincipalPolicy。如需有關限制存取 API 操作的詳細資訊，請參閱 政策範例： AWS Identity and Access Management (IAM)。

在這兩種情況下，API 操作都會在行動和資源清單上模擬一個或多個政策的效能。每個動作會與每個資源搭配使用，並且模擬會判斷政策是否允許或拒絕該資源的動作。您還可以為政策參考的任何內容鍵索引鍵提供值。您可以透過第一個呼叫 `GetContextKeysForCustomPolicy` 或 `GetContextKeysForPrincipalPolicy` 取得政策參考的內容金鑰清單。如果未提供值給內容索引鍵，則該模擬仍會執行。但是，結果可能是不可靠的，因為政策模擬器不能在評估中包含該內容索引鍵。

若要取得上下文金鑰清單 (AWS CLIAWS API)

使用以下內容評估政策清單，並傳回政策中所使用的內容索引鍵清單。

- AWS CLI：`aws iam get-context-keys-for-custom-policy` 與 `aws iam get-context-keys-for-principal-policy`
- AWS API：`GetContextKeysForCustomPolicy`和 `GetContextKeysForPrincipalPolicy`

若要模擬身分與存取權管理政策 (AWS CLIAWS API)

使用以下方法模擬 IAM 政策以判斷使用者的有效許可。

- AWS CLI：`aws iam simulate-custom-policy` 與 `aws iam simulate-principal-policy`
- AWS API：`SimulateCustomPolicy`和 `SimulatePrincipalPolicy`

新增和移除 IAM 身分許可

您可以使用政策來定義身分 (使用者、使用者群組或角色) 的許可。您可以使用、 AWS Command Line Interface (AWS CLI) 或 AWS API 為身分附加和卸離 IAM 政策 AWS Management Console，以新增和移除許可。您也可以使用相同的方法，利用政策來設定僅適用於實體 (使用者或角色) 的許可界限。權限界限是一 AWS 項進階功能，可控制實體可擁有的最大權限。

主題

術語

當您將許可政策與身分 (使用者、使用者群組和角色) 相關聯時，術語和程序會有所不同，具體取決於您使用的是受管政策或內嵌政策：

- 連接 – 與受管政策一起使用。您將受管政策連接到身分 (使用者、使用者群組或角色)。將在政策中套用許可的政策連接到身分。
- Detach (分開) – 與受管政策一起使用。您從 IAM 身分 (使用者、使用者群組或角色) 分開受管政策。分開政策會從身分中移除其許可。
- 嵌入 – 與內嵌政策一起使用。在身分中嵌入內嵌政策 (使用者、使用者群組或角色)。將在政策中套用許可的政策嵌入到身分。因為內嵌政策儲存在身分中，所以它是嵌入的而不是連接的，儘管結果是相似的。

> **Note**
>
> 您只能將服務相關角色的內嵌政策嵌入依賴該角色的服務。請參閱服務的 AWS 文件，確認是否支援此功能。

- 刪除 – 與內嵌政策一起使用。您從 IAM 身分 (使用者、使用者群組或角色) 刪除內嵌政策。刪除政策會從身分中移除其許可。

> **Note**
>
> 您只能在依賴於角色的服務中刪除服務連結角色的內嵌政策。請參閱服務的 AWS 文件，確認是否支援此功能。

您可以使用主控 AWS CLI台或 AWS API 來執行上述任何動作。

其他資訊

- 如需有關受管與內嵌政策之間的差異的詳細資訊，請參閱 受管政策與內嵌政策。
- 如需有關許可界限的詳細資訊，請參閱 IAM 實體的許可界限。
- 如需有關 IAM 政策的一般資訊，請參閱 IAM 中的政策和許可。
- 如需有關驗證 IAM 政策的資訊，請參閱 驗證 IAM 政策。
- AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

檢視身分活動

變更身分 (使用者、使用者群組或角色) 的許可之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

新增 IAM 身分許可 (主控台)

您可以使用將權限新增 AWS Management Console 至身分識別 (使用者、使用者群組或角色)。若要執行此操作，連接可控制許可的受管政策，或指定可做為許可界限的政策。您也可以嵌入內嵌政策。

將受管政策當做身分的許可政策來使用 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇政策。
3. 在政策清單中，選取要連接的政策名稱旁的選項按鈕。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 Actions (動作)，然後選擇 Attach (連接)。

5. 選取一或多個身分以將政策連接到。您可使用搜尋方塊來篩選主體實體清單。在選取身分後，選擇 Attach policy (連接政策)。

使用受管政策設定許可界限 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇要設定的政策名稱。您可以使用搜尋方塊來篩選政策清單。
4. 在政策詳細資訊頁面上，選擇連接的實體索引標籤，然後在必要時，開啟作為許可界限連接區段，然後選擇將該政策設定為許可界限。
5. 選擇一或多個要套用許可界限政策的使用者或角色。您可使用搜尋方塊來篩選主體實體清單。選取主體之後，選擇設定許可界限。

為使用者或角色嵌入內嵌政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在服務導覽窗格中，選擇 Users (使用者) 或者 Roles (角色)。
3. 在清單中，選擇要內嵌政策的使用者或角色的名稱。
4. 選擇許可索引標籤標籤。
5. 選擇新增許可，然後選擇建立內嵌政策。

> **Note**
>
> 您不能在 IAM 中的服務連結的角色中嵌入內嵌政策。由於連結服務定義您是否可以修改角色的許可，因此您可以從服務主控台、API 或 AWS CLI新增額外的政策。若要查看服務的服務連結角色文件，請參閱 AWS 與 IAM 搭配使用的服務 並在服務的 服務連結角色欄位中選擇 Yes (是)。

6. 選擇下列其中一種方法來檢視建立政策所需的步驟：

   - 匯入現有的受管政策 – 您可以將受管政策匯入帳戶內，然後編輯該政策以依據您的特定要求自訂內容。受管政策可以是 AWS 受管理策略或您先前建立的客戶管理策略。

- 使用視覺化編輯器來建立政策 – 您可以在視覺編輯工具中從零開始建構一個新的政策。若您使用視覺化編輯器，您便無需了解 JSON 語法。
- 正在使用 JSON 編輯器建立政策 – 在 JSON 編輯器選項中，您可以使用 JSON 語法來建立政策。您可以輸入新的 JSON 政策文件或者貼上範例政策。

7. 在您建立內嵌政策後，它會自動嵌入您的使用者或角色中。

為使用者群組嵌入內嵌政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 User groups (使用者群組)。
3. 在清單中，選擇要內嵌政策的使用者群組名稱。
4. 選擇 Permissions (許可) 標籤，選擇 Add permissions (新增許可)，然後選擇 Attach policy (連接政策)。
5. 執行以下任意一項：
   - 選擇視覺化選項，可建立政策。如需詳細資訊，請參閱 使用視覺化編輯器來建立政策。
   - 選擇 JSON 選項，可建立政策。如需詳細資訊，請參閱 正在使用 JSON 編輯器建立政策。
6. 當您滿意時，選擇 建立政策。

變更一或多個實體的許可界限 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇要設定的政策名稱。您可以使用搜尋方塊來篩選政策清單。
4. 在政策詳細資訊頁面上，選擇連接的實體索引標籤，然後在必要時，開啟作為許可界限連接區段。選取您想要變更其界限的使用者或角色旁的核取方塊，然後選擇變更。
5. 選取用於許可界限新政策。您可以使用搜尋方塊來篩選政策清單。在選取政策後，選擇設定許可界限。

移除 IAM 身分許可 (主控台)

您可以使用 AWS Management Console 從身分識別 (使用者、使用者群組或角色) 移除權限。若要執行此操作，請分開可控制許可的受管政策，或移除可做為許可界限的政策。您也可以刪除內嵌政策。

中斷連結當做許可政策來使用的受管政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇政策。
3. 在政策清單中，選取要分開的政策名稱旁的選項按鈕。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 Actions (動作)，然後選擇 Detach (分開)。
5. 選取要從中分開政策的身分。您可以使用搜尋方塊來篩選身分清單。在選取身分後，選擇 Detach policy (分開政策)。

移除許可界限 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇要設定的政策名稱。您可以使用搜尋方塊來篩選政策清單。
4. 在政策摘要頁面上，選擇連接的實體索引標籤，然後在必要時，開啟作為許可界限連接區段，接著選擇要從中移除許可界限的實體。然後選擇移除界限。
5. 確認您想要移除界限，然後選擇移除界限。

刪除內嵌政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在服務導覽窗格中，選擇 User groups (使用者群組)、Users (使用者) 或者 Roles (角色)。
3. 在清單中，選擇包含要刪除的政策的使用者群組、使用者、或角色的名稱。
4. 選擇 Permissions (許可) 標籤。
5. 選取政策旁的核取方塊，然後選擇移除。

6. 在確認方塊中，選擇移除。

新增 IAM 政策 (AWS CLI)

您可以使用將權限新增 AWS CLI 至身分識別 (使用者、使用者群組或角色)。若要執行此操作，連接可控制許可的受管政策，或指定可做為許可界限的政策。您也可以嵌入內嵌政策。

將受管政策當做實體的許可政策來使用 (AWS CLI)

1. (選用) 若要檢視受管政策的相關資訊，請執行下列命令：
   - 列出受管政策：aws iam list-policies
   - 取得關於受管政策的詳細資訊：get-policy
2. 若要將受管政策連接到身分 (使用者、使用者群組或角色)，請使用下列其中一項命令：
   - AWS IAM attach-user-policy
   - AWS IAM attach-group-policy
   - AWS IAM attach-role-policy

使用受管政策設定許可界限 (AWS CLI)

1. (選用) 若要檢視受管政策的相關資訊，請執行下列命令：
   - 列出受管政策：aws iam list-policies
   - 取得關於受管政策的詳細資訊：aws iam get-policy
2. 若要使用受管政策來為實體 (使用者或角色) 設定許可界限，請使用下列其中一個命令：
   - AWS IAM put-user-permissions-boundary
   - AWS IAM put-role-permissions-boundary

嵌入內嵌政策 (AWS CLI)

若要將內嵌政策嵌入到身分 (非服務連結角色的使用者、使用者群組或角色)，請使用下列命令：

- AWS IAM put-user-policy
- AWS IAM put-group-policy
- AWS IAM put-role-policy

移除 IAM 政策 (AWS CLI)

您可以使用中斷 AWS CLI 連結控制權限的受管理策略，或移除做為權限界限的策略。您也可以刪除內嵌政策。

中斷連結當做許可政策來使用的受管政策 (AWS CLI)

1. (選用) 若要檢視關於政策的資訊，請執行下列命令：
   - 列出受管政策：aws iam list-policies
   - 取得關於受管政策的詳細資訊：aws iam get-policy
2. (選用) 如果要了解的政策和身分之間的關係，請執行下列命令：
   - 若要列出受管政策所連接的身分 (使用者、使用者群組和角色)：
     - AWS IAM list-entities-for-policy
   - 若要列出連接到身分的受管政策 (使用者、使用者群組或角色)，請使用下列其中一項命令：
     - AWS IAM list-attached-user-policies
     - AWS IAM list-attached-group-policies
     - AWS IAM list-attached-role-policies
3. 若要從身分中分開受管政策 (使用者、使用者群組或角色)，請使用下列其中一項命令：
   - AWS IAM detach-user-policy
   - AWS IAM detach-group-policy
   - AWS IAM detach-role-policy

移除許可界限 (AWS CLI)

1. (選用) 若要檢視目前使用哪個受管政策來為使用者或角色設定許可界限，請執行下列命令：
   - aws iam get-user
   - aws iam get-role
2. (選用) 若要檢視目前在哪些使用者或角色使用受管政策的許可界限，請執行下列命令：
   - AWS IAM list-entities-for-policy
3. (選用) 若要檢視受管政策的相關資訊，請執行下列命令：
   - 列出受管政策：aws iam list-policies

- 取得關於受管政策的詳細資訊：aws iam get-policy

4. 若要從使用者或角色移除許可界限，請使用下列其中一個命令：

   - AWS IAM delete-user-permissions-boundary
   - AWS IAM delete-role-permissions-boundary

若要刪除內嵌政策 (AWS CLI)

1. (選用) 若要列出連接到身分 (使用者、使用者群組或角色) 的所有內嵌政策，請使用下列其中一項命令：

   - AWS IAM list-user-policies
   - AWS IAM list-group-policies
   - AWS IAM list-role-policies

2. (選用) 若要擷取嵌入到身分 (使用者、使用者群組或角色) 中的內嵌政策文件，請使用下列其中一項命令：

   - AWS IAM get-user-policy
   - AWS IAM get-group-policy
   - AWS IAM get-role-policy

3. 若要從身分中刪除內嵌政策 (非服務連結角色的使用者、使用者群組或角色)，請使用下列命令：

   - AWS IAM delete-user-policy
   - AWS IAM delete-group-policy
   - AWS IAM delete-role-policy

新增身分與存取權管理政策AWS

您可以使用 AWS API 來附加控制權限的受管理政策，或指定用作權限界限的策略。您也可以嵌入內嵌政策。

使用受管政策做為實體 (AWS API) 的權限原則

1. (選用) 若要檢視關於政策的資訊，請呼叫下列操作：

   - 若要列出受管理的策略：ListPolicies

- 如果要擷取有關受管理策略的詳細資訊：GetPolicy

2. 若要將受管政策連接到身分 (使用者、使用者群組或角色)，請呼叫下列其中一項操作：

- AttachUserPolicy
- AttachGroupPolicy
- AttachRolePolicy

使用受管政策設定權限界限 (AWS API)

1. (選用) 若要檢視受管政策的相關資訊，請呼叫下列操作：

- 若要列出受管理的策略：ListPolicies
- 如果要擷取有關受管理策略的詳細資訊：GetPolicy

2. 若要使用受管政策來為實體 (使用者或角色) 設定許可界限，請呼叫下列其中一個操作：

- PutUserPermissionsBoundary
- PutRolePermissionsBoundary

內嵌內嵌政策 (AWS API)

若要將內嵌政策嵌入在身分 (非服務連結角色的使用者、使用者群組或角色)，請呼叫下列操作：

- PutUserPolicy
- PutGroupPolicy
- PutRolePolicy

移除身分與存取權管理政策AWS

您可以使用 AWS API 中斷連結控制權限的受管理政策，或移除做為權限界限的政策。您也可以刪除內嵌政策。

卸離用作權限原則 (AWS API) 的受管理政策

1. (選用) 若要檢視關於政策的資訊，請呼叫下列操作：

- 若要列出受管理的策略：ListPolicies
- 如果要擷取有關受管理策略的詳細資訊：GetPolicy

2. (選用) 如果要了解的政策和身分之間的關係，請呼叫下列操作：
   - 若要列出受管政策所連接的身分 (使用者、使用者群組和角色)：
     - ListEntitiesForPolicy
   - 若要列出連接到身分的受管政策 (使用者、使用者群組或角色)，請呼叫下列其中一項操作：
     - ListAttachedUserPolicies
     - ListAttachedGroupPolicies
     - ListAttachedRolePolicies
3. 若要從身分中分開受管政策 (使用者、使用者群組或角色)，請呼叫下列其中一項操作：
   - DetachUserPolicy
   - DetachGroupPolicy
   - DetachRolePolicy

若要移除權限界限 (AWS API)

1. (選用) 若要檢視目前使用哪個受管政策來為使用者或角色設定許可界限，請呼叫下列操作：
   - GetUser
   - GetRole
2. (選用) 若要檢視目前在哪些使用者或角色使用受管政策的許可界限，請呼叫下列操作：
   - ListEntitiesForPolicy
3. (選用) 若要檢視受管政策的相關資訊，請呼叫下列操作：
   - 若要列出受管理的策略：ListPolicies
   - 如果要擷取有關受管理策略的詳細資訊：GetPolicy
4. 若要從使用者或角色移除許可界限，請呼叫下列其中一個操作：
   - DeleteUserPermissionsBoundary
   - DeleteRolePermissionsBoundary

若要刪除內嵌政策 (AWS API)

1. (選用) 若要列出連接到身分 (使用者、使用者群組或角色) 的所有內嵌政策，請呼叫下列其中一項操作：

- ListUserPolicies
- ListGroupPolicies
- ListRolePolicies

2. (選用) 若要擷取嵌入到身分 (使用者、使用者群組或角色) 中的內嵌政策文件，請呼叫下列其中一項操作：

- GetUserPolicy
- GetGroupPolicy
- GetRolePolicy

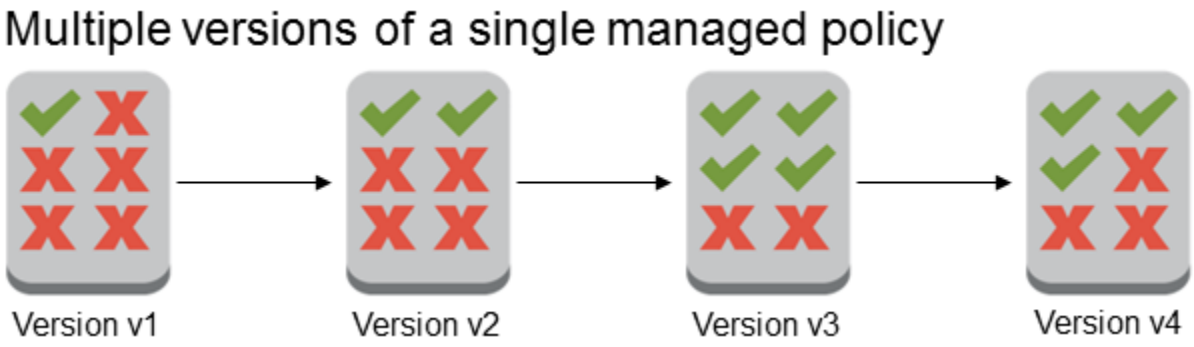
3. 若要從身分中刪除內嵌政策 (非服務連結角色的使用者、使用者群組或角色)，請呼叫下列操作：

- DeleteUserPolicy
- DeleteGroupPolicy
- DeleteRolePolicy

版本控制 IAM 政策

當您變更 IAM 客戶受管政策，以及變 AWS 更 AWS 受管政策時，變更的政策不會覆寫現有政策。IAM 反而會建立新版本的受管政策。IAM 最多可以儲存五個版本的客戶受管政策。IAM 不支援內嵌政策版本控制。

下圖說明客戶受管政策的版本控制。在此範例中，版本 1-4 會進行儲存。您最多可以儲存五個受管政策版本到 IAM。當您編輯建立第六個儲存版本的政策時，您可以選擇任一不再儲存的舊版本。您可以隨時恢復到其他四個儲存版本中的任何一個。



政策版本與 `Version` 政策元素不同。`Version` 政策元素是在政策內使用，並定義政策語言的版本。若要進一步了解 `Version` 政策元素，請參閱 IAM JSON 政策元素：Version。

您可以使用版本來追蹤受管政策的變更。例如，您可以變更受管政策，然後發現該變更有各種意外影響。在這種情況下，您可以復原至舊版的受管政策，做法是將舊版本設定為預設版本。

以下各主題說明如何使用受管政策的版本控制。

主題

用於設定預設政策版本的許可

設定預設政策版本所需的許可，對應於任務的 AWS API 操作。您可以使用 `CreatePolicyVersion` 或 `SetDefaultPolicyVersion` API 操作來設定政策的預設版本。若要讓某人設定現有政策的預設政策版本，您可以允許存取 `iam:CreatePolicyVersion` 動作或 `iam:SetDefaultPolicyVersion` 動作。這個 `iam:CreatePolicyVersion` 動作可讓他們建立新版本的政策，並設定該版本為預設值。這個 `iam:SetDefaultPolicyVersion` 動作可讓他們將任何現有的政策版本設定為預設值。

> **Important**
>
> 拒絕使用者政策中的 `iam:SetDefaultPolicyVersion` 動作不會阻止使用者建立新的政策版本，並設定為預設值。

您可以使用以下政策來拒絕使用者存取變更現有客戶的受管政策：

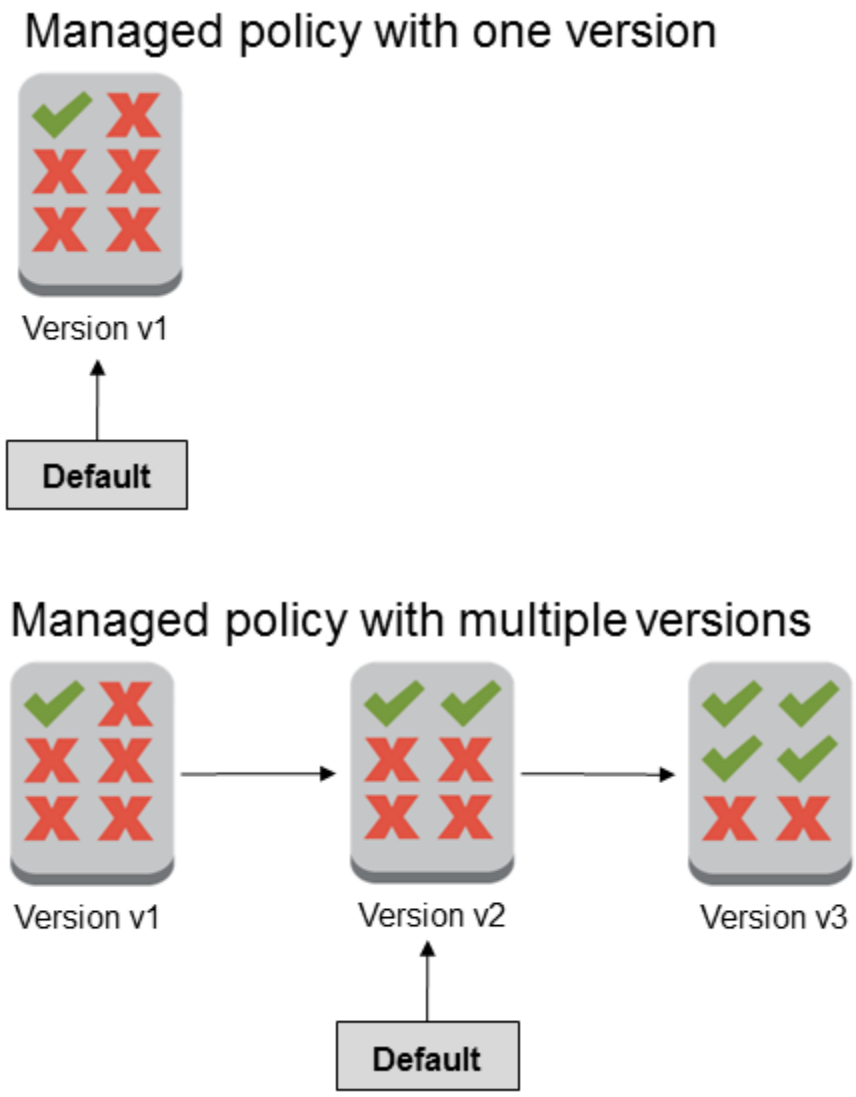
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iam:CreatePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": "arn:aws:iam::*:policy/POLICY-NAME"
        }
```

```
    ]
}
```

設定客戶受管政策的預設版本

其中一個受管政策版本是設定為預設版本。政策的預設版本是生效版本，也就是受管政策所連接的所有主體實體的 (使用者、使用者群組和角色) 的生效版本。

當您建立客戶受管政策，政策從單一版本 (識別為 v1) 開始。對於只有單一版本的受管政策，該版本會自動設定為預設值。對於具有多個版本的客戶受管政策，您可以選擇將哪個版本設定為預設值。對於 AWS 受管理的策略，預設版本由設定 AWS。下圖說明了此概念。



您可以設定預設的客戶受管政策版本，以套用該版本到政策連接的每個 IAM 身分 (使用者、使用者群組和角色)。您無法為受 AWS 管理的策略或內嵌策略設定預設版本。

設定客戶受管政策的預設版本 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇要設定預設版本的政策的政策名稱。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 Policy versions (政策版本) 標籤。選取要設定為預設版本的版本旁的核取方塊，然後選擇 Set as default (設定為預設)。

要了解如何從 AWS Command Line Interface 或 AWS API 設置客戶管理策略的默認版本，請參閱編輯客戶受管政策 (AWS CLI)。

使用版本復原變更

您可以設定客戶受管政策的預設版本來復原您的變更。例如，考量以下情境：

您建立客戶受管政策，讓使用者能夠使用 AWS Management Console管理特定 Amazon S3 儲存貯體。建立後，您的客戶受管政策只有一個版本 (識別為 v1)，所以該版本會自動設定為預設值。該政策按預期運作。

之後，您更新政策來新增許可，以管理第二個 Amazon S3 儲存貯體。IAM 建立新版本的政策 (識別為 v2)，其中包含您的變更。您設定版本 v2 為預設值，不久後您的使用者報告他們沒有獲得使用 Amazon S3 主控台的許可。在這種情況下，您可以復原到版本 v1 的政策，就是您所知按預期運作的版本。為了這樣做，您設定版本 v1 為預設版本。您的使用者現在可以使用 Amazon S3 主控台來管理原始儲存貯體。

之後，在您判斷出政策的版本 v2 錯誤後，您再次更新政策來新增許可，以管理第二個 Amazon S3 儲存貯體。IAM 建立另一個新版本的政策，識別為 v3。您設定版本 v3 為預設值，而此版本按預期運作。此時，您刪除政策的版本 v2。

版本限制

受管政策至多可有 5 個版本。如果您需要從或 AWS API 對受管政策進行五個版本以外的 AWS Command Line Interface變更，您必須先刪除一或多個現有版本。如果您使用 AWS Management Console，則在編輯策略之前不需要刪除版本。當您儲存第六個版本，會出現一個對話方塊，提示您刪除一或多個非預設的政策版本。您可以檢視每個版本的 JSON 政策文件，以協助您做決定。如需此對話方塊的詳細資訊，請參閱the section called “編輯 IAM 政策”。

您可以刪除任何想要的受管政策的版本，除了預設版本以外。當您刪除某個版本，剩餘版本的版本識別碼不會變更。因此，版本識別碼可能不會序列化。例如，若刪除受管政策的版本 v2 和 v4，並新增兩個新版本，剩餘的版本識別碼可能為 v1、v3、v5、v6 和 v7。

編輯 IAM 政策

政策為一個實體，可定義其所連接的身分或資源的許可。政策會以 JSON 文件的形 AWS 式儲存在中，並在 IAM 中以身分識別為基礎的政策附加至主體。您可以將以身分為基礎的政策連接到主體 (或身分)，例如 IAM 使用者群組、使用者或角色。以身分識別為基礎的政策包括 AWS 受管政策、客戶管理的政策和內嵌政策。您可以在 IAM 中編輯客戶受管政策和內嵌政策。 AWS 無法編輯受管理的策略。AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

主題

- 檢視政策存取
- 編輯客戶受管政策 (主控台)
- 編輯內嵌政策 (主控台)
- 編輯客戶受管政策 (AWS CLI)
- 編輯客戶管理政策 (AWS API)

檢視政策存取

變更政策的許可之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

編輯客戶受管政策 (主控台)

您可以編輯客戶受管政策以變更政策中定義的許可。客戶受管政策至多可有 5 個版本。這一點很重要，因為如果您對超過五個版本的受管理策略進行變更，系 AWS Management Console 統會提示您決定要刪除哪個版本。您還可以在編輯之前變更預設版本或或刪除政策版本，以避免出現提示。若要進一步了解版本，請參閱 版本控制 IAM 政策。

編輯客戶受管政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。

3. 在政策清單中，選擇要編輯的政策的政策名稱。您可以使用搜尋方塊來篩選政策清單。
4. 選擇許可索引標籤，然後選擇編輯。
5. 執行以下任意一項：

   - 選擇視覺化選項可變更政策，且無需了解 JSON 語法。您可以變更政策中每個許可區塊的服務、動作、資源或可選條件。您也可以匯入政策以在政策底部新增其他許可。完成變更後，選擇下一步以繼續。
   - 選擇 JSON 選項，可透過在 JSON 文字方塊中輸入或貼上文字來修改政策。您也可以匯入政策以在政策底部新增其他許可。解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。

   > **Note**
   >
   > 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

6. 在檢視與儲存頁面上，檢視此政策中定義的許可，然後選擇儲存變更以儲存工作。
7. 如果受管政策有最多五個版本，選擇儲存變更可顯示一個對話方塊。若要儲存新版本，該政策的最舊非預設版本會遭到移除，並以此新版本取代之。您也可以將新版本設定為預設的政策版本。

   選擇儲存變更，可儲存新的政策版本。

設定客戶受管政策的預設版本 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇要設定預設版本的政策的政策名稱。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 Policy versions (政策版本) 標籤。選取要設定為預設版本的版本旁的核取方塊，然後選擇 Set as default (設定為預設)。

刪除客戶受管政策的版本 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在導覽窗格中，選擇 Policies (政策)。
3. 選擇具有要刪除的版本的客戶受管政策的名稱。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 Policy versions (政策版本) 標籤。選取要刪除的版本旁的核取方塊。然後選擇 Delete (刪除)。
5. 確認您要刪除該版本，然後選擇 Delete (刪除)。

編輯內嵌政策 (主控台)

您可以從 AWS Management Console主控台中編輯內嵌政策。

為使用者、使用者群組或角色編輯內嵌政策 (主控台)

1. 在服務導覽窗格中，選擇 User (使用者)、Users groups (使用者群組) 或者 Roles (角色)。
2. 選擇您要修改的政策的使用者、 使用者群組或角色的名稱。然後選擇 Permissions (許可) 標籤並展開政策。
3. 若要編輯內嵌政策，請選擇 Edit Policy (編輯政策)。
4. 執行以下任意一項：

   - 選擇視覺化選項可變更政策，且無需了解 JSON 語法。您可以變更政策中每個許可區塊的服務、動作、資源或可選條件。您也可以匯入政策以在政策底部新增其他許可。完成變更後，選擇下一步以繼續。
   - 選擇 JSON 選項，可透過在 JSON 文字方塊中輸入或貼上文字來修改政策。您也可以匯入政策以在政策底部新增其他許可。解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。若要儲存變更而不影響目前連接的實體，請清除 Save as default version (儲存為預設版本) 的核取方塊。

   > **Note**
   >
   > 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

5. 在檢視頁面上，查看政策摘要，然後選擇儲存變更以儲存您的工作。

編輯客戶受管政策 (AWS CLI)

您可以從 () 編輯客戶管 AWS Command Line Interface AWS CLI理的策略。

> **Note**
>
> 受管政策至多可有 5 個版本。如果您需要變更五個版本以上的客戶受管政策，則必須先刪除一個或多個現有版本。

若要編輯客戶受管政策 (AWS CLI)

1. (選用) 若要檢視關於政策的資訊，請執行下列命令：
   - 列出受管政策：list-policies
   - 取得關於受管政策的詳細資訊：get-policy
2. (選用) 如果要了解的政策和身分之間的關係，請執行下列命令：
   - 若要列出受管政策所連接的身分 (使用者、使用者群組和角色)：
     - list-entities-for-policy
   - 列出連接到身分的受管政策 (使用者、使用者群組或角色)：
     - list-attached-user-policies
     - list-attached-group-policies
     - list-attached-role-policies
3. 若要編輯客戶受管政策，請執行下列命令：
   - create-policy-version
4. (選用) 若要驗證客戶受管政策，請執行下列 IAM Access Analyzer 命令：
   - validate-policy

若要設定客戶受管政策的預設版本 (AWS CLI)

1. (選用) 若要列出受管政策，請執行下列命令：
   - : list-policies
2. 若要設定客戶受管政策的預設版本，請執行下列命令：

- set-default-policy-version

若要刪除客戶受管政策的版本 (AWS CLI)

1. (選用) 若要列出受管政策，請執行下列命令：
   - : list-policies
2. 若要刪除客戶受管政策，請執行下列命令：
   - delete-policy-version

編輯客戶管理政策 (AWS API)

您可以使用 AWS API 編輯客戶管理政策。

> **Note**
>
> 受管政策至多可有 5 個版本。如果您需要變更五個版本以上的客戶受管政策，則必須先刪除一個或多個現有版本。

若要編輯客戶管理政策 (AWS API)

1. (選用) 若要檢視關於政策的資訊，請呼叫下列操作：
   - 若要列出受管理的策略：ListPolicies
   - 如果要擷取有關受管理策略的詳細資訊：GetPolicy
2. (選用) 如果要了解的政策和身分之間的關係，請呼叫下列操作：
   - 若要列出受管政策所連接的身分 (使用者、使用者群組和角色)：
     - ListEntitiesForPolicy
   - 列出連接到身分的受管政策 (使用者、使用者群組或角色)：
     - ListAttachedUserPolicies
     - ListAttachedGroupPolicies
     - ListAttachedRolePolicies
3. 若要編輯客戶受管政策，請呼叫下列操作：

- CreatePolicyVersion

4. (選用) 若要驗證客戶受管政策，請呼叫下列 IAM Access Analyzer 操作：

   - ValidatePolicy

若要設定客戶管理政策 (AWS API) 的預設版本

1. (選用) 若要列出受管政策，請呼叫下列操作：

   - ListPolicies

2. 若要設定客戶受管政策的預設版本，請呼叫下列操作：

   - SetDefaultPolicyVersion

若要刪除客戶管理政策 (AWS API) 的版本

1. (選用) 若要列出受管政策，請呼叫下列操作：

   - ListPolicies

2. 若要刪除客戶受管政策，請呼叫下列操作：

   - DeletePolicyVersion

刪除 IAM 政策

您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 IAM API 刪除身分與存取權管理政策。

> **Note**
>
> 刪除 IAM 政策的動作具有永久性。刪除政策後，將無法復原。

如需有關受管與內嵌政策之間的差異的詳細資訊，請參閱 受管政策與內嵌政策。

如需有關 IAM 政策的一般資訊，請參閱 IAM 中的政策和許可。

AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

主題

檢視政策存取

刪除政策之前，您應該檢閱其最近的服務層級活動。這很重要，因為您不希望從正在使用該許可的主體 (人員或應用程式) 中移除存取。如需有關檢視上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

刪除 IAM 政策 (主控台)

您可以從 AWS 帳戶移除客戶管理政策。您無法刪除 AWS 受管理的策略。

刪除客戶受管政策 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇政策。
3. 選擇您要刪除的客戶管理政策旁的選項按鈕。您可以使用搜尋方塊來篩選政策清單。
4. 選擇 動作，然後選擇 刪除。
5. 遵循說明以確認您要刪除政策，然後選擇刪除。

為使用者群組群組、使用者或角色刪除內嵌政策 (主控台)

1. 在服務導覽窗格中，選擇 User groups (使用者群組)、Users (使用者) 或者 Roles (角色)。
2. 選擇您要透過政策來刪除的使用者群組、使用者、或角色的名稱。然後選擇 許可 標籤。
3. 選取您要刪除的政策旁的核取方塊，然後選擇移除。若要刪除使用者或角色中的內嵌原則，請選擇移除以確認刪除。如果您要刪除 使用者群組 中的單一內嵌政策，請輸入政策的名稱，然後選擇 刪除 。如果您要刪除 使用者群組 中的多項內嵌政策，請輸入您要刪除的政策數量，後接 **`inline policies`**，然後選擇 刪除 。例如，如果您要刪除三個內嵌政策，請輸入 **`3 inline policies`**。

刪除 IAM 政策 (AWS CLI)

您可以從 AWS Command Line Interface刪除客戶受管政策。

刪除客戶受管政策 (AWS CLI)

1. (選用) 若要檢視關於政策的資訊，請執行下列命令：
   - 列出受管政策：list-policies
   - 取得關於受管政策的詳細資訊：get-policy
2. (選用) 如果要了解的政策和身分之間的關係，請執行下列命令：
   - 若要列出受管政策所連接的身分 (使用者、使用者群組和角色)，請執行以下命令：
     - list-entities-for-policy
   - 若要列出連接到身分 (使用者、使用者群組或角色) 的受管政策，請執行下列其中一項命令：
     - list-attached-user-policies
     - list-attached-group-policies
     - list-attached-role-policies
3. 若要刪除客戶受管政策，請執行下列命令：
   - : delete-policy

若要刪除內嵌政策 (AWS CLI)

1. (選用) 若要列出連接到身分 (使用者、使用者群組或角色) 的所有內嵌政策，請使用下列其中一項命令：
   - AWS IAM list-user-policies
   - AWS IAM list-group-policies
   - AWS IAM list-role-policies
2. (選用) 若要擷取嵌入到身分 (使用者、使用者群組或角色) 中的內嵌政策文件，請使用下列其中一項命令：
   - AWS IAM get-user-policy
   - AWS IAM get-group-policy
   - AWS IAM get-role-policy

3. 若要從身分中刪除內嵌政策 (非服務連結角色的使用者、使用者群組或角色)，請使用下列命令：

   - AWS IAM delete-user-policy
   - AWS IAM delete-group-policy
   - AWS IAM delete-role-policy

刪除身分與存取權管理政策AWS

您可以使用 AWS API 刪除客戶管理政策。

若要刪除客戶管理政策 (AWS API)

1. (選用) 若要檢視關於政策的資訊，請呼叫下列操作：

   - 若要列出受管理的策略：ListPolicies
   - 如果要擷取有關受管理策略的詳細資訊：GetPolicy

2. (選用) 如果要了解的政策和身分之間的關係，請呼叫下列操作：

   - 若要列出受管政策所連接的身分 (使用者、使用者群組和角色)，請呼叫以下操作：
     - ListEntitiesForPolicy
   - 若要列出連接到身分的受管政策 (使用者、使用者群組或角色)，請呼叫下列其中一項操作：
     - ListAttachedUserPolicies
     - ListAttachedGroupPolicies
     - ListAttachedRolePolicies

3. 若要刪除客戶受管政策，請呼叫下列操作：

   - DeletePolicy

若要刪除內嵌政策 (AWS API)

1. (選用) 若要列出連接到身分 (使用者、使用者群組或角色) 的所有內嵌政策，請呼叫下列其中一項操作：

   - ListUserPolicies
   - ListGroupPolicies
   - ListRolePolicies

2. (選用) 若要擷取嵌入到身分 (使用者、使用者群組或角色) 中的內嵌政策文件，請呼叫下列其中一項操作：

   - GetUserPolicy
   - GetGroupPolicy
   - GetRolePolicy

3. 若要從身分中刪除內嵌政策 (非服務連結角色的使用者、使用者群組或角色)，請呼叫下列操作：

   - DeleteUserPolicy
   - DeleteGroupPolicy
   - DeleteRolePolicy

AWS 使用上次存取的資訊精簡權限

身為管理員，您可能會為 IAM 資源 (角色、使用者、使用者群組或政策) 授予超出其所需範圍的許可。IAM 會提供上次存取的資訊，協助您識別未使用的許可，以便您移除這些許可。您可以使用上次存取的資訊來精細化政策，並僅允許存取 IAM 身分和政策所使用的服務和動作。這有助於您更加符合最低許可的最佳實務。您可以檢視 IAM 或 AWS Organizations中存在的身分或政策上次存取的資訊。

您可以透過未使用的存取權分析器持續監控上次存取的資訊。如需詳細資訊，請參閱外部和未使用的存取權調查結果。

主題

- IAM 上次存取的資訊類型
- AWS Organizations上次存取的資訊
- 關於上次存取資訊的注意事項
- 必要許可
- IAM 和 Organizations 實體的疑難排解活動
- AWS 追蹤上次存取資訊的位置
- 檢視 IAM 上次存取的資訊
- 檢視 Organizations 上次存取的資訊
- 使用上次存取資訊的範例案例
- IAM 動作最近存取的資訊服務和動作

IAM 上次存取的資訊類型

您可以檢視 IAM 身分上次存取的兩種類型資訊：允許的 AWS 服務資訊和允許的動作資訊。該信息包括嘗試訪問 AWS API 的日期和時間。對於動作，上次存取的資訊會報告服務管理動作。管理動作包括建立、刪除和修改動作。若要進一步了解如何檢視上次存取 IAM 的資訊，請參閱 檢視 IAM 上次存取的資訊。

如需有關使用上次存取的資訊以決定您授予 IAM 身分許可的案例，請參閱 使用上次存取資訊的範例案例。

若要深入瞭解如何提供管理動作資訊，請參閱關於上次存取資訊的注意事項。

AWS Organizations上次存取的資訊

如果您使用管理帳戶認證登入，則可以檢視組織中 AWS Organizations 實體或策略上次存取的服務資訊。 AWS Organizations 實體包括組織根目錄、組織單位 (OU) 或帳戶。上次存取的資訊 AWS Organizations 包括服務控制原則 (SCP) 所允許之服務的相關資訊。這些資訊會指出組織或帳戶中哪些主體 (根使用者、IAM 使用者或角色) 上次嘗試存取服務，以及何時存取服務。若要進一步瞭解報告以及如何檢視的上次存取資訊 AWS Organizations，請參閱檢視 Organizations 上次存取的資訊。

如需有關使用上次存取的資訊以決定您授予 Organizations 實體許可的案例，請參閱 使用上次存取資訊的範例案例。

關於上次存取資訊的注意事項

在您使用報告中上次存取資訊以變更 IAM 身分或 Organizations 實體的許可之前，請檢閱以下有關資訊的詳細資訊。

- 追蹤期 – 最近的活動會在四小時內顯示在 IAM 主控台中。服務資訊的追蹤期至少為 400 天，具體視服務何時開始追蹤動作資訊而定。Amazon S3 動作資訊的追蹤期從 2020 年 4 月 12 日開始。Amazon EC2、IAM 和 Lambda 行動追蹤期由 2021 年 4 月 7 日開始。所有其他服務的追蹤期從 2023 年 5 月 23 日開始。如需檢視可使用動作上次存取資訊的服務清單，請參閱 IAM 動作最近存取的資訊服務和動作。如需有關哪些區域提供動作上次存取資訊的更多資訊，請參閱 AWS 追蹤上次存取資訊的位置。
- 報告的嘗試次數 — 上次存取的服務資料包括存取 AWS API 的所有嘗試，而不僅僅是成功的嘗試次數。這包括使用 AWS Management Console、透過任何 SDK 的 AWS API 或任何命令列工具進行的所有嘗試。在上次存取的服務相關資料中看到未預期的項目並不表示您的帳戶資訊洩露，因為請求可能已遭拒。有關所有 API 調用以及它們是成功還是拒絕訪問的信息，請參閱您的 CloudTrail 日誌作為授權來源。

- PassRole— 不會追蹤`iam:PassRole`動作，也不會包含在 IAM 動作上次存取的資訊中。
- 動作上次存取的資訊 – 動作上次存取的資訊適用於由 IAM 身分存取的服務管理動作。檢視動作上次存取的報告資訊的服務及其動作清單。

> **Note**
>
> 動作上次存取的資訊無法用於 Amazon S3 資料事件。

- 管理事件 — IAM 為記錄的服務管理事件提供動作資訊 CloudTrail。有時候， CloudTrail管理事件也稱為控制平面作業或控制平面事件。管理事件可讓您檢視對中的資源執行的管理作業 AWS 帳戶。若要進一步了解中的管理事件 CloudTrail，請參閱AWS CloudTrail 使用者指南中的記錄管理事件。
- 報告擁有者 – 只有產生報告的主體才可以檢視報告詳細資訊。這表示當您檢視中的資訊時 AWS Management Console，您可能必須等待資訊產生並載入。如果您使用 AWS CLI 或 AWS API 取得報表詳細資料，您的認證必須與產生報表之主體的認證相符。如果您使用角色或聯合身分使用者的暫時性憑證，您必須在相同的工作階段期間產生和擷取報告。如需有關擔任角色工作階段主體的詳細資訊，請參閱 AWS 政策元素：Principal。
- IAM 資源 – IAM 上次存取的資訊，包括您的帳戶中的 IAM 資源 (角色、使用者、使用者群組和政策)。Organizations 上次存取的資訊包括指定組 Organizations 實體中的主體 (IAM 使用者、IAM 角色或 AWS 帳戶根使用者)。上次存取的資訊不包括未驗證的嘗試。
- IAM 政策類型 – IAM 上次存取的資訊包含由 IAM 身分的政策所許可的服務。這些政策連接至角色或者直接或透過群組連接至使用者。您的報告不包含其他政策類型允許的存取。排除的政策類型包括以資源為基礎的政策、存取控制清單、 AWS Organizations SCP、IAM 許可邊界，以及工作階段政策。服務連結角色所提供的許可是由它們連結的服務所定義，而且無法在 IAM 中修改。若要進一步了解服務連結角色，請參閱使用服務連結角色 若要了解如何評估不同原則類型以允許或拒絕存取，請參閱政策評估邏輯。
- 政策類型 – AWS Organizations 的資訊僅包含由 Organizations 實體的繼承服務控制政策 (SCP) 所許可的服務。SCP 是連接到根帳戶、OU 或帳戶的政策。您的報告不包含其他政策類型允許的存取。排除的政策類型包含身分類型政策、資源類型政策、存取控制清單、IAM 許可邊界，以及工作階段政策。若要了解如何評估不同的政策類型以允許或拒絕存取，請參閱政策評估邏輯。
- 指定策略 ID — 當您使用 AWS CLI 或 AWS API 產生 Organizations 中上次存取資訊的報告時，您可以選擇性地指定策略 ID。產生的報告包含僅由該政策允許的服務資訊。該資訊包含指定 Organizations 實體或實體子系中最新的帳戶活動。如需詳細資訊，請參閱 aws iam generate-organizations-access-report 或 GenerateOrganizationsAccessReport.
- Organizations 管理帳戶 – 您必須登入組織的管理帳戶，才能檢視上次存取的資訊。您可以選擇使用 IAM 主控台、或 AWS API 來檢視管理帳戶的資訊。 AWS CLI產生的報告會列出所有 AWS 服務，因為管理帳戶不受 SCP 的限制。如果您指定的政策 ID 位於 CLI 或 API 中，便會忽略該政策。對於每

個服務，報告包含僅適用於管理帳戶的資訊。不過，其他 Organizations 實體的報告不會傳回管理帳戶中的活動資訊。

- Organizations 設定 – 管理員必須先在組織根目錄中啟用 SCP，您才能產生 Organizations 的資料。

必要許可

若要檢視中上次存取的資訊 AWS Management Console，您必須擁有授與必要權限的策略。

IAM 資訊的許可

若要使用 IAM 主控台檢視 IAM 使用者、角色或政策的上次存取資訊，您必須擁有一個包含下列動作的政策：

- `iam:GenerateServiceLastAccessedDetails`
- `iam:Get*`
- `iam:List*`

這些許可允許使用者檢視下列項目：

- 哪些使用者、群組或角色連接至受管政策
- 使用者或角色可以存取哪些服務
- 他們上一次存取該服務的時間
- 他們上次嘗試使用特定的 Amazon EC2、IAM、Lambda 或 Amazon S3 動作的時間

若要使用 AWS CLI 或 AWS API 檢視 IAM 上次存取的資訊，您必須擁有符合您要使用之作業的許可：

- `iam:GenerateServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetailsWithEntities`
- `iam:ListPoliciesGrantingServiceAccess`

此範例會示範如何建立身分型政策，允許檢視 IAM 上次存取的資訊。此外，還允許對所有 IAM 的唯讀存取。此政策定義了程式設計和主控台存取的許可。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateServiceLastAccessedDetails",
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    }
```

AWS Organizations 資訊的許可

若要使用 IAM 主控台檢視 Organizations 中的根目錄、OU 或帳戶實體，您必須擁有一個包含下列動作的政策：

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribeAccount
- organizations:DescribeOrganization
- organizations:DescribeOrganizationalUnit
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents
- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

若要使用 AWS CLI 或 AWS API 來檢視「Organizations」上次存取的服務資訊，您必須具有包含下列動作的策略：

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribePolicy
- organizations:ListChildren

- organizations:ListParents
- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

此範例會示範如何建立身分型政策，允許檢視 Organizations 的服務上次存取資訊。此外，還允許對所有 Organizations 的唯讀存取。此政策定義了程式設計和主控台存取的許可。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateOrganizationsAccessReport",
            "iam:GetOrganizationsAccessReport",
            "organizations:Describe*",
            "organizations:List*"
        ],
        "Resource": "*"
    }
}
```

您也可以使用 iam: OrganizationsPolicyId 條件金鑰，僅允許針對特定 Organizations 政策產生報告。如需政策範例，請參閱 IAM：檢視 Organizations 政策上次存取的服務資訊。

IAM 和 Organizations 實體的疑難排解活動

在某些情況下，您 AWS Management Console 上次存取的資訊表格可能是空的。或者，您的 AWS CLI 或 AWS API 請求可能返回一組空信息或空字段。在這些情況下，檢閱下列問題：

- 對於上次存取的動作資訊，清單中可能不會傳回您預期看到的動作。這可能是因為 IAM 身分沒有動作的許可，或者尚 AWS 未追蹤上次存取資訊的動作。
- 對於 IAM 使用者，請確定該使用者至少連接一個內嵌或受管政策，無論是直接或透過群組成員資格連接。
- 對於 IAM 群組，請確認群組至少連接一個內嵌或受管政策。
- 對於 IAM 群組，報告只會傳回使用群組的政策存取服務之成員的服務上次存取資訊。若要了解成員是否使用了其他政策，請檢閱該使用者的上次存取資訊。
- 對於 IAM 角色，請確認角色至少連接一個內嵌或受管政策。

- 對於 IAM 實體 (使用者或角色)，請檢閱可能影響該實體之許可的其他政策類型。其中包括以資源為基礎的政策、存取控制清單、 AWS Organizations 政策、IAM 許可界限或工作階段政策。如需詳細資訊，請參閱政策類型或運用單一帳戶評估政策。
- 對於 IAM 政策，請確定指定的受管政策已連接到至少一個使用者、內含成員的群組，或角色。
- 對於 Organizations 實體 (根目錄、OU 或帳戶)，確定您使用 Organizations 管理帳戶憑證進行登入。
- 確定已在組織根目錄中啟用 SCP。
- 動作上次存取的資訊僅適用於 IAM 動作最近存取的資訊服務和動作 中列出的動作。

當您進行變更時，請等待至少 4 小時，活動才會顯示在您的 IAM 主控台報告中。如果您使用 AWS CLI 或 AWS API，則必須產生新報告才能檢視更新的資訊。

AWS 追蹤上次存取資訊的位置

AWS 收集標準 AWS 區域的上次存取資訊。新 AWS 增其他區域時，這些區域會新增至下表，包括 AWS 開始追蹤每個區域資訊的日期。

- 服務資訊：服務追蹤期至少為 400 天，如果區域在過去 400 天內開始追蹤此功能，則服務追蹤期會更短。
- 動作資訊 – Amazon S3 管理動作的追蹤期從 2020 年 4 月 12 日開始。Amazon EC2、IAM 和 Lambda 管理行動追蹤期由 2021 年 4 月 7 日開始。所有其他服務的管理動作追蹤期從 2023 年 5 月 23 日開始。如果某區域的追蹤日期遲於 2023 年 5 月 23 日，則該區域的動作上次存取資訊將從較遲的日期開始。

區域名稱	區域	追蹤開始日期
美國東部 (俄亥俄)	us-east-2	2017 年 10 月 27 日
美國東部 (維吉尼亞北部)	us-east-1	2015 年 10 月 1 日
美國西部 (加利佛尼亞北部)	us-west-1	2015 年 10 月 1 日
美國西部 (奧勒岡)	us-west-2	2015 年 10 月 1 日
非洲 (開普敦)	af-south-1	2020 年 4 月 22 日
亞太區域 (香港)	ap-east-1	2019 年 4 月 24 日

區域名稱	區域	追蹤開始日期
亞太區域 (海德拉巴)	ap-south-2	2022 年 11 月 22 日
亞太區域 (雅加達)	ap-southeast-3	2021 年 12 月 13 日
亞太區域 (墨爾本)	ap-southeast-4	2023 年 1 月 23 日
亞太區域 (孟買)	ap-south-1	2016 年 6 月 27 日
亞太區域 (大阪)	ap-northeast-3	2018 年 2 月 11 日
亞太區域 (首爾)	ap-northeast-2	2016 年 1 月 6 日
亞太區域 (新加坡)	ap-southeast-1	2015 年 10 月 1 日
亞太區域 (雪梨)	ap-southeast-2	2015 年 10 月 1 日
亞太區域 (東京)	ap-northeast-1	2015 年 10 月 1 日
加拿大 (中部)	ca-central-1	2017 年 10 月 28 日
歐洲 (法蘭克福)	eu-central-1	2015 年 10 月 1 日
歐洲 (愛爾蘭)	eu-west-1	2015 年 10 月 1 日
歐洲 (倫敦)	eu-west-2	2017 年 10 月 28 日
歐洲 (米蘭)	eu-south-1	2020 年 4 月 28 日
Europe (Paris)	eu-west-3	2017 年 12 月 18 日
歐洲 (西班牙)	eu-south-2	2022 年 11 月 15 日
歐洲 (斯德哥爾摩)	eu-north-1	2018 年 12 月 12 日
歐洲 (蘇黎世)	eu-central-2	2022 年 11 月 8 日
以色列 (特拉維夫)	il-central-1	2023 年 8 月 1 日
中東 (巴林)	me-south-1	2019 年 7 月 29 日

區域名稱	區域	追蹤開始日期
中東 (阿拉伯聯合大公國)	me-central-1	2022 年 8 月 30 日
南美洲 (聖保羅)	sa-east-1	2015 年 12 月 11 日
AWS GovCloud (美國東部)	us-gov-east-1	2023 年 7 月 1 日
AWS GovCloud (美國西部)	us-gov-west-1	2023 年 7 月 1 日

如果某個區域未在上表中列出，則表示此區域尚不提供上次存取的相關資訊。

地 AWS 區是地理區域中 AWS 資源的集合。區域會分組成分割區。標準區域是屬於該 aws 分區的區域。如需有關不同分割區的詳細資訊，請參閱 AWS 一般參考中的 Amazon Resource Name (ARN) 格式。如需有關「區域」的詳細資訊，請參閱中的「關於 AWS 區域」 AWS 一般參考。

檢視 IAM 上次存取的資訊

您可以使用 AWS Management Console、 AWS CLI或 AWS API 檢視 IAM 上次存取的資訊。檢視顯示上次存取資訊的服務及其動作清單。如需有關上次存取的資訊的詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

您可以在 IAM 中檢視下列資源類型的資訊。在每個案例中，資訊涵蓋指定報告期間允許的服務：

- 使用者 – 檢視使用者上一次嘗試存取每個允許服務的時間。
- 使用者群組 – 檢視有關上一次使用者群組成員嘗試存取每個允許服務的資訊。此報告也包含曾嘗試存取的成員總數。
- 角色 – 檢視上一次有人使用該角色嘗試存取每個允許服務的時間。
- 政策 – 檢視有關上一次使用者或角色嘗試存取每個允許服務的資訊。此報告也包含曾嘗試存取的實體總數。

> **Note**
>
> 在檢視 IAM 中的資源的存取資訊之前，請確定您了解報告期間、報告的實體，以及您資訊的評估政策類型。如需詳細資訊，請參閱 the section called “關於上次存取資訊的注意事項”。

檢視 IAM 的資訊 (主控台)

您可以在 IAM 主控台的 Access Advisor (存取建議程式) 索引標籤上檢視 IAM 上次存取的資訊。

檢視 IAM 的資訊 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 User Groups (使用者群組)、Users (使用者)、Roles (角色) 或 Policies (政策)。
3. 選擇任意使用者、使用者群組、角色或政策名稱以開啟其 Summary (摘要) 頁面，然後選擇 Access Advisor (存取顧問) 標籤。根據您選擇的資源，檢視下列資訊：

   - 使用者群組 – 檢視使用者群組成員可存取的服務清單。您也可以檢視成員上次存取服務的時間、他們使用的使用者群組政策，以及提出要求的使用者群組成員。選擇政策的名稱以了解其為受管政策或內嵌使用者群組政策。選擇使用者群組成員的名稱以檢視該使用者群組的所有成員，以及他們上次存取該服務的時間。
   - 使用者 – 檢視使用者可存取的服務清單。您也可以檢視他們上次存取服務的時間，以及目前與該名使用者關聯的政策有哪些。選擇政策的名稱，以瞭解該政策是受管理政策、內嵌使用者政策還是使用者群組的內嵌政策。
   - 角色 – 檢視角色可存取的服務清單、角色上次存取該服務的時間，以及他們使用哪些政策。選擇政策的名稱以了解其為受管政策或內嵌角色政策。
   - 政策 – 檢視政策中包含允許動作的服務清單。您也可以檢視上次使用政策來存取服務的時間，以及使用政策的實體 (使用者或角色)。上次存取日期也包含透過其他政策授予此政策存取權的時間。選擇實體的名稱以了解哪些實體有連接政策，以及它們上次存取服務的時間。
4. 在表格的服務欄中，選擇其中一項包含動作上次存取資訊的服務之名稱，以檢視 IAM 實體嘗試存取的管理動作清單。您可以檢視 AWS 區域 和時間戳記，顯示上次有人嘗試執行動作的時間。
5. 針對包含動作上次存取資訊的服務，將為其服務和管理動作顯示上次存取欄。檢閱此欄中傳回的下列可能結果。這些結果會根據是否允許服務或動作、存取，以及是否追蹤上次存取的資訊而有所不同。 AWS

   <number of> 天前

   自追蹤期間內使用服務或動作之後的天數。服務的追蹤期為過去 400 天。Amazon S3 動作的追蹤期由 2020 年 4 月 12 日開始。Amazon EC2、IAM 和 Lambda 行動追蹤期由 2021 年 4 月 7 日開始。所有其他服務的追蹤期從 2023 年 5 月 23 日開始。若要深入瞭解每個項目的追蹤開始日期 AWS 區域，請參閱AWS 追蹤上次存取資訊的位置。

在追蹤期內未存取

追蹤的服務或動作尚未由實體在追蹤期間內使用。

您可以擁有清單中未出現的動作許可。如果 AWS目前不包含動作的追蹤資訊，就可能發生這種情況。您不應該只根據有沒有追蹤資訊來決定許可。相反地，我們建議您使用此資訊來告知並支援授予最低權限的整體策略。檢查您的政策以確認存取層級是否適當。

檢視 IAM 的資訊 (AWS CLI)

您可以使用擷 AWS CLI 取有關上次使用 IAM 資源嘗試存取 AWS 服務和 Amazon S3、Amazon EC2、IAM 和 Lambda 動作的相關資訊。IAM 資源可以是使用者、使用者群組、角色或政策。

檢視 IAM 的資訊 (AWS CLI)

1. 產生報告。此請求必須包含您需要報告的 IAM 資源 (使用者、使用者群組、角色或政策) 的 ARN。您可以在報告中指定要產生的資料粒度層級，以檢視任一服務或同時檢視服務和動作的存取詳細資訊。它會傳回 `job-id`，然後您可將它用於 `get-service-last-accessed-details` 和 `get-service-last-accessed-details-with-entities` 操作以監控 `job-status`，直到任務完成。

   - AWS 我 generate-service-last-accessed的詳細信息

2. 使用上個步驟的 `job-id` 參數來擷取報告的詳細資訊。

   - AWS 我 get-service-last-accessed的詳細信息

   根據您在 `generate-service-last-accessed-details` 操作中請求的資源類型與精細程度，此操作會傳回以下資訊：

   - 使用者 – 傳回指定的使用者可存取的服務清單。對於每個服務，此操作會傳回使用者最後一次嘗試的日期與時間，以及該使用者的 ARN。
   - 使用者群組 – 傳回指定使用者群組的成員可使用連接至該使用者群組的政策進行存取的服務清單。對於每個服務，此操作會傳回任何使用者群組成員最後一次嘗試的日期與時間。它也會傳回該使用者的 ARN 以及曾嘗試存取服務的使用者群組成員總數。使用此GetServiceLastAccessedDetailsWithEntities作業擷取所有成員的清單。
   - 角色 – 傳回指定的角色可存取的服務清單。對於每個服務，此操作會傳回角色最後一次嘗試的日期與時間，以及該角色的 ARN。

- 政策 – 傳回指定的政策允許存取的服務清單。對於每個服務，此操作會傳回實體 (使用者或角色) 上次使用政策來嘗試存取服務的日期和時間。它也會傳回實體的 ARN 以及嘗試存取的實體總數。

3. 進一步了解在嘗試存取特定服務時使用使用者群組或政策許可的實體。此操作會傳回實體清單，包括各實體的 ARN、ID、名稱、路徑、類型 (使用者或角色)，以及它們最後一次嘗試存取服務的時間。您也可以針對使用者和角色使用此操作，但只會傳回有關該實體的資訊。

   - AWS IAM get-service-last-accessed-details-with-entities

4. 進一步了解有關身分 (使用者、使用者群組或角色) 在嘗試存取特定服務時使用之以身分為基礎的政策。當您指定身分和服務時，此操作會傳回實體可用於存取指定服務的許可政策清單。此操作可提供政策的目前狀態，而且不倚賴產生的報告。它也不會傳回其他政策類型，例如以資源為基礎的政策、存取控制清單、 AWS Organizations 政策、IAM 許可邊界，或工作階段政策。如需詳細資訊，請參閱政策類型或運用單一帳戶評估政策。

   - AWS IAM 訪 list-policies-granting-service問

檢視與存取權管理 (AWS API) 的資訊

您可以使用 AWS API 擷取有關上次使用 IAM 資源嘗試存取 AWS 服務和 Amazon S3、Amazon EC2、IAM 和 Lambda 動作的相關資訊。IAM 資源可以是使用者、使用者群組、角色或政策。您可以在報告中指定要產生的資料細微層級，以檢視任一服務或同時檢視服務和動作的詳細資訊。

若要檢視與存取權管理 (AWS API) 的資訊

1. 產生報告。此請求必須包含您需要報告的 IAM 資源 (使用者、使用者群組、角色或政策) 的 ARN。它會傳回 `JobId`，然後您可將它用於 `GetServiceLastAccessedDetails` 和 `GetServiceLastAccessedDetailsWithEntities` 操作以監控 `JobStatus`，直到任務完成。

   - GenerateServiceLastAccessedDetails

2. 使用上個步驟的 `JobId` 參數來擷取報告的詳細資訊。

   - GetServiceLastAccessedDetails

   根據您在 `GenerateServiceLastAccessedDetails` 操作中請求的資源類型與精細程度，此操作會傳回以下資訊：

- 使用者 – 傳回指定的使用者可存取的服務清單。對於每個服務，此操作會傳回使用者最後一次嘗試的日期與時間，以及該使用者的 ARN。
- 使用者群組 – 傳回指定使用者群組的成員可使用連接至該使用者群組的政策進行存取的服務清單。對於每個服務，此操作會傳回任何使用者群組成員最後一次嘗試的日期與時間。它也會傳回該使用者的 ARN 以及曾嘗試存取服務的使用者群組成員總數。使用此GetServiceLastAccessedDetailsWithEntities作業擷取所有成員的清單。
- 角色 – 傳回指定的角色可存取的服務清單。對於每個服務，此操作會傳回角色最後一次嘗試的日期與時間，以及該角色的 ARN。
- 政策 – 傳回指定的政策允許存取的服務清單。對於每個服務，此操作會傳回實體 (使用者或角色) 上次使用政策來嘗試存取服務的日期和時間。它也會傳回實體的 ARN 以及嘗試存取的實體總數。

3. 進一步了解在嘗試存取特定服務時使用使用者群組或政策許可的實體。此操作會傳回實體清單，包括各實體的 ARN、ID、名稱、路徑、類型 (使用者或角色)，以及它們最後一次嘗試存取服務的時間。您也可以針對使用者和角色使用此操作，但只會傳回有關該實體的資訊。

   - GetServiceLastAccessedDetailsWithEntities

4. 進一步了解有關身分 (使用者、使用者群組或角色) 在嘗試存取特定服務時使用之以身分為基礎的政策。當您指定身分和服務時，此操作會傳回實體可用於存取指定服務的許可政策清單。此操作可提供政策的目前狀態，而且不倚賴產生的報告。它也不會傳回其他政策類型，例如以資源為基礎的政策、存取控制清單、 AWS Organizations 政策、IAM 許可邊界，或工作階段政策。如需詳細資訊，請參閱政策類型或運用單一帳戶評估政策。

   - ListPoliciesGrantingServiceAccess

檢視 Organizations 上次存取的資訊

您可以 AWS Organizations 使用 IAM 主控台或 AWS API 檢視上次存取的服務資訊。 AWS CLI如需關於資料、必要許可、疑難排解及支援區域的重要資訊，請參閱AWS 使用上次存取的資訊精簡權限。

使用 AWS Organizations 管理帳戶登入資料登入 IAM 主控台時，您可以檢視組織中任何實體的資訊。Organizations 實體包含組織根目錄、組織單位 (OU) 和帳戶。您也可以使用 IAM 主控台來檢視組織中任何服務控制政策 (SCP) 的資訊。IAM 會顯示套用至實體的 SCP 所允許的服務清單。對於每個服務，您可以檢視針對所選擇 Organizations 實體或是該實體的子系的最近帳戶活動資訊。

當您搭配管理帳戶認證使用 AWS CLI 或 AWS API 時，您可以為組織中的任何實體或政策產生報告。實體的程式設計報告包含套用至實體之任何 SCP 所允許的服務清單。對於每個服務，報告包含指定 Organizations 實體或實體子目錄中的最新帳戶活動。

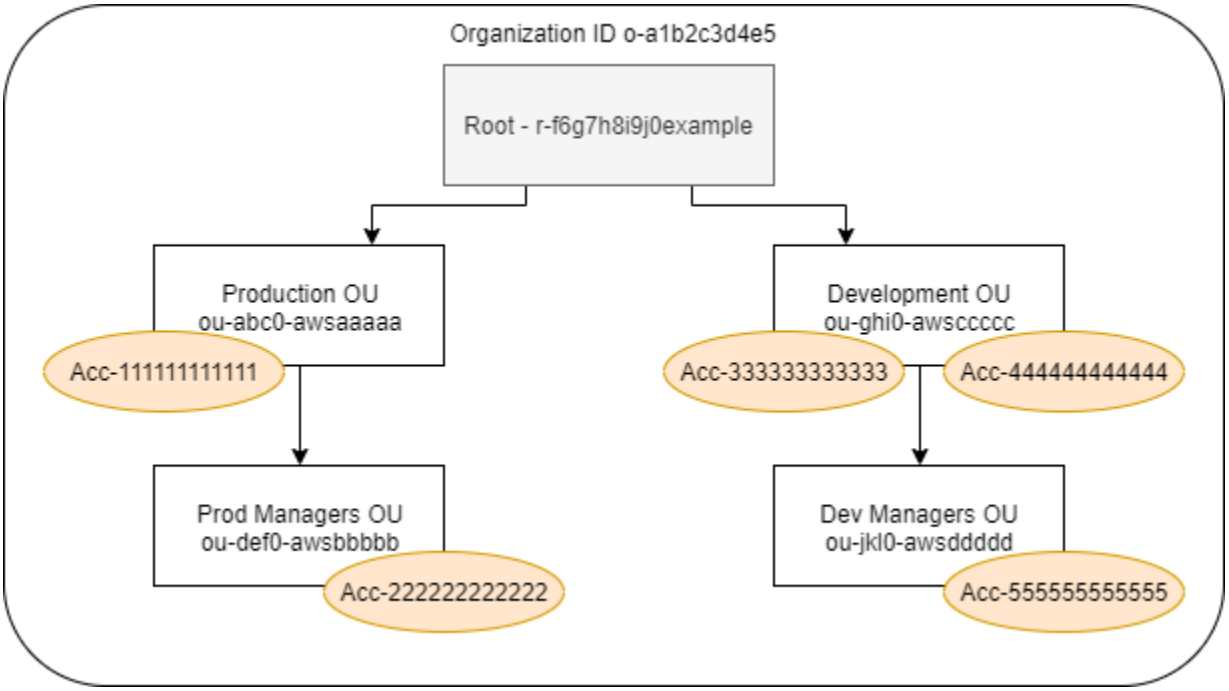
當您產生政策的程式設計報告時，您必須指定 Organizations 實體。這份報告包含指定 SCP 所允許的服務清單。對於每個服務，其包含由政策授予許可之實體或實體子系的最新帳戶活動。如需詳細資訊，請參閱 aws iam generate-organizations-access-report 或 GenerateOrganizationsAccessReport.

檢視報告之前，請確認您了解管理帳戶需求和資訊、報告期間、回報實體和評估的政策類型。如需詳細資訊，請參閱 the section called “關於上次存取資訊的注意事項”。

了解 AWS Organizations 實體路徑

使用 AWS CLI 或 AWS API 產生 AWS Organizations 存取報告時，必須指定實體路徑。路徑是 Organizations 實體結構的文字表示。

您可以使用組織已知的結構來建立實體路徑。例如，假設您在中具有下列組織結構 AWS Organizations。



開發人員管理員 OU 的路徑是使用組織 ID、根目錄和路徑中所有的 OU 包含 OU。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-ghi0-awsccccc/ou-jkl0-awsddddd/
```

生產 OU 中帳戶的路徑是使用組織、根、OU 和帳戶號碼的 ID 建立的。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-abc0-awsaaaaa/111111111111/
```

> **Note**
>
> 組織 ID 是全域唯一，但 OU ID 和根 ID 只有在組織內是唯一。這表示沒有兩個組織共用相同的組織 ID。不過，另一個組織的 OU 或根可能與您的 ID 相同。我們建議您在指定 OU 或根時，一律包含組織 ID。

檢視 Organizations 的資訊 (主控台)

您可以使用 IAM 主控台檢視您的根目錄、OU、帳戶或政策的上次存取資訊。

檢視根目錄的資訊 (主控台)

1. AWS Management Console 使用 Organizations 管理帳戶登入資料登入，然後在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格的 Access reports (存取報告) 區段下方，選擇 Organization activity (組織活動)。
3. 在 Organization activity (組織活動) 頁面，選擇 Root (根)。
4. 在 Details and activity (詳細資訊與活動) 索引標籤上檢視 Service access report (服務存取報告) 部分。該資訊包含直接連接至根目錄之政策所允許的服務清單。該資訊會為您顯示服務上次存取哪個帳戶的服務，以及存取時間。如需關於哪個主體存取服務的詳細資訊，請以該帳戶的管理員身分登入帳戶並檢視 IAM 上次存取資訊。
5. 選擇 Attached SCPs (連接的 SCP) 索引標籤，檢視連接至根目錄的服務控制政策 (SCP) 清單。IAM 會顯示與每個政策連接的目標實體數量。您可以使用此資訊來決定要檢閱哪些 SCP。
6. 選擇 SCP 的名稱，以檢視政策允許的所有服務。對於每個服務，可從中檢視服務上次存取哪個帳戶的服務，以及存取時間。
7. 選擇 Edit in AWS Organizations (在 AWS Organizations中編輯) 以檢視其他詳細資訊，並在 Organizations 主控台中編輯 SCP。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的更新 SCP。

檢視 OU 或帳戶的資訊 (主控台)

1. AWS Management Console 使用 Organizations 管理帳戶登入資料登入，然後在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格的 Access reports (存取報告) 區段下方，選擇 Organization activity (組織活動)。

3. 在 Organization activity (組織活動)頁面，展開組織的結構。然後，選擇 OU 或您要檢視任何帳戶的名稱 (管理帳戶除外)。
4. 在 Details and activity (詳細資訊與活動) 索引標籤上檢視 Service access report (服務存取報告) 部分。該資訊包含 SCP 允許的服務清單，而且這些 SCP 都連接至 OU 或帳戶及其所有父系。該資訊會為您顯示服務上次存取哪個帳戶的服務，以及存取時間。如需關於哪個主體存取服務的詳細資訊，請以該帳戶的管理員身分登入帳戶並檢視 IAM 上次存取資訊。
5. 選擇 Attached SCPs (連接的 SCP) 索引標籤，檢視直接連接至 OU 或帳戶的服務控制政策 (SCP) 清單。IAM 會顯示與每個政策連接的目標實體數量。您可以使用此資訊來決定要檢閱哪些 SCP。
6. 選擇 SCP 的名稱，以檢視政策允許的所有服務。對於每個服務，可從中檢視服務上次存取哪個帳戶的服務，以及存取時間。
7. 選擇 Edit in AWS Organizations (在 AWS Organizations中編輯) 以檢視其他詳細資訊，並在 Organizations 主控台中編輯 SCP。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的更新 SCP。

檢視管理帳戶的資訊 (主控台)

1. AWS Management Console 使用 Organizations 管理帳戶登入資料登入，然後在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格的 Access reports (存取報告) 區段下方，選擇 Organization activity (組織活動)。
3. 在 Organization activity (組織活動) 頁面，展開組織的結構，然後選擇管理帳戶的名稱。
4. 在 Details and activity (詳細資訊與活動) 索引標籤上檢視 Service access report (服務存取報告) 部分。這些資訊包括所有 AWS 服務的清單。管理帳戶不受 SCP 限制。該資訊會為您顯示帳戶上次是否存取服務，以及存取的時間。如需關於哪個主體存取服務的詳細資訊，請以該帳戶的管理員身分登入帳戶並檢視 IAM 上次存取資訊。
5. 選擇 Attached SCPs (連接的 SCP) 索引標籤，以確認沒有任何連接的 SCP，因為帳戶就是管理帳戶。

檢視政策的資訊 (主控台)

1. AWS Management Console 使用 Organizations 管理帳戶登入資料登入，然後在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在導覽窗格的 Access reports (存取報告) 區段下方，選擇 Service control policies (SCPs) (服務控制政策 (SCP))。

3. 在Service control policies (SCPs) (服務控制政策 (SCP)) 頁面上，檢視組織的政策清單。您可以檢視每個政策連接的目標實體數量。
4. 選擇 SCP 的名稱，以檢視政策允許的所有服務。對於每個服務，可從中檢視服務上次存取哪個帳戶的服務，以及存取時間。
5. 選擇 Edit in AWS Organizations (在 AWS Organizations中編輯) 以檢視其他詳細資訊，並在 Organizations 主控台中編輯 SCP。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的更新 SCP。

檢視 Organizations 的資訊 (AWS CLI)

您可以使用 AWS CLI 來擷取 Organizations 根目錄、OU、帳戶或策略的服務上次存取的資訊。

檢視 Organizations 服務上次存取的資訊 (AWS CLI)

1. 將您的 Organizations 管理帳戶憑證和必要的 IAM 和 Organizations 許可搭配使用，然後確認已針對根目錄啟用 SCP。如需詳細資訊，請參閱 關於上次存取資訊的注意事項。
2. 產生報告。要求必須包含您希望產生報告的 Organizations 實體 (根帳戶、OU 或帳戶) 路徑。您可以選擇包含 `organization-policy-id` 參數，以檢視特定政策的報告。命令會傳回 `job-id`，然後您可將它用於 `get-organizations-access-report` 命令以監控 `job-status`，直到任務完成為止。

   - AWS IAM generate-organizations-access-report
3. 使用上個步驟的 `job-id` 參數來擷取報告的詳細資訊。

   - AWS IAM get-organizations-access-report

   此命令會傳回實體成員可以存取的服務清單。對於每個服務，命令會傳回帳戶成員上次嘗試存取的日期及時間，以及帳戶的實體路徑。它還會傳回可用來存取的服務數量，以及並未存取的服務數量。如果您指定選用的 `organizations-policy-id` 參數，則可存取的服務就是指定政策允許的服務。

檢視 Organizations 的資訊 (AWS API)

您可以使用 AWS API 擷取 Organizations 根目錄、OU、帳戶或策略上次存取的服務資訊。

若要檢視 Organizations 服務上次存取的資訊 (AWS API)

1. 將您的 Organizations 管理帳戶憑證和必要的 IAM 和 Organizations 許可搭配使用，然後確認已針對根目錄啟用 SCP。如需詳細資訊，請參閱 關於上次存取資訊的注意事項。
2. 產生報告。要求必須包含您希望產生報告的 Organizations 實體 (根帳戶、OU 或帳戶) 路徑。您可以選擇包含 `OrganizationsPolicyId` 參數，以檢視特定政策的報告。操作會傳回 `JobId`，您可將它用於 `GetOrganizationsAccessReport` 操作以監控 `JobStatus`，直到任務完成為止。

   - GenerateOrganizationsAccessReport
3. 使用上個步驟的 `JobId` 參數來擷取報告的詳細資訊。

   - GetOrganizationsAccessReport

   此操作會傳回實體成員可以存取的服務清單。對於每個服務，操作會傳回帳戶成員上次嘗試存取的日期及時間，以及帳戶的實體路徑。它還會傳回可用來存取的服務數量，以及並未存取的服務數量。如果您指定選用的 `OrganizationsPolicyId` 參數，則可存取的服務就是指定政策允許的服務。

使用上次存取資訊的範例案例

您可以使用上次存取的資訊來決定授與 IAM 實體或 AWS Organizations 實體的許可。如需詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

> **Note**
>
> 在 IAM 中檢視實體或政策的存取資訊之前 AWS Organizations，或確定您瞭解資料的報告期間、報告的實體以及評估的政策類型。如需詳細資訊，請參閱 the section called “關於上次存取資訊的注意事項”。

身為管理員的您，可以在可存取性與最低權限取得平衡，以符合您公司的需求。

使用資訊減少 IAM 群組的許可

您可以使用上次存取資訊以減少 IAM 群組許可，使其僅包含您的使用者所需要的服務。此方法在服務等級的授予最低權限中是一個重要的步驟。

例如，Paulo Santos 是負責定義範例公司使用 AWS 者權限的管理員。這家公司剛開始使用 AWS，而軟體開發團隊尚未定義他們將使用的 AWS 服務。Paulo 打算僅提供該團隊所需服務的存取許可，但由於尚未定義相關服務，因此 Paulo 暫時提供該團隊進階使用者許可。然後，他會使用上次存取的資訊來減少群組的許可。

Paulo 使用以下 JSON 文字建立一個名為 `ExampleDevelopment` 的受管政策。然後，他將其連接至名為 `Development` 的群組，並將所有開發人員新增至該群組。

> **Note**
>
> Paulo 的進階使用者可能需要 `iam:CreateServiceLinkedRole` 許可才能使用某些服務和功能。他了解新增此許可會允許使用者建立任何服務連結的角色。他接受其進階使用者的這種風險。

```
{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToAllServicesExceptPeopleManagement",
            "Effect": "Allow",
            "NotAction": [
                "iam:*",
                "organizations:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequiredIamAndOrgsActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

Paulo 決定在他檢視上次存取資訊前等待 90 天，然後讓 Development 群組使用 AWS Management Console。他檢視群組成員曾經存取的服務清單。他了解到用戶在上週訪問了五個服務： AWS CloudTrail Amazon CloudWatch 日誌，Amazon EC2 和 Amazon S3。 AWS KMS他們在第一次評估時訪問了其他一些服務 AWS，但從那時起就沒有。

Paulo 決定減少政策許可，使其僅包含這五種服務和必要的 IAM 與 Organizations 動作。他使用以下 JSON 文字編輯 ExampleDevelopment 政策。

> **Note**
>
> Paulo 的進階使用者可能需要 iam:CreateServiceLinkedRole 許可才能使用某些服務和功能。他了解新增此許可會允許使用者建立任何服務連結的角色。他接受其進階使用者的這種風險。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToListedServices",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "kms:*",
                "cloudtrail:*",
                "logs:*",
                "ec2:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequiredIamAndOrgsActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

為了進一步降低權限，Paulo 可以在 AWS CloudTrail 事件歷史記錄中查看帳戶的事件。他可在此檢視詳細的事件資訊，以用於減少政策的許可，使其僅包含開發人員需要的動作和資源。如需詳細資訊，請參閱《使用指南》中的〈在 CloudTrail 主控台中檢視 CloudTrail 事件AWS CloudTrail〉。

使用資訊減少 IAM 使用者的許可

您可以使用上次存取資訊，以減少個別 IAM 使用者的許可。

例如，Martha Rivera 是 IT 管理員，負責確保公司中的人員沒有過多 AWS 的權限。在定期安全性檢查中，她會檢查所有 IAM 使用者的許可。在這些使用者中，有一位名為 Nikhil Jayashankar 的應用程式開發人員，過去曾經擔任安全工程師。因為任務需求的變更，Nikhil 同時是 `app-dev` 群組和 `security-team` 群組的成員。`app-dev` 群組為其新任務授予多項服務的許可，包括 Amazon EC2、Amazon EBS、Auto Scaling、Amazon S3、Route 53 和 Elastic Transcoder。舊工作的`security-team`群組將許可授予 IAM 和 CloudTrail.

作為管理員，Martha 登入 IAM 主控台，依序選擇 Users (使用者)、名稱 `nikhilj`，然後選擇 Access Advisor (存取顧問) 標籤。

瑪莎回顧了上次訪問列，並通知 Nikhil 最近沒有訪問 IAM，路線 53 CloudTrail，Amazon Elastic Transcoder 以及許多其他服務。 AWS Nikhil 已經存取 Amazon S3。Martha 從服務清單中選擇 S3，並得知 Nikhil 在過去兩週內執行了一些 S3 `List` 動作。在她的公司內部，Martha 確認 Nikhil 不再需要訪問 IAM，因為他不 CloudTrail 再是內部安全團隊的成員。

Martha 現在已經準備好對服務採取行動，並採取行動上次存取的資訊。不過，不同於先前範例中的群組，像是 `nikhilj` 這樣的 IAM 使用者可能會受到多個政策的約束，並且可能是多個群組的成員。Martha 必須小心處理以避免不慎中斷 `nikhilj` 或其他群組成員的存取。除了了解 Nikhil 應有哪些存取，她也必須判斷 Nikhil 應如何接收這些許可。

Martha 選擇 Permissions (許可) 標籤，她檢視哪些政策直接連接至 `nikhilj`，以及從群組連接的政策。她展開每個政策並檢視政策摘要，以了解哪個政策允許 Nikhil 存取他沒有在使用的服務：

- IAM — 受`IAMFullAccess` AWS 管政策會直接附加至群組`nikhilj`並附加至`security-team`群組。
- CloudTrail — 受`AWS CloudTrailReadOnlyAccess` AWS 管理的原則會附加至`security-team`群組。
- Route 53 – `App-Dev-Route53` 客戶受管政策連接至 `app-dev` 群組。
- Elastic Transcoder – `App-Dev-ElasticTranscoder` 客戶受管政策連接至 `app-dev` 群組。

Martha 決定移除直接附加至的IAMFullAccess AWS 受管理政策。nikhilj她也移除 Nikhil 的 security-team 群組成員資格。這兩個動作會移除不必要的 IAM 和 CloudTrail.

Nikhil 存取 Route 53 和 Elastic Transcoder 的許可是由 app-dev 群組所授予。雖然 Nikhil 沒有使用這些服務，但群組的其他成員可能會使用。Martha 會檢閱 app-dev 群組上次存取的資訊，並得知多位成員最近存取 Route 53 和 Amazon S3。但在去年沒有任何群組成員存取過 Elastic Transcoder。她從群組移除 App-Dev-ElasticTranscoder 客戶受管政策。

然後，Martha 檢閱了 App-Dev-ElasticTranscoder 客戶受管政策的上次存取資訊。她發現該政策未連接至任何其他 IAM 身分。她在公司內部進行調查以確定未來不需要此政策，然後將此政策刪除。

刪除 IAM 資源前使用資訊

您可以在刪除 IAM 資源之前使用上次存取資訊，以確保在最後一次有人使用該資源之後已經過一段特定的時間。這適用於使用者、群組、角色及政策。若要進一步了解這些動作的詳細資訊，請參閱下列主題：

- 使用者 – 刪除使用者
- 群組 – 刪除群組
- 角色 – 刪除角色
- 政策 – 刪除受管政策 (這也會從身分中分開政策)

編輯 IAM 政策前使用資訊

您可以在編輯會影響該資源的政策之前，檢閱上次存取資訊中的 IAM 身分 (使用者、群組或角色) 或 IAM 政策。這是重要的，因為您不會想要移除使用該政策者的存取權。

例如，Arnav Desai 是 Example Corp. 的開發人員和 AWS 管理員當他的團隊開始使用時 AWS，他們提供了所有開發人員進階使用者存取權，允許他們完全存取 IAM 和 Organizations 外的所有服務。作為授予最低權限的第一步，Arnav 希望使用 AWS CLI 來檢閱其帳戶中的受管理策略。

因此，Arnav 首先列出其帳戶中連接至身分的客戶受管許可政策，他使用下列命令：

```
aws iam list-policies --scope Local --only-attached --policy-usage-filter
 PermissionsPolicy
```

他從回應中擷取每個政策的 ARN。然後，Arnav 使用下列命令，為每個政策產生上次存取資訊的報告。

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

從回應中，他從 JobId 欄位擷取所產生報告的 ID。然後，Arnav 輪詢下列命令，直到 JobStatus 欄位傳回 COMPLETED 或 FAILED 值。如果任務失敗，他將會擷取錯誤。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

當任務的狀態為 COMPLETED 時，Arnav 剖析 JSON 格式 ServicesLastAccessed 陣列的內容。

```
 "ServicesLastAccessed": [
        {
            "TotalAuthenticatedEntities": 1,
            "LastAuthenticated": 2018-11-01T21:24:33.222Z,
            "ServiceNamespace": "dynamodb",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/IAMExampleUser",
            "ServiceName": "Amazon DynamoDB"
        },

        {
            "TotalAuthenticatedEntities": 0,
            "ServiceNamespace": "ec2",
            "ServiceName": "Amazon EC2"
        },

        {
            "TotalAuthenticatedEntities": 3,
            "LastAuthenticated": 2018-08-25T15:29:51.156Z,
            "ServiceNamespace": "s3",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:role/IAMExampleRole",
            "ServiceName": "Amazon S3"
        }
    ]
```

Arnav 透過此資訊發現 ExamplePolicy1 政策允許存取三項服務、Amazon DynamoDB、Amazon S3 和 Amazon EC2。名為 IAMExampleUser 的 IAM 使用者與 11 月 1 日最後一次嘗試存取 DynamoDB，另有某人於 8 月 25 日使用了 IAMExampleRole 角色嘗試存取 Amazon S3。另有兩個實體在過去一年嘗試存取 Amazon S3。不過，過去一年無人嘗試存取 Amazon EC2。

這表示 Arnav 可以安全地從政策中移除 Amazon EC2 動作。Arnav 想要檢閱該政策目前的 JSON 文件。首先，他必須使用以下命令判斷政策的版本號碼。

```
aws iam list-policy-versions --policy-arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

Arnav 從回應中的 `Versions` 陣列收集到目前的預設版本號碼。然後，他使用該版本號碼 (v2) 以及以下命令請求 JSON 政策文件。

```
aws iam get-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1
 --version-id v2
```

Arnav 將傳回的 JSON 政策文件存放於 `Document` 陣列的 `PolicyVersion` 欄位。在政策文件中，Arnav 搜尋 `ec2` 命名空間中的動作。如果政策中沒有來自其他命名空間的動作，他將會分開政策與受影響的身分 (使用者、群組和角色)。而後他會刪除政策。在這種情況下，政策並包含 Amazon DynamoDB 與 Amazon S3 服務。因此，Arnav 會從文件中移除 Amazon EC2 動作，並儲存變更。然後，他使用下列命令來更新使用新文件版本的政策，然後將該版本設定為預設的政策版本。

```
aws iam create-policy-version --policy-arn arn:aws:iam::123456789012:policy/
ExamplePolicy1 --policy-document file://UpdatedPolicy.json --set-as-default
```

`ExamplePolicy1` 政策現在已更新，以移除不必要 Amazon EC2 服務的存取權。

其他 IAM 案例

有關 IAM 資源 (使用者、群組、角色或政策) 上次嘗試存取服務時間的資訊，可在您完成下列任一項任務時提供協助：

- 政策 – 編輯現有的客戶受管政策或內嵌政策以移除許可
- 政策 – 將內嵌政策轉換為受管政策，然後刪除它
- 政策 – 將明確拒絕新增至現有的政策
- 政策 – 從身分 (使用者、群組或角色) 分開受管政策
- 實體 – 設定許可界限以控制實體 (使用者或角色) 可擁有的最大許可
- 群組 – 從群組移除使用者

使用資訊來調整組織單位的許可

您可以使用上次存取資訊，以強化 AWS Organizations中組織單位 (OU) 的許可。

例如，約翰·斯泰爾斯是 AWS Organizations 管理員。他負責確保公司中的人員 AWS 帳戶 沒有過多的權限。在定期安全稽核中，他會檢查其組織的許可。他的 `Development` OU 包含通常用來測試新

AWS 服務的帳戶。John 決定要定期檢查超過 180 天未存取的服務報告。然後，他的 OU 成員會移除存取那些服務的許可。

John 使用自己的管理帳戶憑證登入 IAM 主控台。在 IAM 主控台，他為 `Development` OU 定位 Organizations 資料。他檢閱「服務存取」報告表格，並看到兩個在他偏好的 180 天內未存取的 AWS 服務。他記得新增許可的開發團隊存取 Amazon Lex 和 AWS Database Migration Service. John 聯絡開發團隊，並確認他們不再有測試這些服務的商業需求。

Martha 現在已經準備好對上次存取的資訊採取行動。他選擇 Edit in AWS Organizations (在 AWS Organizations中編輯)，而且收到提醒，表示已將 SCP 連接至多個實體。他選擇 Continue (繼續)。在中 AWS Organizations，他會檢閱目標，以瞭解 SCP 所附加的「Organizations」實體。所有實體都位於 `Development` OU 內。

約翰決定拒絕訪問 Amazon Lex 和 `NewServiceTest` SCP 中的 AWS Database Migration Service 行動。這個動作移除了不必要的服務存取。

IAM 動作最近存取的資訊服務和動作

下表列出顯示上次存取 IAM 動作資訊的 AWS 服務。如需每個服務中的動作清單，請參閱服務授權參考中AWS 服務的動作、資源和條件金鑰。

服務	服務前綴
AWS Identity and Access Management 存取分析器	access-analyzer
AWS Account Management	帳戶
AWS Certificate Manager	acm
Amazon Managed Workflows for Apache Airflow	airflow
AWS Amplify	mplify
AWS Amplify UI 生成器	amplifyuibuilder
Amazon AppIntegrations	app-integrations
AWS AppConfig	appconfig
Amazon AppFlow	appflow

服務	服務前綴
AWS 應用程式成本分析工具	application-cost-profiler
Amazon CloudWatch 應用洞察	applicationinsights
AWS App Mesh	appmesh
Amazon AppStream 2.0	appstream
AWS AppSync	appsync
Amazon Managed Service for Prometheus	aps
Amazon Athena	athena
AWS Audit Manager	auditmanager
AWS Auto Scaling	自動擴展
AWS Marketplace	aws-marketplace
AWS Backup	備份
AWS Batch	批次
Amazon Braket	braket
AWS Budgets	預算
AWS Cloud9	Cloud9
AWS CloudFormation	cloudformation
Amazon CloudFront	cloudfront
AWS CloudHSM	cloudhsm
Amazon CloudSearch	cloudsearch
AWS CloudTrail	cloudtrail

服務	服務前綴
Amazon CloudWatch	cloudwatch
AWS CodeArtifact	codeartifact
AWS CodeDeploy	codedeploy
Amazon CodeGuru 分析器	codeguru-profiler
Amazon 評論 CodeGuru 家	codeguru-reviewer
AWS CodePipeline	codepipeline
AWS CodeStar	codestar
AWS CodeStar 通知	codestar-notifications
Amazon Cognito 身分	cognito-identity
Amazon Cognito 使用者集區	cognito-idp
Amazon Cognito Sync	cognito-sync
Amazon Comprehend Medical	comprehendmedical
AWS Compute Optimizer	compute-optimizer
AWS Config	config
Amazon Connect	connect
AWS Cost and Usage Report	cur
AWS Glue DataBrew	databrew
AWS Data Exchange	dataexchange
AWS Data Pipeline	datapipeline

服務	服務前綴
DynamoDB Accelerator	dax
AWS Device Farm	devicefarm
Amazon DevOps 大師	devops-guru
AWS Direct Connect	directconnect
Amazon Data Lifecycle Manager	dlm
AWS Database Migration Service	dms
Amazon DocumentDB Elastic Clusters	docdb-elastic
AWS Directory Service	ds
Amazon DynamoDB	dynamodb
Amazon Elastic Block Store	ebs
Amazon Elastic Compute Cloud	ec2
Amazon Elastic Container Registry	ecr
Amazon Elastic Container Registry Public	ecr-public
Amazon Elastic Container Service	ecs
Amazon Elastic Kubernetes Service	eks
Amazon Elastic Inference	elastic-inference
Amazon ElastiCache	elasticache
AWS Elastic Beanstalk	elasticbeanstalk
Amazon Elastic File System	elasticfilesystem
Elastic Load Balancing	elasticloadbalancing

服務	服務前綴
Amazon Elastic Transcoder	elastictranscoder
Amazon EMR on EKS (EMR 容器)	emr-containers
Amazon EMR Serverless	emr-serverless
Amazon OpenSearch 服務	es
Amazon EventBridge	事件
Amazon CloudWatch 顯然	evidently
Amazon FinSpace	finspace
Amazon 數據 Firehose	firehose
AWS Fault Injection Service	fis
AWS Firewall Manager	fms
Amazon Fraud Detector	frauddetector
Amazon FSx	fsx
Amazon GameLift	gamelift
Amazon Location Service	geo
Amazon S3 Glacier	glacier
Amazon Managed Grafana	grafana
AWS IoT Greengrass	greengrass
AWS Ground Station	groundstation
Amazon GuardDuty	guardduty
AWS HealthLake	healthlake

服務	服務前綴
Amazon Honeycode	honeycode
AWS Identity and Access Management	iam
AWS 身分存放區	identitystore
EC2 Image Builder	imagebuilder
Amazon Inspector Classic	inspector
Amazon Inspector	inspector2
AWS IoT	iot
AWS IoT Analytics	iotanalytics
AWS IoT Core Device Advisor	iotdeviceadvisor
AWS IoT Events	iotevents
AWS IoT Fleet Hub	iotfleethub
AWS IoT SiteWise	iotsitewise
AWS IoT TwinMaker	iottwinmaker
AWS IoT Wireless	iotwireless
Amazon Interactive Video Service	ivs
Amazon Interactive Video Service Chat	ivschat
Amazon Managed Streaming for Apache Kafka	kafka
Amazon Managed Streaming for Kafka Connect	kafkaconnect
Amazon Kendra	kendra
Amazon Kinesis	kinesis

服務	服務前綴
Amazon Kinesis Analytics V2	kinesisanalytics
AWS Key Management Service	kms
AWS Lambda	lambda
Amazon Lex	lex
AWS License Manager 訂閱管理員	license-manager-linux-subscriptions
Amazon Lightsail	lightsail
Amazon CloudWatch 日誌	日誌
Amazon Lookout for Equipment	lookoutequipment
Amazon Lookout for Metrics	lookoutmetrics
Amazon Lookout for Vision	lookoutvision
AWS Mainframe Modernization	m2
Amazon Managed Blockchain	managedblockchain
AWS Elemental MediaConnect	mediaconnect
AWS Elemental MediaConvert	mediaconvert
AWS Elemental MediaLive	medialive
AWS Elemental MediaStore	mediastore
AWS Elemental MediaTailor	mediatailor
Amazon MemoryDB for Redis	memorydb
AWS Application Migration Service	mgn

服務	服務前綴
AWS Migration Hub	mgh
AWS Migration Hub 策略建議	migration hub-strategy
Amazon Pinpoint	mobiletargeting
Amazon MQ	mq
AWS Network Manager	networkmanager
Amazon Nimble Studio	nimble
AWS HealthOmics	omics
AWS OpsWorks	opsworks
AWS OpsWorks CM	opsworks-cm
AWS Outposts	outposts
AWS Organizations	組織
AWS Panorama	panorama
AWS Performance Insights	pi
Amazon EventBridge 管道	pipes
Amazon Polly	polly
Amazon Connect Customer Profiles	profile
Amazon QLDB	qldb
AWS Resource Access Manager	ram
AWS 資源回收筒	rbin
Amazon Relational Database Service	rds

服務	服務前綴
Amazon Redshift	redshift
Amazon Redshift 資料 API	redshift-data
AWS Migration Hub Refactor Spaces	refactor-spaces
Amazon Rekognition	rekognition
AWS Resilience Hub	resiliencehub
AWS 資源總管	resource-explorer-2
AWS Resource Groups	resource-groups
AWS RoboMaker	robomaker
AWS Identity and Access Management 任何角色	rolesanywhere
Amazon Route 53	route53
Amazon Route 53 Recovery Controls	route53-recovery-control-config
Amazon Route 53 Recovery Readiness	route53-recovery-readiness
Amazon Route 53 Resolver	route53resolver
AWS CloudWatch朗姆酒	rum
Amazon Simple Storage Service	s3
Amazon S3 on Outposts	s3-outposts
Amazon SageMaker 地理空間	sagemaker-geospatial
Savings Plans	savingsplans

服務	服務前綴
Amazon EventBridge 模式	schemas
Amazon SimpleDB	sdb
AWS Secrets Manager	secretsmanager
AWS Security Hub	securityhub
Amazon Security Lake	securitylake
AWS Serverless Application Repository	serverlessrepo
AWS Service Catalog	servicecatalog
AWS Cloud Map	servicediscovery
Service Quotas	servicequotas
Amazon Simple Email Service	ses
AWS Shield	shield
AWS Signer	signer
AWS SimSpace Weaver	simspaceweaver
AWS Server Migration Service	sms
Amazon Pinpoint 簡訊和語音服務	sms-voice
AWS Snowball	snowball
Amazon Simple Queue Service	sqs
AWS Systems Manager	ssm
AWS Systems Manager Incident Manager	ssm-incidents
適用於 SAP 的 AWS Systems Manager	ssm-sap

服務	服務前綴
AWS Step Functions	states
AWS Security Token Service	sts
Amazon Simple Workflow Service	swf
Amazon CloudWatch Synthetics	synthetics
AWS Resource Groups Tagging API	標籤
Amazon Textract	textract
Amazon Timestream	timestream
AWS 電信網絡生成器	tnb
Amazon Transcribe	transcribe
AWS Transfer Family	傳輸
Amazon Translate	translate
Amazon Connect Voice ID	voiceid
Amazon VPC Lattice	vpc-lattice
AWS WAFV2	wafv2
AWS Well-Architected Tool	wellarchitected
Amazon Connect Wisdom	wisdom
Amazon WorkLink	worklink
Amazon WorkSpaces	工作區
AWS X-Ray	xray

用於動作最近存取資訊的動作

下表會列出可使用動作最近存取資訊的動作。

服務前綴	動作
access-analyzer	訪問分析器：規則 ApplyArchive
	存取分析器:產生 CancelPolicy
	訪問分析器：CheckAccessNotGranted
	訪問分析器：CheckNoNewAccess
	存取分析器：預覽 CreateAccess
	訪問分析器：CreateAnalyzer
	訪問分析器：規則 CreateArchive
	訪問分析器：DeleteAnalyzer
	訪問分析器：規則 DeleteArchive
	存取分析器：預覽 GetAccess
	訪問分析器：資源 GetAnalyzed
	訪問分析器：GetAnalyzer
	訪問分析器：規則 GetArchive
	訪問分析器：GetFinding
	存取分析器：原則 GetGenerated
	訪問分析器：ListAccessPreviewFindings
	存取分析器：預覽 ListAccess
	訪問分析器：資源 ListAnalyzed
	訪問分析器：ListAnalyzers

服務前綴	動作
	訪問分析器：規則 ListArchive
	訪問分析器：ListFindings
	訪問分析器：世代 ListPolicy
	存取分析器:產生 StartPolicy
	訪問分析器：掃描 StartResource
	訪問分析器：規則 UpdateArchive
	訪問分析器：UpdateFindings
	訪問分析器：ValidatePolicy
帳戶	帳戶：DeleteAlternate聯繫
	帳戶：DisableRegion
	帳戶：EnableRegion
	帳戶：GetAlternate聯繫
	帳戶：GetContact信息
	帳戶：GetRegionOptStatus
	帳戶：ListRegions
	帳戶：PutAlternate聯繫
	帳戶：PutContact信息

服務前綴	動作
acm	ACM：DeleteCertificate ACM：DescribeCertificate ACM：ExportCertificate ACM：GetAccount配置 ACM：GetCertificate ACM：ImportCertificate ACM：ListCertificates ACM：PutAccount配置 ACM：RenewCertificate ACM：RequestCertificate ACM: ResendValidation 電子郵件 ACM：UpdateCertificate選項
airflow	氣流：CreateCli令牌 氣流：CreateEnvironment 氣流：CreateWebLoginToken 氣流：DeleteEnvironment 氣流：GetEnvironment 氣流：ListEnvironments 氣流：PublishMetrics 氣流：UpdateEnvironment

服務前綴	動作
mplify	放大：CreateApp 放大:環境 CreateBackend 放大：CreateBranch 放大：CreateDeployment 放大:CreateDomain關聯 放大：鉤 CreateWeb 放大：DeleteApp 放大:環境 DeleteBackend 放大：DeleteBranch 放大:DeleteDomain關聯 放大：DeleteJob 放大：鉤 DeleteWeb 放大：GenerateAccess記錄 放大：GetApp 放大：網址 GetArtifact 放大:環境 GetBackend 放大：GetBranch 放大:GetDomain關聯 放大：GetJob 放大：鉤 GetWeb 放大：ListApps

服務前綴	動作
	放大：ListArtifacts
	放大：環境 ListBackend
	放大：ListBranches
	放大:ListDomain協會
	放大：ListJobs
	放大：鉤 ListWeb
	放大：StartDeployment
	放大：StartJob
	放大：StopJob
	放大：UpdateApp
	放大：UpdateBranch
	放大:UpdateDomain關聯
	放大：鉤 UpdateWeb

服務前綴	動作
amplifyuibuilder	放大器:CreateComponent 放大器:CreateForm 放大器:CreateTheme 放大器:DeleteComponent 放大器:DeleteForm 放大器:DeleteTheme 放大器:ExportComponents 放大器:ExportThemes 放大器:Job GetCodegen 放大器:工作 ListCodegen 放大器:ListComponents 放大器:ListForms 放大器:ListThemes 放大器:標誌 ResetMetadata 放大器:Job StartCodegen 放大器:UpdateComponent 放大器:UpdateForm 放大器:UpdateTheme

服務前綴	動作
app-integrations	應用程序集成：CreateApplication
	應用程式整合：整合 CreateData
	應用程式整合：整合 CreateEvent
	應用程序集成：DeleteApplication
	應用程式整合：整合 DeleteData
	應用程式整合：整合 DeleteEvent
	應用程序集成：GetApplication
	應用程式整合：整合 GetData
	應用程式整合：整合 GetEvent
	應用程式整合：關聯 ListApplication
	應用程序集成：ListApplications
	應用程序集成：ListDataIntegrationAssociations
	應用程式整合：整合 ListData
	應用程序集成：ListEventIntegrationAssociations
	應用程式整合：整合 ListEvent
	應用程序集成：UpdateApplication
	應用程式整合：整合 UpdateData
	應用程式整合：整合 UpdateEvent

服務前綴	動作
appconfig	應用程序配置：CreateApplication 應用程序配置文件 CreateConfiguration 應用程序配置：策略 CreateDeployment 應用程序配置：CreateEnvironment 應用程序配置：CreateExtension 應用程序配置：CreateExtension關聯 應用程序配置：CreateHostedConfigurationVersion 應用程序配置：DeleteApplication 應用程序配置文件 DeleteConfiguration 應用程序配置：策略 DeleteDeployment 應用程序配置：DeleteEnvironment 應用程序配置：DeleteExtension 應用程序配置：DeleteExtension關聯 應用程序配置：DeleteHostedConfigurationVersion 應用程序配置：GetApplication 應用程序配置：GetConfiguration 應用程序配置文件 GetConfiguration 應用程序配置：GetDeployment 應用程序配置：策略 GetDeployment 應用程序配置：GetEnvironment 應用程序配置：GetExtension

服務前綴	動作
	應用程序配置：GetExtension關聯 應用程序配置：GetHostedConfigurationVersion 應用程序配置：ListApplications 應用程序配置文件 ListConfiguration 應用程序配置：ListDeployments 應用程序配置：策略 ListDeployment 應用程序配置：ListEnvironments 應用程序配置：ListExtension關聯 應用程序配置：ListExtensions 應用程序配置：ListHostedConfigurationVersions 應用程序配置：StartDeployment 應用程序配置：StopDeployment 應用程序配置：UpdateApplication 應用程序配置文件 UpdateConfiguration 應用程序配置：策略 UpdateDeployment 應用程序配置：UpdateEnvironment 應用程序配置：UpdateExtension 應用程序配置：UpdateExtension關聯 應用程序配置：ValidateConfiguration

服務前綴	動作
appflow	應用程序流程：執CancelFlow行
	應用程式流程:CreateConnector設定
	應用程序流程：CreateFlow
	應用程式流程:DeleteConnector設定
	應用程序流程：DeleteFlow
	應用程序流程：DescribeConnector
	應用程式流程:DescribeConnector實體
	應用程式流程:DescribeConnector設定
	應用程序流程：DescribeConnectors
	應用程序流程：DescribeFlow
	應用程序流程：DescribeFlowExecutionRecords
	應用程式流程:ListConnector實體
	應用程序流程：ListConnectors
	應用程序流程：ListFlows
	應用程序流程：RegisterConnector
	應用程序流程：ResetConnectorMetadataCache
	應用程序流程：StartFlow
	應用程序流程：StopFlow
	應用程序：UnRegister連接器
	應用程式流程:UpdateConnector設定
	應用程式流程:註冊 UpdateConnector

服務前綴	動作
	應用程序流程：UpdateFlow
application-cost-profiler	應用程式成本效能分析工具：定義 DeleteReport 應用程式成本效能分析工具：定義 GetReport 應用程式成本效能分析工具：用法 ImportApplication 應用程式成本效能分析工具：定義 ListReport 應用程式成本效能分析工具：定義 PutReport 應用程式成本效能分析工具：定義 UpdateReport

服務前綴	動作
applicationinsights	應用見解：AddWorkload
	應用見解：CreateApplication
	應用見解：CreateComponent
	應用程式洞察:CreateLog模式
	應用見解：DeleteApplication
	應用見解：DeleteComponent
	應用程式洞察:DeleteLog模式
	應用見解：DescribeApplication
	應用見解：DescribeComponent
	應用程式洞察：DescribeComponent組態
	應用見解：DescribeComponentConfigurationRecommendation
	應用程式洞察:DescribeLog模式
	應用見解：DescribeObservation
	應用見解：DescribeProblem
	應用見解:DescribeProblem觀察
	應用見解：DescribeWorkload
	應用見解：ListApplications
	應用見解：ListComponents
	應用程式洞察：歷史 ListConfiguration
	應用程式洞察:ListLog模式
	應用見解：ListLogPatternSets

服務前綴	動作
	應用見解：ListProblems
	應用見解：ListWorkloads
	應用見解：RemoveWorkload
	應用見解：UpdateApplication
	應用見解：UpdateComponent
	應用程式洞察：UpdateComponent組態
	應用程式洞察:UpdateLog模式
	應用見解：UpdateWorkload

服務前綴	動作
appmesh	應用網格:CreateGateway路線 應用網格：CreateMesh 應用網格：CreateRoute 應用程式網格:閘CreateVirtual道 應用網格:節點 CreateVirtual 應用程式:CreateVirtual路由器 應用程式網格:CreateVirtual服務 應用網格:DeleteGateway路線 應用網格：DeleteMesh 應用網格：DeleteRoute 應用程式網格:閘DeleteVirtual道 應用網格:節點 DeleteVirtual 應用程式:DeleteVirtual路由器 應用程式網格:DeleteVirtual服務 應用網格:DescribeGateway路線 應用網格：DescribeMesh 應用網格：DescribeRoute 應用程式網格:閘DescribeVirtual道 應用網格:節點 DescribeVirtual 應用程式:DescribeVirtual路由器 應用程式網格:DescribeVirtual服務

服務前綴	動作
	應用網格:ListGateway路線 應用網格：ListMeshes 應用網格：ListRoutes 應用程式網格:閘ListVirtual道 應用網格:節點 ListVirtual 應用程式:ListVirtual路由器 應用程式網格：ListVirtual服務 應用程式網格:StreamAggregated資源 應用網格:UpdateGateway路線 應用網格：UpdateMesh 應用網格：UpdateRoute 應用程式網格:閘UpdateVirtual道 應用網格:節點 UpdateVirtual 應用程式:UpdateVirtual路由器 應用程式網格:UpdateVirtual服務

服務前綴	動作
appstream	appstream：AssociateAppBlockBuilderAppBlock
	appstream：艦隊 AssociateApplication
	appstream：AssociateApplicationToEntitlement
	appstream：AssociateFleet
	appstream：BatchAssociateUserStack
	appstream：BatchDisassociateUserStack
	appstream：CopyImage
	appstream: 封鎖 CreateApp
	appstream：CreateAppBlockBuilder
	應用程式串流:串流網址 CreateApp BlockBuilder
	appstream：CreateApplication
	appstream：Config CreateDirectory
	appstream：CreateEntitlement
	appstream：CreateFleet
	appstream：CreateImage生成器
	appstream：網址 CreateImage BuilderStreaming
	appstream：CreateStack
	appstream：網址 CreateStreaming
	appstream: 影像 CreateUpdated
	appstream：CreateUsageReportSubscription
	appstream：CreateUser

服務前綴	動作
	appstream: 封鎖 DeleteApp
	appstream：DeleteAppBlockBuilder
	appstream：DeleteApplication
	appstream：Config DeleteDirectory
	appstream：DeleteEntitlement
	appstream：DeleteFleet
	appstream：DeleteImage
	appstream：DeleteImage生成器
	appstream：DeleteImage權限
	appstream：DeleteStack
	appstream：DeleteUsageReportSubscription
	appstream：DeleteUser
	appstream: DescribeApp BlockBuilder AppBlock 關聯
	appstream：DescribeAppBlockBuilders
	appstream：塊 DescribeApp
	appstream：DescribeApplicationFleetAssociations
	appstream：DescribeApplications
	appstream：DescribeDirectory配置
	appstream：DescribeEntitlements
	appstream：DescribeFleets
	appstream：DescribeImage建設者

服務前綴	動作
	appstream：DescribeImage權限
	appstream：DescribeImages
	appstream：DescribeSessions
	appstream：DescribeStacks
	appstream：DescribeUsageReportSubscriptions
	appstream：DescribeUsers
	appstream：DescribeUserStackAssociations
	appstream：DisableUser
	appstream：DisassociateAppBlockBuilderAppBlock
	appstream：艦隊 DisassociateApplication
	appstream：DisassociateApplicationFromEntitlement
	appstream：DisassociateFleet
	appstream：EnableUser
	appstream：ExpireSession
	appstream：艦隊 ListAssociated
	appstream: 堆疊 ListAssociated
	appstream: ListEntitled 應用
	appstream：StartAppBlockBuilder
	appstream：StartFleet
	appstream：StartImage生成器
	appstream：StopAppBlockBuilder

服務前綴	動作
	appstream：StopFleet
	appstream：StopImage生成器
	appstream：UpdateAppBlockBuilder
	appstream：UpdateApplication
	appstream：Config UpdateDirectory
	appstream：UpdateEntitlement
	appstream：UpdateFleet
	appstream：UpdateImage權限
	appstream：UpdateStack

服務前綴	動作
appsync	應用程序同步：AssociateApi 應用程序同步：AssociateMergedGraphqlApi 應用程序同步：AssociateSourceGraphqlApi 應用程序同步：緩CreateApi存 應用程序同步：密鑰 CreateApi 應用程式同步：CreateData來源 應用程式同步：CreateDomain名稱 應用程序同步：CreateFunction 應用程序同步：CreateGraphqlApi 應用程序同步：CreateResolver 應用程序同步：CreateType 應用程序同步：緩DeleteApi存 應用程序同步：密鑰 DeleteApi 應用程式同步：DeleteData來源 應用程式同步：DeleteDomain名稱 應用程序同步：DeleteFunction 應用程序同步：DeleteGraphqlApi 應用程序同步：DeleteResolver 應用程序同步：DeleteType 應用程序同步：DisassociateApi 應用程序同步：DisassociateMergedGraphqlApi

服務前綴	動作
	應用程序同步：DisassociateSourceGraphqlApi
	應用程序同步：EvaluateCode
	應用程序同步：EvaluateMapping模板
	應用程序同步：緩FlushApi存
	應用程式同步:GetApi關聯
	應用程序同步：緩GetApi存
	應用程式同步：GetData來源
	應用程序同步：GetDataSourceIntrospection
	應用程式同步：GetDomain名稱
	應用程序同步：GetFunction
	應用程序同步：GetGraphqlApi
	應用程式同步：GetGraphqlApiEnvironment變數
	應用程式同步:網GetIntrospection要
	應用程序同步：GetResolver
	應用程序同步：GetSchemaCreationStatus
	應用程序同步：GetSourceApiAssociation
	應用程序同步：GetType
	應用程序同步：密鑰 ListApi
	應用程式同步:ListData來源
	應用程式同步：ListDomain名稱
	應用程序同步：ListFunctions

服務前綴	動作
	應用程序同步：ListGraphqlApi
	應用程序同步：ListResolvers
	應用程序同步：ListResolversByFunction
	應用程序同步：ListSourceApiAssociations
	應用程序同步：ListTypes
	應用程序同步：ListTypesByAssociation
	應用程式同步：PutGraphqlApiEnvironment變數
	應用程序同步：StartDataSourceIntrospection
	應用程式同步:StartSchema建立
	應用程序同步：StartSchema合併
	應用程序同步：緩UpdateApi存
	應用程序同步：密鑰 UpdateApi
	應用程式同步：UpdateData來源
	應用程式同步：UpdateDomain名稱
	應用程序同步：UpdateFunction
	應用程序同步：UpdateGraphqlApi
	應用程序同步：UpdateResolver
	應用程序同步：UpdateSourceApiAssociation
	應用程序同步：UpdateType

服務前綴	動作
aps	AP：CreateAlertManagerDefinition aps: CreateLogging 配置 AP：CreateRuleGroupsNamespace AP：CreateScraper AP：CreateWorkspace AP：DeleteAlertManagerDefinition aps: DeleteLogging 配置 AP：DeleteRuleGroupsNamespace AP：DeleteScraper AP：DeleteWorkspace AP：DescribeAlertManagerDefinition aps: DescribeLogging 配置 AP：DescribeRuleGroupsNamespace AP：DescribeScraper AP：DescribeWorkspace AP：GetDefaultScraperConfiguration AP：ListRuleGroupsNamespaces AP：ListScrapers AP：ListWorkspaces AP：PutAlertManagerDefinition AP：PutRuleGroupsNamespace

服務前綴	動作
	aps: UpdateLogging 配置 aps: UpdateWorkspace 別名

服務前綴	動作
athena	雅典娜：BatchGetNamedQuery 雅典娜：BatchGetPreparedStatement 雅典娜：BatchGetQueryExecution 雅典娜：CancelCapacity預約 雅典娜：CreateCapacity預約 雅典娜：CreateData目錄 雅典娜:CreateNamed查詢 雅典娜：CreateNotebook 雅典娜：CreatePrepared聲明 雅典娜：CreatePresignedNotebookUrl 雅典娜:CreateWork集團 雅典娜：DeleteCapacity預約 雅典娜：DeleteData目錄 雅典娜:DeleteNamed查詢 雅典娜：DeleteNotebook 雅典娜：DeletePrepared聲明 雅典娜:DeleteWork集團 雅典娜：ExportNotebook 雅典娜：GetCalculation執行 雅典娜：GetCalculationExecutionCode 雅典娜：GetCalculationExecutionStatus

服務前綴	動作
	雅典娜：GetCapacityAssignmentConfiguration 雅典娜：GetCapacity預約 雅典娜：GetDatabase 雅典娜：GetData目錄 雅典娜:GetNamed查詢 雅典娜:GetNotebook元數據 雅典娜：GetPrepared聲明 雅典娜：GetQuery執行 雅典娜：GetQuery結果 雅典娜：GetQueryResultsStream 雅典娜：GetQueryRuntimeStatistics 雅典娜：GetSession 雅典娜：GetSession狀態 雅典娜:GetTable元數據 雅典娜:GetWork集團 雅典娜：ImportNotebook 雅典娜：DP ListApplication 大小 雅典娜：ListCalculation處決 雅典娜:ListCapacity預訂 雅典娜：ListDatabases 雅典娜：ListData目錄

服務前綴	動作
	雅典娜：ListEngine版本 雅典娜：ListExecutors 雅典娜:ListNamed查詢 雅典娜:ListNotebook元數據 雅典娜:ListNotebook會話 雅典娜:ListPrepared聲明 雅典娜：ListQuery處決 雅典娜：ListSessions 雅典娜:ListTable元數據 雅典娜:ListWork群組 雅典娜：PutCapacityAssignmentConfiguration 雅典娜：StartCalculation執行 雅典娜：StartQuery執行 雅典娜：StartSession 雅典娜：StopCalculation執行 雅典娜：StopQuery執行 雅典娜：TerminateSession 雅典娜：UpdateCapacity預約 雅典娜：UpdateData目錄 雅典娜:UpdateNamed查詢 雅典娜：UpdateNotebook

服務前綴	動作
	雅典娜:UpdateNotebook元數據 雅典娜：UpdatePrepared聲明 雅典娜:UpdateWork集團

服務前綴	動作
auditmanager	稽核管理員:AssociateAssessmentReportEvidence資料夾
	稽核經理:BatchAssociateAssessmentReport證據
	稽核經理:評估 BatchCreate DelegationBy
	稽核經理:評估 BatchDelete DelegationBy
	稽核經理:BatchDisassociateAssessmentReport證據
	稽核經理：BatchImportEvidenceToAssessmentControl
	稽核經理：CreateAssessment
	稽核經理:框架 CreateAssessment
	稽核經理:CreateAssessment報告
	稽核經理：CreateControl
	稽核經理：DeleteAssessment
	稽核經理:框架 DeleteAssessment
	稽核經理：DeleteAssessmentFrameworkShare
	稽核經理:DeleteAssessment報告
	稽核經理：DeleteControl
	稽核經理：DeregisterAccount
	稽核經理：DeregisterOrganizationAdminAccount
	稽核管理員:DisassociateAssessmentReportEvidence資料夾
	稽核經理:狀態 GetAccount
	稽核經理：GetAssessment
	稽核經理:框架 GetAssessment

服務前綴	動作
	稽核經理：GetAssessmentReportUrl
	稽核管理員:GetChange記錄
	稽核經理：GetControl
	稽核經理：GetDelegations
	稽核經理：GetEvidence
	稽核管理員:GetEvidenceByEvidence資料夾
	稽核管理員：GetEvidenceFileUpload網址
	稽核管理員:GetEvidence資料夾
	稽核經理:評估 GetEvidence FoldersBy
	稽核經理：GetEvidenceFoldersByAssessmentControl
	稽核經理：GetInsights
	稽核經理：GetInsightsByAssessment
	稽核經理：GetOrganizationAdminAccount
	稽核經理：GetServicesInScope
	稽核經理：GetSettings
	稽核管理員:ListAssessmentControlInsightsByControl網域
	稽核經理:框架 ListAssessment
	稽核經理:ListAssessmentFrameworkShare請求
	稽核經理:ListAssessment報告
	稽核經理：ListAssessments
	稽核經理：ListControlDomainInsights

服務前綴	動作
	稽核經理：ListControlDomainInsightsByAssessment
	稽核經理：ListControlInsightsByControlDomain
	稽核經理：ListControls
	稽核經理:ListKeywordsForData來源
	稽核經理：ListNotifications
	稽核經理：RegisterAccount
	稽核經理：RegisterOrganizationAdminAccount
	稽核經理：StartAssessmentFrameworkShare
	稽核經理：UpdateAssessment
	稽核經理:控UpdateAssessment制
	稽核經理:狀態 UpdateAssessment ControlSet
	稽核經理:框架 UpdateAssessment
	稽核經理：UpdateAssessmentFrameworkShare
	稽核經理:狀態 UpdateAssessment
	稽核經理：UpdateControl
	稽核經理：UpdateSettings
	稽核經理：ValidateAssessmentReportIntegrity

服務前綴	動作
自動擴展	自動調度資源：AttachInstances 自動調度：AttachLoad平衡器 自動調度資源:群AttachLoadBalancerTarget組 自動調度資源：AttachTraffic來源 自動調度資源：BatchDeleteScheduledAction 自動調度資源：BatchPutScheduledUpdateGroupAction 自動調度：CancelInstance重新整理 自動調度資源：CompleteLifecycle動作 自動調度資源：CreateAutoScalingGroup 自動調度資源：CreateLaunch組態 自動調度資源：DeleteAutoScalingGroup 自動調度資源：DeleteLaunch組態 自動調度資源：鉤 DeleteLifecycle 自動調度資源：DeleteNotification組態 自動調度資源：DeletePolicy 自動調度資源：DeleteScheduled動作 自動調度資源：池 DeleteWarm 自動調度資源：DescribeAccount限制 自動調度資源：類型 DescribeAdjustment 自動調度資源：DescribeAutoScalingGroups 自動調度資源：DescribeAutoScalingInstances

服務前綴	動作
	自動調度資源：類型 DescribeAuto ScalingNotification
	自動調度：DescribeInstance重新整理
	自動調度資源：DescribeLaunch組態
	自動調度資源：掛鉤 DescribeLifecycle
	自動調度資源：DescribeLifecycleHookTypes
	自動調度：DescribeLoad平衡器
	自動調度資源:群DescribeLoadBalancerTarget組
	自動調度資源：DescribeMetricCollectionTypes
	自動調度資源：DescribeNotification組態
	自動調度資源：DescribePolicies
	自動調度資源：DescribeScaling活動
	自動調度資源：DescribeScalingProcessTypes
	自動調度資源：DescribeScheduled動作
	自動調度資源：DescribeTerminationPolicyTypes
	自動調度資源：DescribeTraffic來源
	自動調度資源：池 DescribeWarm
	自動調度資源：DetachInstances
	自動調度：DetachLoad平衡器
	自動調度資源:群DetachLoadBalancerTarget組
	自動調度資源：DetachTraffic來源
	自動調度資源：DisableMetrics集合

服務前綴	動作
	自動調度資源：EnableMetrics集合 自動調度資源：EnterStandby 自動調度資源：ExecutePolicy 自動調度資源：ExitStandby 自動調度資源：GetPredictiveScalingForecast 自動調度資源：鉤 PutLifecycle 自動調度資源：PutNotification組態 自動調度資源：政策 PutScaling 自動調度資源：PutScheduledUpdateGroup動作 自動調度資源：池 PutWarm 自動調度資源：RecordLifecycleActionHeartbeat 自動調度資源：ResumeProcesses 自動調度：RollbackInstance重新整理 自動調度資源：SetDesired容量 自動調度資源：Health SetInstance 自動調度資源：SetInstance保護 自動調度：StartInstance重新整理 自動調度資源：SuspendProcesses 自動調度資源：TerminateInstanceInAutoScalingGroup 自動調度資源：UpdateAutoScalingGroup
aws-marketplace	AWS 市場：GetEntitlements

服務前綴	動作
備份	備份：CancelLegal保持 備份：CreateBackup計劃 備份:CreateBackup選擇 備份:CreateBackup文件庫 備份：CreateFramework 備份：CreateLegal保持 備份：CreateLogicallyAirGappedBackupVault 備份：CreateReport計劃 備份：CreateRestoreTestingPlan 備份：CreateRestoreTestingSelection 備份：DeleteBackup計劃 備份:DeleteBackup選擇 備份:DeleteBackup文件庫 備份：DeleteBackupVaultAccess策略 備份:DeleteBackupVaultLock配置 備份：DeleteBackupVaultNotifications 備份：DeleteFramework 備份:DeleteRecovery點 備份：DeleteReport計劃 備份：DeleteRestoreTestingPlan 備份：DeleteRestoreTestingSelection

服務前綴	動作
	備份:DescribeBackupJob 備份:DescribeBackup文件庫 備份:DescribeCopyJob 備份：DescribeFramework 備份:DescribeGlobal設定 備份:DescribeProtected資源 備份:DescribeRecovery點 備份:DescribeRegion設定 備份:DescribeReportJob 備份：DescribeReport計劃 備份:DescribeRestoreJob 備份:DisassociateRecovery點 備份:DisassociateRecoveryPointFrom父系 備份：ExportBackupPlanTemplate 備份：GetBackup計劃 備份 GetBackup PlanFrom 備份:GetBackupPlanFrom範本 備份:GetBackup選擇 備份：GetBackupVaultAccess策略 備份：GetBackupVaultNotifications 備份：GetLegal保持

服務前綴	動作
	備份:GetRecoveryPointRestore中繼資料 備份：GetRestoreJobMetadata 備份:GetRestoreTestingInferred中繼資料 備份：GetRestoreTestingPlan 備份：GetRestoreTestingSelection 備份：GetSupportedResourceTypes 備份:ListBackup工作 備份：ListBackupJobSummaries 備份:ListBackup計劃 備份：ListBackupPlanTemplates 備份：ListBackupPlanVersions 備份:ListBackup選擇 備份:ListBackup儲存庫 備份:ListCopy工作 備份：ListCopyJobSummaries 備份：ListFrameworks 備份：ListLegal保留 備份:ListProtected資源 備份：ListRecoveryPointsByBackupVault 備份：ListRecoveryPointsByLegalHold 備份:ListRecoveryPointsBy資源

服務前綴	動作
	備份:ListReport工作 備份:ListReport計劃 備份:ListRestore工作 備份：ListRestoreJobsByProtectedResource 備份：ListRestoreJobSummaries 備份：ListRestoreTestingPlans 備份：ListRestoreTestingSelections 備份：PutBackupVaultAccess策略 備份:PutBackupVaultLock配置 備份：PutBackupVaultNotifications 備份：PutRestoreValidationResult 備份:StartBackupJob 備份:StartCopyJob 備份:StartReportJob 備份:StartRestoreJob 備份:StopBackupJob 備份：UpdateBackup計劃 備份：UpdateFramework 備份:UpdateGlobal設定 備份：UpdateRecoveryPointLifecycle 備份:UpdateRegion設定

服務前綴	動作
	備份：UpdateReport計劃 備份：UpdateRestoreTestingPlan 備份：UpdateRestoreTestingSelection

服務前綴	動作
批次	批次：CancelJob 批次：CreateCompute環境 批處理：CreateJob隊列 批次：CreateScheduling策略 批次：DeleteCompute環境 批處理：DeleteJob隊列 批次：DeleteScheduling策略 批次：DeregisterJob定義 批次：DescribeCompute環境 批次：DescribeJob定義 批次：DescribeJob佇列 批次：DescribeJobs 批次：DescribeScheduling原則 批次：ListJobs 批次：ListScheduling原則 批次：RegisterJob定義 批次：SubmitJob 批次：TerminateJob 批次：UpdateCompute環境 批處理：UpdateJob隊列 批次：UpdateScheduling策略

服務前綴	動作
braket	布拉克:協議 AcceptUser 胸針：AccessBraket功能 胸章：CancelJob 布拉克：CancelQuantum任務 胸章：CreateJob 布拉克：CreateQuantum任務 胸章：GetDevice 胸章：GetJob 布拉克：GetQuantum任務 胸針:狀態 GetService LinkedRole 胸章：GetUserAgreementStatus 胸章：SearchDevices 胸章：SearchJobs 口頭：SearchQuantum任務

服務前綴	動作
預算	預算：ModifyBudget
	預算:CreateBudget行動
	預算：ModifyBudget
	預算：ModifyBudget
	預算：ModifyBudget
	預算:DeleteBudget行動
	預算：ModifyBudget
	預算：ModifyBudget
	預算：ViewBudget
	預算:DescribeBudget行動
	預算：DescribeBudgetActionHistories
	預算:DescribeBudgetActionsFor帳戶
	預算:DescribeBudgetActionsFor預算
	預算：ViewBudget
	預算：ViewBudget
	預算：ViewBudget
	預算：ViewBudget
	預算：ViewBudget
	預算:ExecuteBudget行動
	預算：ModifyBudget
	預算:UpdateBudget行動

服務前綴	動作
	預算：ModifyBudget 預算：ModifyBudget
Cloud9	雲 9：EC2 CreateEnvironment 雲 9: CreateEnvironment 會員 雲 9: DeleteEnvironment 雲 9: DeleteEnvironment 會員 雲 9: DescribeEnvironment 會員資格 雲 9: DescribeEnvironments 雲 9：狀態 DescribeEnvironment 雲 9: ListEnvironments 雲 9: UpdateEnvironment 雲 9: UpdateEnvironment 會員

服務前綴	動作
cloudformation	雲形：BatchDescribeTypeConfigurations
	雲形：堆棧 CancelUpdate
	雲形：ContinueUpdate回滾
	雲形：CreateChange設置
	雲形:CreateGenerated模板
	雲形：CreateStack
	雲形：CreateStack執行個體
	雲形：CreateStack設置
	雲形：DeactivateType
	雲形：DeleteChange設置
	雲形:DeleteGenerated模板
	雲形：DeleteStack
	雲形：DeleteStack執行個體
	雲形：DeleteStack設置
	雲形：DeregisterType
	雲形：DescribeAccount限制
	雲形：DescribeChange設置
	雲形：DescribeChangeSetHooks
	雲形:DescribeGenerated模板
	雲形：DescribeOrganizations訪問
	雲形：DescribePublisher

服務前綴	動作
	雲形：掃描 DescribeResource
	雲形：狀態 DescribeStack DriftDetection
	雲形：DescribeStack事件
	雲形：DescribeStack實例
	雲形：DescribeStack資源
	雲形：DescribeStackResourceDrifts
	雲形：DescribeStack資源
	雲形：DescribeStacks
	雲形：DescribeStack設置
	雲形：DescribeStackSetOperation
	雲形：DescribeType
	雲形：DescribeType註冊
	雲形：漂移 DetectStack
	雲形：DetectStackResourceDrift
	雲形：DetectStackSetDrift
	雲形：EstimateTemplate成本
	雲形：ExecuteChange設置
	雲形:GetGenerated模板
	雲形：政策 GetStack
	雲形：GetTemplate
	雲形：GetTemplate摘要

服務前綴	動作
	雲形：ImportStacksToStack設置
	雲形：ListChange集
	雲形：ListExports
	雲形：ListGenerated模板
	雲形：ListImports
	雲形：ListResourceScanRelated資源
	雲形：ListResourceScanResources
	雲形:掃描 ListResource
	雲形：漂移 ListStack InstanceResource
	雲形：ListStack執行個體
	雲形：ListStack資源
	雲形：ListStackSetAutoDeploymentTargets
	雲形：ListStackSetOperation結果
	雲形：ListStackSetOperations
	雲形：ListStack集
	雲形:ListType註冊
	雲形：ListTypes
	雲形：ListType版本
	雲形：PublishType
	雲形：RecordHandler進展
	雲形：RegisterPublisher

服務前綴	動作
	雲形：RegisterType
	雲形：RollbackStack
	雲形：政策 SetStack
	雲形：SetType配置
	雲形：SetTypeDefaultVersion
	雲形：SignalResource
	雲形：掃描 StartResource
	雲形：StopStackSetOperation
	雲形：TestType
	雲形:UpdateGenerated模板
	雲形：UpdateStack
	雲形：UpdateStack執行個體
	雲形：UpdateStack設置
	雲形：UpdateTermination保護
	雲形：ValidateTemplate

服務前綴	動作
cloudfront	雲端：AssociateAlias 雲前：策略 CreateCache 雲端：CreateCloudFrontOriginAccessIdentity 雲端：CreateContinuousDeploymentPolicy 雲前：Config CreateField LevelEncryption 雲端：CreateFieldLevelEncryption設定檔 雲端：CreateFunction 雲端：CreateInvalidation 雲端:CreateKey群組 雲端：CreateKeyValueStore 雲端：訂閱 CreateMonitoring 雲端：CreateOriginAccessControl 雲端：CreateOriginRequestPolicy 雲端：CreatePublic金鑰 雲端：CreateRealtimeLogConfig 雲端：CreateResponseHeadersPolicy 雲前：策略 DeleteCache 雲端：DeleteCloudFrontOriginAccessIdentity 雲端：DeleteContinuousDeploymentPolicy 雲端：DeleteDistribution 雲前：Config DeleteField LevelEncryption

服務前綴	動作
	雲端：DeleteFieldLevelEncryption設定檔 雲端：DeleteFunction 雲端:DeleteKey群組 雲端：DeleteKeyValueStore 雲端：訂閱 DeleteMonitoring 雲端：DeleteOriginAccessControl 雲端：DeleteOriginRequestPolicy 雲端：DeletePublic金鑰 雲端：DeleteRealtimeLogConfig 雲端：DeleteResponseHeadersPolicy 雲端:DeleteStreaming分佈 雲端：DescribeFunction 雲端：DescribeKeyValueStore 雲前：策略 GetCache 雲端：GetCachePolicyConfig 雲端：GetCloudFrontOriginAccessIdentity 雲前：Config GetCloud FrontOrigin AccessIdentity 雲端：GetContinuousDeploymentPolicy 雲前：Config GetContinuous DeploymentPolicy 雲前：Config GetDistribution 雲端：GetFieldLevelEncryption

服務前綴	動作
	雲前：Config GetField LevelEncryption 雲端：GetFieldLevelEncryption設定檔 雲端：GetFieldLevelEncryptionProfileConfig 雲端：GetFunction 雲端：GetInvalidation 雲端:GetKey群組 雲端：GetKeyGroupConfig 雲端：訂閱 GetMonitoring 雲端：GetOriginAccessControl 雲前：Config GetOrigin AccessControl 雲端：GetOriginRequestPolicy 雲前：Config GetOrigin RequestPolicy 雲端：GetPublic金鑰 雲端：GetPublicKeyConfig 雲端：GetRealtimeLogConfig 雲端：GetResponseHeadersPolicy 雲前：Config GetResponse HeadersPolicy 雲端:GetStreaming分佈 雲端：GetStreamingDistributionConfig 雲端：政策 ListCache 雲端：ListCloudFrontOriginAccessIdentities

服務前綴	動作
	雲端:ListConflicting別名 雲端：ListContinuousDeploymentPolicies 雲端：ListDistributions 雲端：ListDistributionsByCachePolicyId 雲端:ListDistributionsByKey群組 雲前：ID ListDistributions ByOrigin RequestPolicy 雲端：ListDistributionsByRealtimeLogConfig 雲前：ID ListDistributions ByResponse HeadersPolicy 雲端:ACLid ListDistributions ByWeb 雲前：ListFieldLevelEncryption配置 雲端：ListFieldLevelEncryption設定檔 雲端：ListFunctions 雲端：ListInvalidations 雲端:ListKey群組 雲端：ListKeyValueStores 雲端：ListOriginAccessControls 雲端：ListOriginRequestPolicies 雲端：ListPublic金鑰 雲端：ListRealtimeLogConfigs 雲端：ListResponseHeadersPolicies 雲前端:ListStreaming分佈

服務前綴	動作
	雲端：PublishFunction 雲端：TestFunction 雲前：策略 UpdateCache 雲端：UpdateCloudFrontOriginAccessIdentity 雲端：UpdateContinuousDeploymentPolicy 雲端：UpdateDistribution 雲前：Config UpdateField LevelEncryption 雲端：UpdateFieldLevelEncryption設定檔 雲端：UpdateFunction 雲端:UpdateKey群組 雲端：UpdateKeyValueStore 雲端：UpdateOriginAccessControl 雲端：UpdateOriginRequestPolicy 雲端：UpdatePublic金鑰 雲端：UpdateRealtimeLogConfig 雲端：UpdateResponseHeadersPolicy

服務前綴	動作
cloudhsm	雲：CreateHapg
	雲：CreateHsm
	雲端：用戶端 CreateLuna
	雲：DeleteBackup
	雲：DeleteHapg
	雲：DeleteHsm
	雲端：用戶端 DeleteLuna
	雲：DescribeBackups
	雲：DescribeClusters
	雲：DescribeHapg
	雲：DescribeHsm
	雲端：用戶端 DescribeLuna
	雲：GetConfig
	雲：InitializeCluster
	雲：區域 ListAvailable
	雲：ListHapgs
	雲：ListHsms
	雲端:用戶端 ListLuna
	雲友：屬性 ModifyBackup
	雲：ModifyCluster
	雲：ModifyHapg

服務前綴	動作
	雲：ModifyHsm 雲端：用戶端 ModifyLuna 雲：RestoreBackup

服務前綴	動作
cloudsearch	雲搜索:BuildSuggesters 雲搜索:CreateDomain 雲搜索:DefineAnalysis計劃 雲搜索:DefineExpression 雲搜索:字段 DefineIndex 雲搜索:DefineSuggester 雲搜索:DeleteAnalysis計劃 雲搜索:DeleteDomain 雲搜索:DeleteExpression 雲搜索:字段 DeleteIndex 雲搜索:DeleteSuggester 雲搜索:DescribeAnalysis計劃 雲搜索:DescribeAvailability選項 雲搜索:DescribeDomainEndpointOptions 雲搜索:DescribeDomains 雲搜索:DescribeExpressions 雲搜索：字段 DescribeIndex 雲搜索:DescribeScaling參數 雲搜索:DescribeServiceAccessPolicies 雲搜索:DescribeSuggesters 雲搜索:IndexDocuments

服務前綴	動作
	雲搜索:ListDomain名稱 雲搜索:UpdateAvailability選項 雲搜索:UpdateDomainEndpointOptions 雲搜索:UpdateScaling參數 雲搜索:UpdateServiceAccessPolicies

服務前綴	動作
cloudtrail	雲步道：CancelQuery
	雲步道：CreateChannel
	雲步道：CreateEventDataStore
	雲步道：CreateTrail
	雲步道：DeleteChannel
	雲步道：DeleteEventDataStore
	雲步道：策略 DeleteResource
	雲步道：DeleteTrail
	雲步道：DeregisterOrganizationDelegatedAdmin
	雲步道：DescribeQuery
	雲步道：DescribeTrails
	雲步道：DisableFederation
	雲步道：GetChannel
	雲步道：GetEventDataStore
	雲步道:GetEventDataStore資料
	雲步道：GetEvent選擇器
	雲步道：GetImport
	雲步道：GetInsight選擇器
	雲步道：GetQuery結果
	雲步道：策略 GetResource
	雲步道：GetTrail

服務前綴	動作
	雲軌跡：狀態 GetTrail
	雲步道：ListChannels
	雲步道：ListEventDataStores
	雲步道：失敗 ListImport
	雲步道：ListImports
	雲步道：鑰匙 ListPublic
	雲步道：ListQueries
	雲步道：ListTrails
	雲步道：LookupEvents
	雲步道：PutEvent選擇器
	雲步道：PutInsight選擇器
	雲步道：策略 PutResource
	雲步道：RegisterOrganizationDelegatedAdmin
	雲步道：RestoreEventDataStore
	雲步道：StartEventDataStore攝入
	雲步道：StartImport
	雲步道：StartLogging
	雲步道：StartQuery
	雲步道：StopEventDataStore攝入
	雲步道：StopImport
	雲步道：StopLogging

服務前綴	動作
	雲步道：UpdateChannel 雲步道：UpdateEventDataStore 雲步道：UpdateTrail

服務前綴	動作
cloudwatch	雲觀察:DeleteAlarms 雲觀察:檢測DeleteAnomaly器 雲觀察:DeleteDashboards 雲觀察:DeleteInsight規則 雲觀察:DeleteMetric流 雲觀察:歷史 DescribeAlarm 雲觀察:DescribeAlarms 雲觀察:DescribeAlarmsForMetric 雲觀察:探測DescribeAnomaly器 雲觀察:DescribeInsight規則 雲觀察:DisableAlarm動作 雲觀察:DisableInsight規則 雲觀察:EnableAlarm動作 雲觀察:EnableInsight規則 雲觀察:GetDashboard 雲觀察:GetInsightRuleReport 雲觀察:GetMetric流 雲觀察:ListDashboards 雲觀察:ListManagedInsightRules 雲觀察:ListMetric流 雲觀察:檢測PutAnomaly器

服務前綴	動作
	雲觀察:PutComposite警報 雲觀察:PutDashboard 雲觀察:PutInsight規則 雲觀察:PutManagedInsightRules 雲觀察:PutMetric警報 雲觀察:PutMetric流 雲觀察:狀態 SetAlarm 雲觀察:StartMetric流 雲觀察:StopMetric流

服務前綴	動作
codeartifact	代碼工件：AssociateExternal連接
	代碼工件：CopyPackage版本
	代碼工件：CreateDomain
	代碼工件：CreateRepository
	代碼工件：DeleteDomain
	代碼工件：DeleteDomainPermissionsPolicy
	代碼工件：DeletePackage
	代碼工件：DeletePackage版本
	代碼工件：DeleteRepository
	代碼工件：DeleteRepositoryPermissionsPolicy
	代碼工件：DescribeDomain
	代碼工件：DescribePackage
	代碼工件：DescribePackage版本
	代碼工件：DescribeRepository
	代碼工件：DisassociateExternal連接
	代碼工件：DisposePackage版本
	代碼工件：GetAssociatedPackageGroup
	代碼工件：令牌 GetAuthorization
	代碼工件：GetDomainPermissionsPolicy
	代碼工件：GetPackageVersionAsset
	代碼工件：GetPackageVersionReadme

服務前綴	動作
	代碼工件:端點 GetRepository
	代碼工件：GetRepositoryPermissionsPolicy
	代碼工件：ListDomains
	代碼人工標誌:群ListPackage組
	代碼工件：ListPackages
	代碼工件：ListPackageVersionAssets
	代碼工件：ListPackageVersionDependencies
	代碼工件：ListPackage版本
	代碼工件：ListRepositories
	代碼工件：ListRepositoriesInDomain
	代碼工件：PublishPackage版本
	代碼工件：PutDomainPermissionsPolicy
	代碼工件:PutPackage中繼資料
	代碼工件：PutPackageOriginConfiguration
	代碼工件：PutRepositoryPermissionsPolicy
	代碼工件：存儲ReadFrom庫
	代碼工件：UpdatePackageVersionsStatus
	代碼工件：UpdateRepository

服務前綴	動作
codedeploy	代碼部署：BatchGetApplicationRevisions 代碼部署:BatchGet應用程序 代碼部署：BatchGetDeploymentGroups 代碼部署：BatchGetDeploymentInstances 程式碼部署:BatchGet部署 代碼部署：BatchGetDeploymentTargets 代碼部署：BatchGetOnPremises實例 代碼部署：ContinueDeployment 代碼部署：CreateApplication 代碼部署：CreateDeployment 代碼部署：Config CreateDeployment 程式碼部署:群CreateDeployment組 代碼部署：DeleteApplication 代碼部署：Config DeleteDeployment 程式碼部署:群DeleteDeployment組 代碼部署：令牌 DeleteGit HubAccount 代碼部署：I DeleteResources ByExternal d 代碼部署：DeregisterOnPremisesInstance 代碼部署：GetApplication 代碼部署：修訂 GetApplication 代碼部署：GetDeployment

服務前綴	動作
	代碼部署：Config GetDeployment 程式碼部署:群GetDeployment組 代碼部署：GetDeployment實例 程式碼部署:目標 GetDeployment 代碼部署：GetOnPremisesInstance 程式碼部署:修訂 ListApplication 代碼部署：ListApplications 代碼部署：ListDeployment配置 程式碼部署:群ListDeployment組 代碼部署：ListDeployment實例 代碼部署：ListDeployments 程式碼部署:目標 ListDeployment 代碼部署：ListGitHubAccountTokenNames 代碼部署：ListOnPremisesInstances 代碼部署：PutLifecycleEventHookExecutionStatus 代碼部署：修訂 RegisterApplication 代碼部署：RegisterOnPremisesInstance 代碼部署：SkipWaitTimeForInstanceTermination 代碼部署：StopDeployment 代碼部署：UpdateApplication 程式碼部署:群UpdateDeployment組

服務前綴	動作
codeguru-profiler	代碼古魯分析器：頻道 AddNotification 程式碼大師效能分析工具：資料 BatchGet FrameMetric 代碼古魯分析器：ConfigureAgent 程式碼古組效能分析工具：群組 CreateProfiling 程式碼古組效能分析工具：群組 DeleteProfiling 程式碼古組效能分析工具：群組 DescribeProfiling 程式碼大師效能分析工具：摘要 GetFindings ReportAccount 程式碼大師效能分析工具：組態 GetNotification 代碼古魯分析器：GetPolicy 代碼古魯分析器：GetProfile 代碼古魯分析器：GetRecommendations 程式碼大師效能分析工具：報告 ListFindings 程式碼古組分析工具：時代 ListProfile 程式碼古組效能分析工具：群組 ListProfiling 代碼古魯分析器：PutPermission 程式碼古組分析工具:頻道 RemoveNotification 代碼古魯分析器：RemovePermission 代碼古魯分析器：SubmitFeedback 程式碼古組效能分析工具：群組 UpdateProfiling

服務前綴	動作
codeguru-reviewer	代碼庫-審閱者：AssociateRepository 代碼庫-審閱者：評論 CreateCode 代碼庫-審閱者：評論 DescribeCode 程式碼大師-檢閱者：意見反應 DescribeRecommendation 程式碼大師-檢閱者:關聯 DescribeRepository 代碼庫-審閱者：DisassociateRepository 代碼庫-審閱者：評論 ListCode 程式碼大師-檢閱者：意見反應 ListRecommendation 代碼庫-審閱者：ListRecommendations 程式碼大師-檢閱者：關聯 ListRepository 程式碼大師-檢閱者：意見反應 PutRecommendation

服務前綴	動作
codepipeline	代碼管道：AcknowledgeJob
	代碼管道：AcknowledgeThirdPartyJob
	代碼管道：CreateCustomActionType
	代碼管道：CreatePipeline
	代碼管道：DeleteCustomActionType
	代碼管道：DeletePipeline
	代碼管道：DeleteWebhook
	代碼管道：黨 DeregisterWebhook WithThird
	代碼管道：類型 GetAction
	代碼管道：GetJob詳細信息
	代碼管道：GetPipeline
	代碼管道：執GetPipeline行
	代碼管道：狀態 GetPipeline
	代碼管道：GetThirdPartyJob詳細信息
	代碼管道：執ListAction行
	代碼管道：類型 ListAction
	代碼管道：執ListPipeline行
	代碼管道：ListPipelines
	代碼管道：ListWebhooks
	程式碼管線:PollFor工作
	程式碼管線:PollForThirdParty工作

服務前綴	動作
	代碼管道：修訂 PutAction
	程式碼管線：PutApproval結果
	代碼管道：PutJobFailureResult
	代碼管道：PutJobSuccessResult
	代碼管道：PutThirdPartyJobFailureResult
	代碼管道：PutThirdPartyJobSuccessResult
	代碼管道：PutWebhook
	代碼管道：黨 RegisterWebhook WithThird
	代碼管道：RollbackStage
	代碼管道：執StartPipeline行
	代碼管道：執StopPipeline行
	代碼管道：類型 UpdateAction
	代碼管道：UpdatePipeline

服務前綴	動作
codestar	代碼星:AssociateTeam成員 代碼星：CreateProject 代碼星：CreateUser配置文件 代碼星：DeleteProject 代碼星：DeleteUser配置文件 代碼星：DescribeProject 代碼星：DescribeUser配置文件 代碼星:DisassociateTeam成員 代碼星：ListProjects 代碼星：ListResources 代碼之星:ListTeam成員 代碼星：ListUser配置文件 代碼星：UpdateProject 代碼星:UpdateTeam成員 代碼星：UpdateUser配置文件

服務前綴	動作
codestar-notifications	代碼星通知：規則 CreateNotification
	代碼星通知：規則 DeleteNotification
	代碼星通知：DeleteTarget
	代碼星通知：規則 DescribeNotification
	代碼星通知：類型 ListEvent
	代碼星通知：規則 ListNotification
	代碼星通知：ListTargets
	codestar-notifications:Subscribe
	codestar-notifications:Unsubscribe
	代碼星通知：規則 UpdateNotification

服務前綴	動作
cognito-identity	身份識別：池 CreateIdentity 認知身份:DeleteIdentities 身份識別：池 DeleteIdentity 認知身份:DescribeIdentity 身份識別：池 DescribeIdentity 認知身份:GetIdentityPoolRoles 認知身份:ListIdentities 身份識別：池 ListIdentity 身份識別：身份 LookupDeveloper 身份識別：身份 MergeDeveloper 認知身份:SetIdentityPoolRoles 身份識別：身份 UnlinkDeveloper 身份識別：池 UpdateIdentity

服務前綴	動作
cognito-idp	知識-IDP：屬性 AddCustom
	知識庫尼托-IDP: 集團 AdminAdd UserTo
	印度尼托-IDP: AdminConfirm SignUp
	已知的 IDP: 使用者 AdminCreate
	已知的 IDP: 使用者 AdminDelete
	印度尼托-IDP: AdminDelete UserAttributes
	已知的 IDP: 使用者 AdminDisable ProviderFor
	已知的 IDP: 使用者 AdminDisable
	已知的 IDP: 使用者 AdminEnable
	已知的 IDP: 設備 AdminForget
	已知的 IDP: 設備 AdminGet
	已知的 IDP: 使用者 AdminGet
	認知 IDP：身份驗證 AdminInitiate
	已知的 IDP: 使用者 AdminLink ProviderFor
	已知的 IDP: 設備 AdminList
	已知的 IDP: 使用者 AdminList GroupsFor
	知識庫尼托-IDP: 事件 AdminList UserAuth
	知識庫尼托-IDP: 集團 AdminRemove UserFrom
	印度尼托-IDP: AdminReset UserPassword
	認知識-IDP: 挑戰 AdminRespond ToAuth
	已知 IDP: 使用者作業參考 AdminSet

服務前綴	動作
	印度尼托-IDP: AdminSet UserPassword
	印度尼托-IDP: AdminSet UserSettings
	知識庫尼托-IDP: 反饋 AdminUpdate AuthEvent
	印度尼托-IDP: AdminUpdate DeviceStatus
	印度尼托-IDP: AdminUpdate UserAttributes
	知識-IDP: 出 AdminUser GlobalSign
	知識庫尼托-IDP：令牌 AssociateSoftware
	印度尼托-IDP: ChangePassword
	印度尼托-IDP: ConfirmDevice
	認證 IDP: 密碼 ConfirmForgot
	知識庫尼托-IDP：向上 ConfirmSign
	印度尼托-IDP: CreateGroup
	認識 IDP: 供應商 CreateIdentity
	已知的 IDP: 伺服器 CreateResource
	印度尼托-IDP: CreateUser ImportJob
	認識 IDP: 游泳池 CreateUser
	印度尼托-IDP: CreateUser PoolClient
	印度尼托-IDP: CreateUser PoolDomain
	印度尼托-IDP: DeleteGroup
	認識 IDP: 供應商 DeleteIdentity
	已知的 IDP: 伺服器 DeleteResource

服務前綴	動作
	印度尼托-IDP: DeleteUser
	知識-IDP：屬性 DeleteUser
	認識 IDP: 游泳池 DeleteUser
	印度尼托-IDP: DeleteUser PoolClient
	印度尼托-IDP: DeleteUser PoolDomain
	認識 IDP: 供應商 DescribeIdentity
	已知的 IDP: 伺服器 DescribeResource
	知識-IDP：配置 DescribeRisk
	印度尼托-IDP: DescribeUser ImportJob
	認識 IDP: 游泳池 DescribeUser
	印度尼托-IDP: DescribeUser PoolClient
	印度尼托-IDP: DescribeUser PoolDomain
	印度尼托-IDP: ForgetDevice
	印度尼托-IDP: ForgotPassword
	cognito-idp:GetCSVHeader
	印度尼托-IDP: GetDevice
	印度尼托-IDP: GetGroup
	認識 IDP: 識別碼 GetIdentity ProviderBy
	印度尼托-IDP: GetLog DeliveryConfiguration
	知識產權-IDP: 證書 GetSigning
	cognito-idp:GetUICustomization

服務前綴	動作
	印度尼托-IDP: GetUser
	認知識-IDP：代碼 GetUser AttributeVerification
	知識-IDP：Config GetUser PoolMfa
	知識-IDP: 輸出 GlobalSign
	印度尼托-IDP: InitiateAuth
	印度尼托-IDP: ListDevices
	印度尼托-IDP: ListGroups
	認識 IDP: 供應商 ListIdentity
	認證 IDP: 伺服器 ListResource
	印度尼托-IDP: ListUser ImportJobs
	印度尼托-IDP: ListUser PoolClients
	知識庫尼托-IDP: 池 ListUser
	印度尼托-IDP: ListUsers
	印度尼托-IDP: ListUsers InGroup
	認知識-IDP：代碼 ResendConfirmation
	印度尼托-IDP: RespondTo AuthChallenge
	印度尼托-IDP: RevokeToken
	印度尼托-IDP: SetLog DeliveryConfiguration
	知識-IDP：配置 SetRisk
	cognito-idp:SetUICustomization
	認知 IDP: 多功能打印機參考 SetUser

服務前綴	動作
	知識-IDP：Config SetUser PoolMfa
	知識-IDP: 設置 SetUser
	印度尼托-IDP: SignUp
	印度尼托-IDP: StartUser ImportJob
	印度尼托-IDP: StopUser ImportJob
	印度尼托-IDP: UpdateAuth EventFeedback
	已知的 IDP：狀態 UpdateDevice
	印度尼托-IDP: UpdateGroup
	認識 IDP: 供應商 UpdateIdentity
	已知的 IDP: 伺服器 UpdateResource
	知識-IDP：屬性 UpdateUser
	認識 IDP: 游泳池 UpdateUser
	印度尼托-IDP: UpdateUser PoolClient
	印度尼托-IDP: UpdateUser PoolDomain
	知識庫尼托-IDP：令牌 VerifySoftware
	知識庫尼托-IDP：屬性 VerifyUser

服務前綴	動作
cognito-sync	齒輪同步：BulkPublish 齒輪同步：DeleteDataset 齒輪同步：DescribeDataset 齒輪同步：DescribeIdentityPoolUsage 齒輪同步：用法 DescribeIdentity 齒輪同步：GetBulkPublishDetails 齒輪同步：事件 GetCognito 齒輪同步：GetIdentityPoolConfiguration 齒輪同步：ListDatasets 齒輪同步：ListIdentityPoolUsage 齒輪同步：ListRecords 齒輪同步：RegisterDevice 齒輪同步：事件 SetCognito 齒輪同步：SetIdentityPoolConfiguration 齒輪同步：數據集 SubscribeTo 齒輪同步：數據集 UnsubscribeFrom 齒輪同步：UpdateRecords

服務前綴	動作
comprehendmedical	醫療領域:檢測 V2Job DescribeEntities 理解醫學:描述 10 厘米 InferenceJob 理解医学:形容贝 DetectionJob 醫療領域:Job DescribeRx NormInference 理解医学:说明 InferenceJob 醫療領域:V2 DetectEntities comprehendmedical:DetectPHI comprehendmedical:InferICD10CM 理解醫療：規範 InferRx comprehendmedical:InferSNOMEDCT 醫學理解:檢測 V2jobs ListEntities 理解醫學:列表 10 厘米 InferenceJobs 理解医学:列斯特皮 DetectionJobs 醫療領域:工作 ListRx NormInference 理解医学:名单 InferenceJobs 醫療領域:檢測 V2Job StartEntities 理解醫學:星光 10 厘米 InferenceJob 理解醫學:起始 DetectionJob 醫療領域:Job StartRx NormInference 理解醫學:起步 InferenceJob 醫療領域:檢測 V2Job StopEntities

服務前綴	動作
	理解醫療:停止 10 厘米 InferenceJob 理解医学:斯托皮 DetectionJob 醫療領域:Job StopRx NormInference 理解医学:停止 InferenceJob
compute-optimizer	運算最佳化器：偏好設定 DeleteRecommendation 計算優化器：DescribeRecommendationExportJobs 運算最佳化工具：建議 ExportAuto ScalingGroup 運算最佳化程式:匯出 VolumeRecommendations 计算优化器:出口 C2 InstanceRecommendations 運算最佳化程式:匯出 ServiceRecommendations 計算優化器：ExportLambdaFunctionRecommendations 運算最佳化工具：建議 ExportLicense 運算RecommendationProjected最佳化器:取得 C2 度量 計算最佳化器:極限 ServiceRecommendation ProjectedMetrics 計算優化器：GetEffectiveRecommendationPreferences 運算最佳化程式：狀態 GetEnrollment 運算最佳化程式：組織 GetEnrollment StatusesFor 運算最佳化器：偏好設定 GetRecommendation 運算最佳化程式：摘要 GetRecommendation 運算最佳化器：偏好設定 PutRecommendation 運算最佳化程式：狀態 UpdateEnrollment

服務前綴	動作
config	配置：BatchGetResourceConfig
	配置：DeleteAggregation授權
	配置：DeleteConfig規則
	配置：DeleteConfiguration聚合
	配置：DeleteConfiguration錄音機
	配置：DeleteConformance包
	配置：DeleteDelivery頻道
	配置：DeleteEvaluation結果
	配置：DeleteOrganizationConfigRule
	配置：DeleteOrganizationConformancePack
	配置：DeletePendingAggregationRequest
	配置：DeleteRemediation配置
	配置：DeleteRemediation例外
	Config：DeleteResource配置
	配置：DeleteRetention配置
	配置：DeleteStored查詢
	配置：DeliverConfig快照
	配置：DescribeAggregateComplianceByConfigRules
	配置：DescribeAggregateComplianceByConformancePacks
	配置：DescribeAggregation授權
	配置：DescribeComplianceByConfig規則

服務前綴	動作
	配置：DescribeComplianceByResource
	配置：DescribeConfigRuleEvaluation狀態
	配置：DescribeConfig規則
	配置：DescribeConfiguration聚合器
	配置：DescribeConfigurationAggregatorSources狀態
	配置：DescribeConfiguration記錄器
	配置：DescribeConfigurationRecorderStatus
	配置：DescribeConformancePackCompliance
	配置：DescribeConformance包
	配置：DescribeConformancePackStatus
	配置：DescribeDelivery頻道
	配置：DescribeDeliveryChannelStatus
	配置：DescribeOrganizationConfigRules
	配置：DescribeOrganizationConfigRule狀態
	配置：DescribeOrganizationConformancePacks
	配置：DescribeOrganizationConformancePack狀態
	配置：DescribePendingAggregationRequests
	配置：DescribeRemediation配置
	配置：DescribeRemediation例外
	配置：DescribeRemediationExecutionStatus
	配置：DescribeRetention配置

服務前綴	動作
	配置：GetComplianceDetailsByConfigRule
	配置：GetComplianceDetailsBy資源
	配置：GetComplianceSummaryByConfigRule
	配置：GetComplianceSummaryByResourceType
	配置：GetConformancePackCompliance詳細信息
	配置：GetConformancePackCompliance摘要
	配置：GetCustomRulePolicy
	配置：GetDiscoveredResourceCounts
	配置：GetOrganizationConfigRuleDetailedStatus
	配置：GetOrganizationConformancePackDetailedStatus
	配置：GetOrganizationCustomRule策略
	配置：GetResourceConfigHistory
	配置：GetResourceEvaluationSummary
	配置：GetStored查詢
	配置：ListConformancePackCompliance分數
	配置：ListDiscovered資源
	配置：ListResource評估
	配置：ListStored查詢
	配置：PutConfig規則
	配置：PutConfiguration聚合
	配置：PutConfiguration錄音機

服務前綴	動作
	配置：PutConformance包
	配置：PutDelivery頻道
	配置：PutEvaluations
	配置：PutExternal評估
	配置：PutOrganizationConfigRule
	配置：PutOrganizationConformancePack
	配置：PutRemediation配置
	配置：PutRemediation例外
	Config：PutResource配置
	配置：PutRetention配置
	配置：PutStored查詢
	Config：SelectResource配置
	配置：StartConfigRulesEvaluation
	配置：StartConfiguration錄音機
	配置：StartRemediation執行
	配置：StartResource評估
	配置：StopConfiguration錄音機

服務前綴	動作
connect	連接：ActivateEvaluation表單 連接：AssociateApproved原點 連接：AssociateBot 連接：AssociateDefault詞彙 連接：AssociateFlow 連接：AssociateInstanceStorageConfig 連接：AssociateLambda功能 連接：AssociateLex機器人 連接：AssociatePhoneNumberContact流 連接：AssociateQueueQuickConnects 連接：AssociateRoutingProfileQueues 連接:AssociateSecurity鍵 連接:AssociateUser精通 連接：BatchGetFlowAssociation 連接：BatchPut聯繫 連接：ClaimPhone數 連線：CreateAgent狀態 連接：CreateContact流 連接：CreateContactFlowModule 連接：CreateEvaluation表單 連接：CreateHoursOfOperation

服務前綴	動作
	連接：CreateInstance 連接:CreateIntegration關聯 連接：CreateParticipant 連接：CreatePersistentContactAssociation 連接：CreatePredefined屬性 連接：CreatePrompt 連接：CreateQueue Connect：CreateQuick連接 連線：CreateRouting設定檔 連接：CreateRule 連線：CreateSecurity設定檔 連線:CreateTask範本 連接：CreateTrafficDistributionGroup 連接：CreateUse案例 連接：CreateUser 連接：CreateUserHierarchyGroup 連接：CreateView 連接：CreateView版本 連接：CreateVocabulary 連接：DeactivateEvaluation表單 連接：DeleteContact評估

服務前綴	動作
	連接：DeleteContact流 連接：DeleteContactFlowModule 連接：DeleteEvaluation表單 連接：DeleteHoursOfOperation 連接：DeleteInstance 連接:DeleteIntegration關聯 連接：DeletePredefined屬性 連接：DeletePrompt 連接：DeleteQueue Connect：DeleteQuick連接 連線：DeleteRouting設定檔 連接：DeleteRule 連線：DeleteSecurity設定檔 連線:DeleteTask範本 連接：DeleteTrafficDistributionGroup 連接：DeleteUse案例 連接：DeleteUser 連接：DeleteUserHierarchyGroup 連接：DeleteView 連接：DeleteVocabulary 連線：DescribeAgent狀態

服務前綴	動作
	連接：DescribeContact評估 連接：DescribeContact流 連接：DescribeContactFlowModule 連接：DescribeEvaluation表單 連接：DescribeInstance屬性 連接：DescribeInstanceStorageConfig 連接：DescribePhone數 連接：DescribeRule 連接：DescribeTrafficDistributionGroup 連接：DescribeUserHierarchyGroup 連接：DescribeUserHierarchyStructure 連接：DescribeView 連接：DescribeVocabulary 連接：DisassociateApproved原點 連接：DisassociateBot 連接：DisassociateFlow 連接：DisassociateInstanceStorageConfig 連接：DisassociateLambda功能 連接：DisassociateLex機器人 連接：DisassociatePhoneNumberContact流 連接：DisassociateQueueQuickConnects

服務前綴	動作
	連接：DisassociateRoutingProfileQueues 連接:DisassociateSecurity鍵 連接:DisassociateUser精通 連接：DismissUser聯繫 連接：GetContact屬性 連接：GetCurrentMetricData 連接：GetCurrentUserData 連接：GetFederation令牌 連接:GetFlow關聯 連接:GetMetric資料 連線:GetMetric資料影像 連接：GetPrompt文件 連線:GetTask範本 連接:GetTraffic分配 連接：ImportPhone數 連接:ListApproved起源 連接：ListBots 連線：ListContact評估 連接：ListContactFlowModules 連線：ListContact流程 連接：ListContact參考

服務前綴	動作
	連接：ListDefault詞彙 連接：ListEvaluation表格 連接：ListEvaluationFormVersions 連線:ListFlow關聯 連接：ListHoursOfOperations 連接：ListInstance屬性 連接：ListInstanceStorageConfigs 連線:ListIntegration關聯 連接：ListLambda功能 連接:ListLex機器人 連接：ListPhone數字 連接：ListPhone編號 連接：ListPredefined屬性 連接：ListPrompts 連接：ListQueueQuickConnects 連接：ListQueues ListQuick連接：連接 連接：ListRealtimeContactAnalysis段 V2 連接：ListRoutingProfileQueues 連線：ListRouting設定檔 連接：ListRules

服務前綴	動作
	連接：ListSecurity鑰匙 連接：ListSecurityProfileApplications 連接：ListSecurityProfilePermissions 連線：ListSecurity設定檔 連線:ListTask範本 連接：ListTrafficDistributionGroups 連接：ListUse案例 連接：ListUserHierarchyGroups 連接:ListUser精通 連接：ListUsers 連接：ListViews 連接：ListView版本 連接：MonitorContact 連接：PauseContact 連線：PutUser狀態 連接：ReleasePhone數 連接：ReplicateInstance 連接：ResumeContact 連接：ResumeContact錄音 連接：SearchAvailablePhoneNumbers 連接：SearchContacts

服務前綴	動作
	連接：SearchHoursOfOperations 連接：SearchPredefined屬性 連接：SearchPrompts 連接：SearchQueues SearchQuick連接：連接 連線：SearchRouting設定檔 連線：SearchSecurity設定檔 連接：SearchVocabularies 連接：SendChatIntegrationEvent 連接：StartChat聯繫 連接：StartContact評估 連接：StartContact錄音 連線：StartContact串流 連接：StartOutboundVoiceContact 連接：StartTask聯繫 連接：StartWebRTC 聯繫 連接：StopContact 連接：StopContact錄音 連線：StopContact串流 連接：SubmitContact評估 連接：SuspendContact錄音

服務前綴	動作
	連接：TransferContact 連線：UpdateAgent狀態 連接：UpdateContact 連接：UpdateContact屬性 連接：UpdateContact評估 連接：UpdateContactFlowContent 連接：UpdateContactFlowMetadata 連接:UpdateContactFlowModule內容 連線:UpdateContactFlowModule中繼資料 連接：UpdateContactFlowName 連接：UpdateContactRoutingData 連線：UpdateContact排程 連接：UpdateEvaluation表單 連接：UpdateHoursOfOperation 連接：UpdateInstance屬性 連接：UpdateInstanceStorageConfig 連接：UpdateParticipantRoleConfig 連接：UpdatePhone數 連接：UpdatePhoneNumberMetadata 連接：UpdatePredefined屬性 連接：UpdatePrompt

服務前綴	動作
	連接：UpdateQueueHoursOf操作 連接：UpdateQueueMaxContacts 連接：UpdateQueue名稱 連接：UpdateQueueOutboundCallerConfig 連線：UpdateQueue狀態 連接：UpdateQuickConnectConfig 連接：UpdateQuickConnectName 連接：UpdateRoutingProfileAgentAvailabilityTimer 連接：UpdateRoutingProfileConcurrency 連接：UpdateRoutingProfileDefaultOutboundQueue 連接：UpdateRoutingProfileName 連接：UpdateRoutingProfileQueues 連接：UpdateRule 連線：UpdateSecurity設定檔 連線:UpdateTask範本 連接:UpdateTraffic分配 連線：UpdateUser階層 連接：UpdateUserHierarchyGroup名稱 連接：UpdateUserHierarchyStructure 連接：UpdateUserIdentityInfo 連接：UpdateUserPhoneConfig

服務前綴	動作
	連接:UpdateUser精通 連接：UpdateUserRoutingProfile 連接：UpdateUserSecurityProfiles 連接:UpdateView內容 連線:UpdateView中繼資料
cur	庫爾：DeleteReport定義 庫爾：DescribeReport定義 庫爾：ModifyReport定義 庫爾：PutReport定義

服務前綴	動作
databrew	數據庫：BatchDeleteRecipeVersion 數據庫：CreateDataset 資料庫：Job CreateProfile 數據庫：CreateProject 數據庫：CreateRecipe 資料庫：Job CreateRecipe 數據庫：CreateRuleset 數據庫：CreateSchedule 數據庫：DeleteDataset 數據庫：DeleteJob 數據庫：DeleteProject 資料庫:版本 DeleteRecipe 數據庫：DeleteRuleset 數據庫：DeleteSchedule 數據庫：DescribeDataset 數據庫：DescribeJob 數據庫：運行 DescribeJob 數據庫：DescribeProject 數據庫：DescribeRecipe 數據庫：DescribeRuleset 數據庫：DescribeSchedule

服務前綴	動作
	數據庫：ListDatasets 數據庫：運行 ListJob 數據庫：ListJobs 數據庫：ListProjects 數據庫：ListRecipes 資料庫：版本 ListRecipe 數據庫：ListRulesets 數據庫：ListSchedules 數據庫：PublishRecipe 數據庫：SendProjectSessionAction 數據庫：運行 StartJob 資料庫：工作階段 StartProject 數據庫：運行 StopJob 數據庫：UpdateDataset 資料庫：Job UpdateProfile 數據庫：UpdateProject 數據庫：UpdateRecipe 資料庫：Job UpdateRecipe 數據庫：UpdateRuleset 數據庫：UpdateSchedule

服務前綴	動作
dataexchange	數據交換：CancelJob
	數據交換:CreateData設置
	資料交換:CreateEvent動作
	數據交換：CreateJob
	數據交換：CreateRevision
	數據交換：DeleteAsset
	資料交換:DeleteEvent動作
	數據交換：DeleteRevision
	資料交換:GetEvent動作
	數據交換：GetJob
	數據交換：ListDataSetRevisions
	數據交換:ListData集
	資料交換:ListEvent動作
	數據交換：ListJobs
	資料交換:ListRevision資產
	數據交換：RevokeRevision
	數據交換：SendDataSetNotification
	數據交換：StartJob
	數據交換：UpdateAsset
	數據交換:UpdateData設置
	資料交換:UpdateEvent動作

服務前綴	動作
	數據交換：UpdateRevision
datapipeline	數據帕林:ActivatePipeline 數據帕林:CreatePipeline 數據帕林:DeactivatePipeline 數據帕林:DeletePipeline 數據帕林:DescribeObjects 數據帕林:DescribePipelines 數據帕林:EvaluateExpression 數據帕林：定義 GetPipeline 數據帕林:ListPipelines 資料替補線：任務 PollFor 數據帕林：定義 PutPipeline 數據帕林:QueryObjects 數據帕林:進展 ReportTask 數據帕林:ReportTaskRunnerHeartbeat 數據帕林:SetStatus 資料替補線:狀態 SetTask 數據帕林：定義 ValidatePipeline

服務前綴	動作
dax	達克斯：CreateCluster 達克斯：DecreaseReplication因素 達克斯：DeleteCluster 達克斯:DeleteParameter集團 達克斯:DeleteSubnet集團 達克斯：DescribeClusters 達克斯：DescribeDefault參數 達克斯：DescribeEvents 達克斯：DescribeParameter組 達克斯：DescribeParameters 達克斯：DescribeSubnet組 達克斯：IncreaseReplication因素 達克斯：RebootNode 達克斯：UpdateCluster 達克斯:UpdateParameter集團 達克斯:UpdateSubnet集團

服務前綴	動作
devicefarm	裝置:池 CreateDevice 設備：配置文件 CreateInstance 設備：配置文件 CreateNetwork 裝置:CreateProject 裝置:CreateRemoteAccessSession 裝置:CreateTestGridProject 裝置:CreateTestGridUrl 裝置:CreateUpload devicefarm:CreateVPCEConfiguration 裝置:池 DeleteDevice 設備：配置文件 DeleteInstance 設備：配置文件 DeleteNetwork 裝置:DeleteProject 裝置:DeleteRemoteAccessSession 裝置:DeleteRun 裝置:DeleteTestGridProject 裝置:DeleteUpload devicefarm:DeleteVPCEConfiguration 設備：設置 GetAccount 裝置:GetDevice 設備：實例 GetDevice

服務前綴	動作
	裝置:池 GetDevice 裝置:GetDevicePoolCompatibility 設備：配置文件 GetInstance 裝置:GetJob 設備：配置文件 GetNetwork 裝置：狀態 GetOffering 裝置:GetProject 裝置:GetRemoteAccessSession 裝置:GetRun 裝置:GetSuite 裝置:GetTest 裝置:GetTestGridProject 裝置:GetTestGridSession 裝置:GetUpload devicefarm:GetVPCEConfiguration 裝置:ListArtifacts 設備：實例 ListDevice 設備:池 ListDevice 裝置:ListDevices 設備：配置文件 ListInstance 裝置:ListJobs

服務前綴	動作
	設備：配置文件 ListNetwork 裝置:促銷 ListOffering 裝置:ListOfferings 設備：交易 ListOffering 裝置:ListProjects 裝置:ListRemoteAccessSessions 裝置:ListRuns 裝置:ListSamples 裝置:ListSuites 裝置:ListTestGridProjects 裝置:動作 ListTest GridSession 裝置：文物 ListTest GridSession 裝置:ListTestGridSessions 裝置:ListTests 裝置：問題 ListUnique 裝置:ListUploads devicefarm:ListVPCEConfigurations 裝置:PurchaseOffering 裝置:RenewOffering 裝置:ScheduleRun 裝置:StopJob

服務前綴	動作
	裝置:StopRemoteAccessSession 裝置:StopRun 設備：實例 UpdateDevice 裝置:池 UpdateDevice 設備：配置文件 UpdateInstance 設備：配置文件 UpdateNetwork 裝置:UpdateProject 裝置:UpdateTestGridProject 裝置:UpdateUpload devicefarm:UpdateVPCEConfiguration

服務前綴	動作
devops-guru	開發大師:頻道 AddNotification
	開發大師：DeleteInsight
	開發大師：Health DescribeAccount
	開發大師：概述 DescribeAccount
	開發大師：DescribeAnomaly
	開發大師：DescribeEventSourcesConfig
	開發大師：DescribeFeedback
	開發大師：DescribeInsight
	開發大師：Health DescribeOrganization
	開發大師：概述 DescribeOrganization
	開發大師：Health DescribeOrganization ResourceCollection
	開發大師：DescribeResourceCollectionHealth
	開發大師：整合 DescribeService
	開發大師:估計 GetCost
	開發大師：收藏 GetResource
	開發大師：ListAnomaliesForInsight
	開發大師：ListAnomalousLogGroups
	開發大師：ListEvents
	開發大師：ListInsights
	開發大師：資源 ListMonitored
	開發大師：頻道 ListNotification

服務前綴	動作
	開發大師：洞察 ListOrganization
	開發大師：ListRecommendations
	開發大師：PutFeedback
	開發大師:頻道 RemoveNotification
	開發大師：SearchInsights
	開發大師：洞察 SearchOrganization
	開發大師:估計 StartCost
	開發大師：UpdateEventSourcesConfig
	開發大師：收藏 UpdateResource
	開發大師：整合 UpdateService

服務前綴	動作
directconnect	直接連接：AcceptDirectConnectGatewayAssociationProposal 直接連接：AllocateConnectionOnInterconnect 直接連接：連接 AllocateHosted 直接連接：AllocatePrivateVirtualInterface 直接連接：AllocatePublicVirtualInterface 直接連接：AllocateTransitVirtualInterface 直接連接：AssociateConnectionWithLag 直接連接：連接 AssociateHosted 直接連接：AssociateMacSecKey 直接連接：接口 AssociateVirtual 直接連接：ConfirmConnection 直接連接：協議 ConfirmCustomer 直接連接：ConfirmPrivateVirtualInterface 直接連接：ConfirmPublicVirtualInterface 直接連接：ConfirmTransitVirtualInterface directconnect:CreateBGPPeer 直接連接：CreateConnection 直接連接：CreateDirectConnectGateway 直接連接：關聯 CreateDirect ConnectGateway 直接連接：CreateDirectConnectGatewayAssociationProposal 直接連接：CreateInterconnect

服務前綴	動作
	直接連接：CreateLag 直接連接：CreatePrivateVirtualInterface 直接連接：CreatePublicVirtualInterface 直接連接：CreateTransitVirtualInterface directconnect:DeleteBGPPeer 直接連接：DeleteConnection 直接連接：DeleteDirectConnectGateway 直接連接：關聯 DeleteDirect ConnectGateway 直接連接：DeleteDirectConnectGatewayAssociationProposal 直接連接：DeleteInterconnect 直接連接：DeleteLag 直接連接：接口 DeleteVirtual 直接連接：Loa DescribeConnection 直接連接：DescribeConnections 直接連接：DescribeConnectionsOnInterconnect 直接連接:中繼資料 DescribeCustomer 直接連接：DescribeDirectConnectGatewayAssociationProposals 直接連接:關聯 DescribeDirect ConnectGateway 直接連接：附件 DescribeDirect ConnectGateway 直接連接：DescribeDirectConnectGateways 直接連接：連接 DescribeHosted

服務前綴	動作
	直接連接：Loa DescribeInterconnect 直接連接：DescribeInterconnects 直接連接：DescribeLags 直接連接：DescribeLoa 直接連接：DescribeLocations 直接連接：配置 DescribeRouter 直接連接：閘道 DescribeVirtual 直接連接：接口 DescribeVirtual 直接連接：DisassociateConnectionFromLag 直接連接：DisassociateMacSecKey 直接連接：歷史 ListVirtual InterfaceTest 直接連接：StartBgpFailoverTest 直接連接：StopBgpFailoverTest 直接連接：UpdateConnection 直接連接：UpdateDirectConnectGateway 直接連接：關聯 UpdateDirect ConnectGateway 直接連接：UpdateLag 直接連接：UpdateVirtualInterfaceAttributes

服務前綴	動作
dlm	DLM: CreateLifecycle 政策
	DLM: DeleteLifecycle 政策
	DLM: GetLifecycle 政策
	DLM: GetLifecycle 政策
	DLM: UpdateLifecycle 政策

服務前綴	動作
dms	dms: ApplyPending MaintenanceAction dms: BatchStart 建議使用 dms: CancelReplication TaskAssessment 執行 dms: CreateData 供應商 dms: CreateEndpoint dms: 訂閱 CreateEvent dms: CreateInstance 設定檔 dms: CreateMigration 專案 dms: Config CreateReplication dms: CreateReplication 執行個體 dms: CreateReplication SubnetGroup dms: CreateReplication 任務 dms: DeleteCertificate dms: DeleteConnection dms: DeleteData 供應商 dms: DeleteEndpoint dms: 訂閱 DeleteEvent dms: DeleteFleet AdvisorCollector dms: DeleteFleet AdvisorDatabases dms: DeleteInstance 設定檔 dms: DeleteMigration 專案

服務前綴	動作
	dms: Config DeleteReplication dms: DeleteReplication 執行個體 dms: DeleteReplication SubnetGroup dms: DeleteReplication 任務 dms: DeleteReplication TaskAssessment 執行 dms: DescribeAccount 屬性 dms: DescribeApplicable IndividualAssessments dms: DescribeCertificates dms: DescribeConnections dms: DescribeEndpoints dms: DescribeEndpoint 設定 dms: 類型 DescribeEndpoint dms: DescribeEngine 版本 dms: 產品類別 DescribeEvent dms: DescribeEvents dms: 訂閱 DescribeEvent dms: DescribeFleet AdvisorCollectors dms: DescribeFleet AdvisorDatabases dms: DescribeFleet AdvisorLsa 分析 dms: DescribeFleet AdvisorSchema ObjectSummary dms: DescribeFleet AdvisorSchemas

服務前綴	動作
	dms: DescribeMetadata ModelImports dms: DescribeOrderable ReplicationInstances dms: DescribePending MaintenanceActions dms: DescribeRecommendation 限制 dms: DescribeRecommendations dms: DescribeRefresh SchemasStatus dms: DescribeReplication 組態設定 dms: DescribeReplication 執行個體 dms: DescribeReplication InstanceTask 記錄檔 dms: DescribeReplications dms: DescribeReplication SubnetGroups dms: DescribeReplication TableStatistics dms: DescribeReplication TaskAssessment 結果 dms: DescribeReplication TaskAssessment 執行中 dms: 評估 DescribeReplication TaskIndividual dms: DescribeReplication 任務 dms: DescribeSchemas dms: DescribeTable 統計資料 dms: ExportMetadata ModelAssessment dms: 型GetMetadata號 dms: ImportCertificate

服務前綴	動作
	dms: ListMetadata ModelAssessment ActionItems dms: ModifyEndpoint dms: 訂閱 ModifyEvent dms: Config ModifyReplication dms: ModifyReplication 執行個體 dms: ModifyReplication SubnetGroup dms: ModifyReplication 任務 dms: MoveReplication 任務 dms: RebootReplication 執行個體 dms: RefreshSchemas dms: ReloadReplication 表格 dms: ReloadTables dms: RunFleet AdvisorLsa 分析 dms: StartMetadata ModelAssessment dms: StartMetadata ModelConversion dms: StartMetadata ModelExport ToTarget dms: StartRecommendations dms: StartReplication dms: StartReplication 任務 dms: StartReplication TaskAssessment dms: StopReplication 任務

服務前綴	動作
	dms: TestConnection dms: UpdateSubscriptions ToEvent 橋接器
docdb-elastic	文檔彈性：快照 CopyCluster 文檔彈性：DeleteCluster 文檔彈性：快照 DeleteCluster 文檔彈性：GetCluster 文檔彈性：快照 GetCluster 文檔彈性：ListClusters 文檔彈性：快照 ListCluster 文檔彈性：RestoreClusterFromSnapshot 文檔彈性：StartCluster 文檔彈性：StopCluster 文檔彈性：UpdateCluster

服務前綴	動作
ds	ds: AcceptShared 目錄 ds: AddIp 路線 ds: AddRegion ds: CancelSchema 擴充功能 ds: ConnectDirectory ds: CreateAlias ds: CreateComputer ds: CreateConditional 貨運代理 ds: CreateDirectory ds: CreateLog 訂閱 ds: CreateMicrosoft 廣告 ds: CreateSnapshot ds: CreateTrust ds: DeleteConditional 貨運代理 ds: DeleteDirectory ds: DeleteLog 訂閱 ds: DeleteSnapshot ds: DeleteTrust ds: DeregisterCertificate ds: DeregisterEvent 主題 ds: DescribeCertificate

服務前綴	動作
	ds: DescribeClient AuthenticationSettings ds: DescribeConditional 貨運代理 ds: DescribeDirectories ds: DescribeDomain 控制器 ds: DescribeEvent 主題 ds:DescribeLDAPSSettings ds: DescribeRegions ds: DescribeSettings ds: DescribeShared 目錄 ds: DescribeSnapshots ds: DescribeTrusts ds: DescribeUpdate 目錄 ds: DisableClient 身份驗證 ds:DisableLDAPS ds: DisableRadius ds: DisableSso ds: EnableClient 身份驗證 ds:EnableLDAPS ds: EnableRadius ds: EnableSso ds: GetDirectory 限制

服務前綴	動作
	ds: GetSnapshot 限制 ds: ListCertificates ds: ListIp 路線 ds: ListLog 訂閱 ds: ListSchema 擴充功能 ds: RegisterCertificate ds: RegisterEvent 主題 ds: RejectShared 目錄 ds: RemoveIp 路線 ds: RemoveRegion ds: ResetUser 密碼 ds: RestoreFrom 快照 ds: ShareDirectory ds: StartSchema 擴充功能 ds: UnshareDirectory ds: UpdateConditional 貨運代理 ds: UpdateDirectory 安裝程式 ds: UpdateNumber OfDomain 控制器 ds: UpdateRadius ds: UpdateSettings ds: UpdateTrust

服務前綴	動作
	ds: VerifyTrust

服務前綴	動作
dynamodb	動態：CreateBackup
	動態:表 CreateGlobal
	動態：CreateTable
	動態：DeleteBackup
	動態：DeleteTable
	動態：DescribeBackup
	動態:備份 DescribeContinuous
	動態:洞察 DescribeContributor
	動態：DescribeEndpoints
	動態：DescribeExport
	動態:表 DescribeGlobal
	動態：DescribeGlobalTableSettings
	動態：DescribeImport
	動態：DescribeKinesisStreamingDestination
	動態：DescribeLimits
	動態：DescribeStream
	動態：DescribeTable
	動態:縮放 DescribeTable ReplicaAuto
	動態：DescribeTimeToLive
	動態：DisableKinesisStreamingDestination
	動態：EnableKinesisStreamingDestination

服務前綴	動作
	動態：ExportTableToPointInTime 動態:政策 GetResource 動態：ImportTable 動態：ListBackups 動態:洞察 ListContributor 動態：ListExports 動態:表 ListGlobal 動態：ListImports 動態：ListStreams 動態：ListTables 動態：RestoreTableFromBackup 動態：RestoreTableToPointInTime 動態:備份 UpdateContinuous 動態:洞察 UpdateContributor 動態:表 UpdateGlobal 動態：UpdateGlobalTableSettings 動態：UpdateKinesisStreamingDestination 動態：UpdateTable 動態:縮放 UpdateTable ReplicaAuto 動態：UpdateTimeToLive

服務前綴	動作
ebs	ebs：CompleteSnapshot ebs：StartSnapshot

服務前綴	動作
ec2	ec2：AcceptAddress轉移
	ec2：AcceptReservedInstancesExchange報價
	ec2：AcceptTransitGatewayMulticastDomainAssociations
	ec2：AcceptTransitGatewayPeering附件
	ec2：AcceptTransitGatewayVpc附件
	ec2：AcceptVpcEndpointConnections
	ec2：AcceptVpcPeeringConnection
	ec2：AdvertiseByoip西德
	ec2：AllocateAddress
	ec2：AllocateHosts
	ec2：AllocateIpamPoolCidr
	ec2：ApplySecurityGroupsToClientVpnTargetNetwork
	ec2：AssignIpv6 地址
	ec2：AssignPrivateIpAddresses
	ec2：AssignPrivateNatGateway地址
	ec2：AssociateAddress
	ec2：AssociateClientVpnTarget網絡
	ec2：AssociateDhcp選項
	ec2：AssociateEnclaveCertificateIam角色
	ec2：AssociateIamInstanceProfile
	ec2：AssociateInstanceEventWindow

服務前綴	動作
	ec2：自助AssociateIpam餐
	ec2：AssociateIpamResourceDiscovery
	ec2：AssociateNatGatewayAddress
	ec2：AssociateRoute表
	ec2：AssociateSubnetCidrBlock
	ec2: AssociateTransit GatewayMulticast 網域名稱
	ec2：AssociateTransitGatewayPolicy表
	ec2：AssociateTransitGatewayRoute表
	ec2：AssociateTrunk接口
	ec2：AssociateVpcCidrBlock
	ec2：AttachClassicLinkVpc
	ec2：AttachInternet網關
	ec2：AttachNetwork接口
	ec2：AttachVerifiedAccessTrust供應商
	ec2：AttachVolume
	ec2：AttachVpn網關
	ec2：AuthorizeClientVpnIngress
	ec2：AuthorizeSecurityGroupEgress
	ec2：AuthorizeSecurityGroupIngress
	ec2：BundleInstance
	ec2：CancelBundle任務

服務前綴	動作
	ec2: CancelCapacity 預約
	ec2：CancelCapacityReservationFleets
	ec2：CancelConversion任務
	ec2：CancelExport任務
	ec2：CancelImageLaunchPermission
	ec2：CancelImport任務
	ec2：CancelReservedInstancesListing
	ec2：CancelSpotFleetRequests
	ec2：CancelSpotInstanceRequests
	ec2：ConfirmProduct實例
	ec2：CopyFpga圖片
	ec2：CopyImage
	ec2：CopySnapshot
	ec2: CreateCapacity 預約
	ec2：CreateCapacityReservationFleet
	ec2：CreateCarrier網關
	ec2：CreateClientVpnEndpoint
	ec2：CreateClientVpnRoute
	ec2：CreateCoip西德
	ec2：CreateCoip游泳池
	ec2：CreateCustomer網關

服務前綴	動作
	ec2：CreateDefault子網
	ec2：CreateDefaultVPC
	ec2：CreateDhcp選項
	ec2：CreateEgressOnlyInternet網關
	ec2：CreateFleet
	ec2：CreateFlow日誌
	ec2：CreateFpga圖片
	ec2：CreateImage
	ec2：CreateInstanceConnectEndpoint
	ec2：CreateInstanceEventWindow
	ec2：CreateInstanceExportTask
	ec2：CreateInternet網關
	ec2：CreateIpam
	ec2：CreateIpam游泳池
	ec2：CreateIpamResourceDiscovery
	ec2：CreateIpam範圍
	ec2：CreateKey配對
	ec2：CreateLaunch模板
	ec2：CreateLaunchTemplateVersion
	ec2：CreateLocalGatewayRoute
	ec2：CreateLocalGatewayRoute表

服務前綴	動作
	ec2：CreateLocalGatewayRouteTableVirtualInterfaceGroup關聯
	ec2：CreateLocalGatewayRouteTableVpc關聯
	ec2：CreateManagedPrefixList
	ec2：CreateNat網關
	ec2：CreateNetwork十字韌帶
	ec2：CreateNetworkAclEntry
	ec2：CreateNetworkInsightsAccess範圍
	ec2：CreateNetworkInsightsPath
	ec2：CreateNetwork接口
	ec2：CreateNetworkInterfacePermission
	ec2: CreatePlacement 群組
	ec2：游泳CreatePublic池
	ec2：CreateReplaceRootVolume任務
	ec2：CreateReservedInstancesListing
	ec2：CreateRestoreImageTask
	ec2：CreateRoute
	ec2：CreateRoute表
	ec2: CreateSecurity 群組
	ec2：CreateSnapshot
	ec2：CreateSnapshots
	ec2：CreateSpotDatafeedSubscription

服務前綴	動作
	ec2：CreateStoreImageTask
	ec2：CreateSubnet
	ec2：CreateSubnetCidrReservation
	ec2：CreateTrafficMirrorFilter
	ec2：CreateTrafficMirrorFilter規則
	ec2：CreateTrafficMirrorSession
	ec2：CreateTrafficMirrorTarget
	ec2：CreateTransit網關
	ec2：CreateTransitGatewayConnect
	ec2：CreateTransitGatewayConnect對等
	ec2: CreateTransit GatewayMulticast 網域名稱
	ec2：CreateTransitGatewayPeering附件
	ec2：CreateTransitGatewayPolicy表
	ec2：CreateTransitGatewayPrefixListReference
	ec2：CreateTransitGatewayRoute
	ec2：CreateTransitGatewayRoute表
	ec2：CreateTransitGatewayRouteTableAnnouncement
	ec2：CreateTransitGatewayVpc附件
	ec2：CreateVerifiedAccessEndpoint
	ec2：CreateVerifiedAccessGroup
	ec2：CreateVerifiedAccessInstance

服務前綴	動作
	ec2：CreateVerifiedAccessTrust供應商
	ec2：CreateVolume
	ec2：CreateVpc
	ec2：CreateVpc端點
	ec2：CreateVpcEndpointConnection通知
	ec2：CreateVpcEndpointService配置
	ec2：CreateVpcPeeringConnection
	ec2：CreateVpn連接
	ec2：CreateVpnConnectionRoute
	ec2：CreateVpn網關
	ec2：DeleteCarrier網關
	ec2：DeleteClientVpnEndpoint
	ec2：DeleteClientVpnRoute
	ec2：DeleteCoip西德
	ec2：DeleteCoip游泳池
	ec2：DeleteCustomer網關
	ec2：DeleteDhcp選項
	ec2：DeleteEgressOnlyInternet網關
	ec2：DeleteFleets
	ec2：DeleteFlow日誌
	ec2：DeleteFpga圖片

服務前綴	動作
	ec2：DeleteInstanceConnectEndpoint
	ec2：DeleteInstanceEventWindow
	ec2：DeleteInternet網關
	ec2：DeleteIpam
	ec2：DeleteIpam游泳池
	ec2：DeleteIpamResourceDiscovery
	ec2：DeleteIpam範圍
	ec2：DeleteKey配對
	ec2：DeleteLaunch模板
	ec2：DeleteLaunchTemplateVersions
	ec2：DeleteLocalGatewayRoute
	ec2：DeleteLocalGatewayRoute表
	ec2：DeleteLocalGatewayRouteTableVirtualInterfaceGroup關聯
	ec2：DeleteLocalGatewayRouteTableVpc關聯
	ec2：DeleteManagedPrefixList
	ec2：DeleteNat網關
	ec2：DeleteNetwork十字韌帶
	ec2：DeleteNetworkAclEntry
	ec2：DeleteNetworkInsightsAccess範圍
	ec2：DeleteNetworkInsightsAccessScopeAnalysis
	ec2：DeleteNetworkInsightsAnalysis

服務前綴	動作
	ec2：DeleteNetworkInsightsPath
	ec2：DeleteNetwork接口
	ec2：DeleteNetworkInterfacePermission
	ec2: DeletePlacement 群組
	ec2：游泳DeletePublic池
	ec2：DeleteQueuedReservedInstances
	ec2：DeleteRoute
	ec2：DeleteRoute表
	ec2: DeleteSecurity 群組
	ec2：DeleteSnapshot
	ec2：DeleteSpotDatafeedSubscription
	ec2：DeleteSubnet
	ec2：DeleteSubnetCidrReservation
	ec2：DeleteTrafficMirrorFilter
	ec2：DeleteTrafficMirrorFilter規則
	ec2：DeleteTrafficMirrorSession
	ec2：DeleteTrafficMirrorTarget
	ec2：DeleteTransit網關
	ec2：DeleteTransitGatewayConnect
	ec2：DeleteTransitGatewayConnect對等
	ec2: DeleteTransit GatewayMulticast 網域名稱

服務前綴	動作
	ec2：DeleteTransitGatewayPeering附件
	ec2：DeleteTransitGatewayPolicy表
	ec2：DeleteTransitGatewayPrefixListReference
	ec2：DeleteTransitGatewayRoute
	ec2：DeleteTransitGatewayRoute表
	ec2：DeleteTransitGatewayRouteTableAnnouncement
	ec2：DeleteTransitGatewayVpc附件
	ec2：DeleteVerifiedAccessEndpoint
	ec2：DeleteVerifiedAccessGroup
	ec2：DeleteVerifiedAccessInstance
	ec2：DeleteVerifiedAccessTrust供應商
	ec2：DeleteVolume
	ec2：DeleteVpc
	ec2：DeleteVpcEndpointConnection通知
	ec2：DeleteVpc端點
	ec2：DeleteVpcEndpointService配置
	ec2：DeleteVpcPeeringConnection
	ec2：DeleteVpn連接
	ec2：DeleteVpnConnectionRoute
	ec2：DeleteVpn網關
	ec2：DeprovisionByoip西德

服務前綴	動作
	ec2：自助DeprovisionIpam餐
	ec2：DeprovisionIpamPoolCidr
	ec2：有DeprovisionPublic效 PoolCidr
	ec2：DeregisterImage
	ec2：DeregisterInstanceEventNotification屬性
	ec2：DeregisterTransitGatewayMulticastGroupMembers
	ec2：DeregisterTransitGatewayMulticastGroupSources
	ec2：DescribeAccount屬性
	ec2：DescribeAddresses
	ec2：DescribeAddresses屬性
	ec2：DescribeAddress轉移
	ec2：DescribeAggregateIdFormat
	ec2：DescribeAvailability區域
	ec2：DescribeAwsNetworkPerformanceMetricSubscriptions
	ec2：DescribeBundle任務
	ec2：DescribeByoip西德爾斯
	ec2：DescribeCapacityReservationFleets
	ec2：DescribeCapacity保留
	ec2：DescribeCarrier閘道器
	ec2：DescribeClassicLinkInstances
	ec2：DescribeClientVpnAuthorization規則

服務前綴	動作
	ec2：DescribeClientVpnConnections
	ec2：DescribeClientVpnEndpoints
	ec2：DescribeClientVpnRoutes
	ec2：DescribeClientVpnTarget網絡
	ec2：DescribeCoip池
	ec2：DescribeConversion任務
	ec2：DescribeCustomer閘道器
	ec2：DescribeDhcp選項
	ec2：DescribeEgressOnlyInternet閘道器
	ec2：DescribeElasticGPU
	ec2：DescribeExportImageTasks
	ec2：DescribeExport任務
	ec2：DescribeFastLaunchImages
	ec2：DescribeFastSnapshotRestores
	ec2：DescribeFleet歷史記錄
	ec2：DescribeFleet實例
	ec2：DescribeFleets
	ec2：DescribeFlow日誌
	ec2：DescribeFpgaImageAttribute
	ec2：DescribeFpga圖片
	ec2：DescribeHostReservationOfferings

服務前綴	動作
	ec2：DescribeHost保留
	ec2：DescribeHosts
	ec2：DescribeIamInstanceProfile關聯
	ec2：DescribeIdentityIdFormat
	ec2：DescribeId格式
	ec2：DescribeImage屬性
	ec2：DescribeImages
	ec2：DescribeImportImageTasks
	ec2：DescribeImportSnapshotTasks
	ec2：DescribeInstance屬性
	ec2：DescribeInstanceConnectEndpoints
	ec2：DescribeInstanceCreditSpecifications
	ec2：DescribeInstanceEventNotification屬性
	ec2：DescribeInstanceEventWindows
	ec2：DescribeInstances
	ec2：DescribeInstance狀態
	ec2: DescribeInstance 拓撲
	ec2：DescribeInstanceTypeOfferings
	ec2：DescribeInstance類型
	ec2：DescribeInternet閘道器
	ec2：自助DescribeIpam餐

服務前綴	動作
	ec2：DescribeIpam池
	ec2：DescribeIpamResourceDiscoveries
	ec2：DescribeIpamResourceDiscovery關聯
	ec2：DescribeIpams
	ec2：DescribeIpam範圍
	ec2：DescribeIpv6 個池
	ec2：DescribeKey對
	ec2：DescribeLaunch模板
	ec2：DescribeLaunchTemplateVersions
	ec2：DescribeLocalGatewayRoute表
	ec2：DescribeLocalGatewayRouteTableVirtualInterfaceGroup關聯
	ec2：DescribeLocalGatewayRouteTableVpc關聯
	ec2：DescribeLocal閘道器
	ec2：DescribeLocalGatewayVirtualInterfaceGroups
	ec2：DescribeLocalGatewayVirtual接口
	ec2：DescribeLocked快照
	ec2：DescribeMac主機
	ec2：DescribeManagedPrefixLists
	ec2：DescribeMoving地址
	ec2：DescribeNat閘道器
	ec2：DescribeNetworkACL

服務前綴	動作
	ec2：DescribeNetworkInsightsAccessScopeAnalyses
	ec2：DescribeNetworkInsightsAccess範圍
	ec2：DescribeNetworkInsightsAnalyses
	ec2：DescribeNetworkInsightsPaths
	ec2：DescribeNetworkInterfaceAttribute
	ec2：DescribeNetworkInterfacePermissions
	ec2：DescribeNetwork接口
	ec2：DescribePlacement群組
	ec2：DescribePrefix列表
	ec2：DescribePrincipalIdFormat
	ec2：有限DescribePublic池
	ec2：DescribeRegions
	ec2：DescribeReplaceRootVolume任務
	ec2：DescribeReserved實例
	ec2：DescribeReservedInstancesListings
	ec2：DescribeReservedInstancesModifications
	ec2：DescribeReservedInstancesOfferings
	ec2：DescribeRoute表
	ec2：DescribeScheduledInstanceAvailability
	ec2：DescribeScheduled實例
	ec2：DescribeSecurityGroupReferences

服務前綴	動作
	ec2：DescribeSecurityGroupRules
	ec2：DescribeSecurity群組
	ec2：DescribeSnapshot屬性
	ec2：DescribeSnapshots
	ec2：DescribeSnapshotTierStatus
	ec2：DescribeSpotDatafeedSubscription
	ec2：DescribeSpotFleetInstances
	ec2：DescribeSpotFleetRequest歷史記錄
	ec2：DescribeSpotFleetRequests
	ec2：DescribeSpotInstanceRequests
	ec2：DescribeSpotPriceHistory
	ec2：DescribeStaleSecurityGroups
	ec2：DescribeStoreImageTasks
	ec2：DescribeSubnets
	ec2：DescribeTrafficMirrorFilters
	ec2：DescribeTrafficMirrorSessions
	ec2：DescribeTrafficMirrorTargets
	ec2：DescribeTransitGatewayAttachments
	ec2：DescribeTransitGatewayConnect同行
	ec2：DescribeTransitGatewayConnects
	ec2: DescribeTransit GatewayMulticast 網域名稱

服務前綴	動作
	ec2：DescribeTransitGatewayPeering附件
	ec2：DescribeTransitGatewayPolicy表
	ec2：DescribeTransitGatewayRouteTableAnnouncements
	ec2：DescribeTransitGatewayRoute表
	ec2：DescribeTransit閘道器
	ec2：DescribeTransitGatewayVpc附件
	ec2：DescribeTrunkInterfaceAssociations
	ec2：DescribeVerifiedAccessEndpoints
	ec2：DescribeVerifiedAccessGroups
	ec2：DescribeVerifiedAccessInstanceLoggingConfigurations
	ec2：DescribeVerifiedAccessInstances
	ec2：DescribeVerifiedAccessTrust供應商
	ec2：DescribeVolume屬性
	ec2：DescribeVolumes
	ec2：DescribeVolumes修改
	ec2：DescribeVolume狀態
	ec2：DescribeVpc屬性
	ec2：DescribeVpcClassicLink
	ec2：DescribeVpcClassicLinkDnsSupport
	ec2：DescribeVpcEndpointConnection通知
	ec2：DescribeVpcEndpointConnections

服務前綴	動作
	ec2：DescribeVpc端點
	ec2：DescribeVpcEndpointService配置
	ec2：DescribeVpcEndpointService權限
	ec2：DescribeVpcEndpointServices
	ec2：DescribeVpcPeeringConnections
	ec2：DescribeVpcs
	ec2：DescribeVpn連接
	ec2：DescribeVpn閘道器
	ec2：DetachClassicLinkVpc
	ec2：DetachInternet網關
	ec2：DetachNetwork接口
	ec2：DetachVerifiedAccessTrust供應商
	ec2：DetachVolume
	ec2：DetachVpn網關
	ec2：DisableAddress轉移
	ec2：DisableAwsNetworkPerformanceMetricSubscription
	ec2：DisableEbsEncryptionBy默認
	ec2：DisableFast啟動
	ec2：DisableFastSnapshotRestores
	ec2：DisableImage
	ec2：DisableImageBlockPublic訪問

服務前綴	動作
	ec2：DisableImage棄用
	ec2：DisableImageDeregistrationProtection
	ec2：DisableIpamOrganizationAdmin帳戶
	ec2：DisableSerialConsoleAccess
	ec2：DisableSnapshotBlockPublic訪問
	ec2：DisableTransitGatewayRouteTablePropagation
	ec2：DisableVgwRoutePropagation
	ec2：DisableVpcClassicLink
	ec2：DisableVpcClassicLinkDnsSupport
	ec2：DisassociateAddress
	ec2：DisassociateClientVpnTarget網絡
	ec2：DisassociateEnclaveCertificateIam角色
	ec2：DisassociateIamInstanceProfile
	ec2：DisassociateInstanceEventWindow
	ec2：自助DisassociateIpam餐
	ec2：DisassociateIpamResourceDiscovery
	ec2：DisassociateNatGatewayAddress
	ec2：DisassociateRoute表
	ec2：DisassociateSubnetCidrBlock
	ec2: DisassociateTransit GatewayMulticast 網域名稱
	ec2：DisassociateTransitGatewayPolicy表

服務前綴	動作
	ec2：DisassociateTransitGatewayRoute表
	ec2：DisassociateTrunk接口
	ec2：DisassociateVpcCidrBlock
	ec2：EnableAddress轉移
	ec2：EnableAwsNetworkPerformanceMetricSubscription
	ec2：EnableEbsEncryptionBy默認
	ec2：EnableFast啟動
	ec2：EnableFastSnapshotRestores
	ec2：EnableImage
	ec2：EnableImageBlockPublic訪問
	ec2：EnableImage棄用
	ec2：EnableImageDeregistrationProtection
	ec2：EnableIpamOrganizationAdmin帳戶
	ec2：EnableReachabilityAnalyzerOrganization共享
	ec2：EnableSerialConsoleAccess
	ec2：EnableSnapshotBlockPublic訪問
	ec2：EnableTransitGatewayRouteTablePropagation
	ec2：EnableVgwRoutePropagation
	EC2：EnableVolumeIO
	ec2：EnableVpcClassicLink
	ec2：EnableVpcClassicLinkDnsSupport

服務前綴	動作
	ec2：ExportClientVpnClientCertificateRevocation列表
	ec2：ExportClientVpnClient配置
	ec2：ExportImage
	ec2：ExportTransitGatewayRoutes
	ec2：GetAssociatedEnclaveCertificateIamRoles
	ec2：GetAssociatedIPV6 PoolCidrs
	ec2：GetAwsNetworkPerformance數據
	ec2：GetCapacityReservationUsage
	ec2：GetCoipPoolUsage
	ec2：GetConsole輸出
	ec2：GetConsole屏幕截圖
	ec2：GetDefaultCreditSpecification
	ec2：GetEbsDefaultKmsKeyId
	ec2：GetEbsEncryptionBy默認
	ec2：GetFlowLogsIntegration模板
	ec2: GetGroups ForCapacity 預約
	ec2：GetHostReservationPurchase預覽
	ec2：GetImageBlockPublicAccessState
	ec2：GetInstanceMetadataDefaults
	ec2: GetInstance TpmEk 酒吧
	ec2：GetInstanceTypesFromInstanceRequirements

服務前綴	動作
	ec2：GetInstanceUefiData
	ec2：GetIpamAddressHistory
	ec2：GetIpamDiscoveredAccounts
	ec2：GetIpamDiscoveredPublic地址
	ec2：GetIpamDiscoveredResource西德爾斯
	ec2：GetIpamPoolAllocations
	ec2：GetIpamPoolCidrs
	ec2：GetIpamResourceCidrs
	ec2：GetLaunchTemplateData
	ec2：GetManagedPrefixList關聯
	ec2：GetManagedPrefixList條目
	ec2：GetNetworkInsightsAccessScopeAnalysis發現結果
	ec2：GetNetworkInsightsAccessScopeContent
	ec2：GetPassword數據
	ec2：GetReservedInstancesExchange報價
	ec2：GetSecurityGroupsForVPC
	ec2：GetSerialConsoleAccess狀態
	ec2：GetSnapshotBlockPublicAccessState
	ec2：GetSpotPlacementScores
	ec2：GetSubnetCidrReservations
	ec2：GetTransitGatewayAttachment傳播

服務前綴	動作
	ec2：GetTransitGatewayMulticastDomainAssociations
	ec2：GetTransitGatewayPolicyTableAssociations
	ec2：GetTransitGatewayPolicyTableEntries
	ec2：GetTransitGatewayPrefixListReferences
	ec2：GetTransitGatewayRouteTableAssociations
	ec2：GetTransitGatewayRouteTablePropagations
	ec2：GetVerifiedAccessEndpoint政策
	ec2：GetVerifiedAccessGroup政策
	ec2：GetVpnConnectionDeviceSampleConfiguration
	ec2：GetVpnConnectionDevice類型
	ec2：GetVpnTunnelReplacement狀態
	ec2：ImportClientVpnClientCertificateRevocation列表
	ec2：ImportImage
	ec2：ImportInstance
	ec2：ImportKey配對
	ec2：ImportSnapshot
	ec2：ImportVolume
	ec2：ListImagesInRecycle垃圾桶
	ec2：ListSnapshotsInRecycle垃圾桶
	ec2：LockSnapshot
	ec2：ModifyAddress屬性

服務前綴	動作
	ec2：ModifyAvailabilityZoneGroup
	ec2: ModifyCapacity 預約
	ec2：ModifyCapacityReservationFleet
	ec2：ModifyClientVpnEndpoint
	ec2：ModifyDefaultCreditSpecification
	ec2：ModifyEbsDefaultKmsKeyId
	ec2：ModifyFleet
	ec2：ModifyFpgaImageAttribute
	ec2：ModifyHosts
	ec2：ModifyIdentityIdFormat
	ec2：ModifyId格式
	ec2：ModifyImage屬性
	ec2：ModifyInstance屬性
	ec2：ModifyInstanceCapacityReservation屬性
	ec2：ModifyInstanceCreditSpecification
	ec2：ModifyInstanceEventStart時間
	ec2：ModifyInstanceEventWindow
	ec2：ModifyInstanceMaintenanceOptions
	ec2：ModifyInstanceMetadataDefaults
	ec2：ModifyInstanceMetadataOptions
	ec2: ModifyInstance 放置

服務前綴	動作
	ec2：ModifyIpam
	ec2：ModifyIpam游泳池
	ec2：ModifyIpamResourceCidr
	ec2：ModifyIpamResourceDiscovery
	ec2：ModifyIpam範圍
	ec2：ModifyLaunch模板
	ec2：ModifyLocalGatewayRoute
	ec2：ModifyManagedPrefixList
	ec2：ModifyNetworkInterfaceAttribute
	ec2：ModifyPrivateDnsName選項
	ec2：ModifyReserved實例
	ec2：ModifySecurityGroupRules
	ec2：ModifySnapshot屬性
	ec2：ModifySnapshot層
	ec2：ModifySpotFleetRequest
	ec2：ModifySubnet屬性
	ec2：ModifyTrafficMirrorFilterNetworkServices
	ec2：ModifyTrafficMirrorFilter規則
	ec2：ModifyTrafficMirrorSession
	ec2：ModifyTransit網關
	ec2：ModifyTransitGatewayPrefixListReference

服務前綴	動作
	ec2：ModifyTransitGatewayVpc附件
	ec2：ModifyVerifiedAccessEndpoint
	ec2：ModifyVerifiedAccessEndpoint政策
	ec2：ModifyVerifiedAccessGroup
	ec2：ModifyVerifiedAccessGroup政策
	ec2：ModifyVerifiedAccessInstance
	ec2：ModifyVerifiedAccessInstanceLoggingConfiguration
	ec2：ModifyVerifiedAccessTrust供應商
	ec2：ModifyVolume
	ec2：ModifyVolume屬性
	ec2：ModifyVpc屬性
	ec2：ModifyVpc端點
	ec2：ModifyVpcEndpointConnection通知
	ec2：ModifyVpcEndpointService配置
	ec2：ModifyVpcEndpointServicePayerResponsibility
	ec2：ModifyVpcEndpointService權限
	ec2：ModifyVpcPeeringConnection選項
	ec2: ModifyVpc 租賃
	ec2：ModifyVpn連接
	ec2：ModifyVpnConnectionOptions
	ec2：ModifyVpnTunnelCertificate

服務前綴	動作
	ec2：ModifyVpnTunnelOptions
	ec2：MonitorInstances
	ec2：MoveAddressToVpc
	ec2：MoveByoipCidrTo伊帕姆
	ec2：ProvisionByoip西德
	ec2：自助ProvisionIpam餐
	ec2：ProvisionIpamPoolCidr
	ec2：有ProvisionPublic效 PoolCidr
	ec2: PurchaseHost 預約
	ec2：PurchaseReservedInstancesOffering
	ec2：PurchaseScheduled實例
	ec2：RebootInstances
	ec2：RegisterImage
	ec2：RegisterInstanceEventNotification屬性
	ec2：RegisterTransitGatewayMulticastGroupMembers
	ec2：RegisterTransitGatewayMulticastGroupSources
	ec2：RejectTransitGatewayMulticastDomainAssociations
	ec2：RejectTransitGatewayPeering附件
	ec2：RejectTransitGatewayVpc附件
	ec2：RejectVpcEndpointConnections
	ec2：RejectVpcPeeringConnection

服務前綴	動作
	ec2：ReleaseAddress
	ec2：ReleaseHosts
	ec2：ReleaseIpamPoolAllocation
	ec2：ReplaceIamInstanceProfile關聯
	ec2：ReplaceNetworkAclAssociation
	ec2：ReplaceNetworkAclEntry
	ec2：ReplaceRoute
	ec2：ReplaceRouteTableAssociation
	ec2：ReplaceTransitGatewayRoute
	ec2：ReplaceVpn隧道
	ec2：ReportInstance狀態
	ec2：RequestSpot艦隊
	ec2：RequestSpot實例
	ec2：ResetAddress屬性
	ec2：ResetEbsDefaultKmsKeyId
	ec2：ResetFpgaImageAttribute
	ec2：ResetImage屬性
	ec2：ResetInstance屬性
	ec2：ResetNetworkInterfaceAttribute
	ec2：ResetSnapshot屬性
	ec2：RestoreAddressToClassic

服務前綴	動作
	ec2：RestoreImageFromRecycle垃圾桶
	ec2：RestoreManagedPrefixList版本
	ec2：RestoreSnapshotFromRecycle垃圾桶
	ec2：RestoreSnapshot層
	ec2：RevokeClientVpnIngress
	ec2：RevokeSecurityGroupEgress
	ec2：RevokeSecurityGroupIngress
	ec2：RunInstances
	ec2：RunScheduled實例
	ec2：SearchLocalGatewayRoutes
	ec2：SearchTransitGatewayMulticast群組
	ec2：SearchTransitGatewayRoutes
	ec2：SendDiagnostic中斷
	ec2：StartInstances
	ec2：StartNetworkInsightsAccessScopeAnalysis
	ec2：StartNetworkInsightsAnalysis
	ec2：StartVpcEndpointServicePrivateDns驗證
	ec2：StopInstances
	ec2：TerminateClientVpnConnections
	ec2：TerminateInstances
	ec2：UnassignIpv6 地址

服務前綴	動作
	ec2：UnassignPrivateIpAddresses
	ec2：UnassignPrivateNatGateway地址
	ec2：UnlockSnapshot
	ec2：UnmonitorInstances
	ec2：UpdateSecurityGroupRuleDescriptionsEgress
	ec2：UpdateSecurityGroupRuleDescriptionsIngress
	ec2：WithdrawByoip西德

服務前綴	動作
ecr	ECR: BatchCheck LayerAvailability ecr: BatchDelete 圖片 ecr: BatchGet 圖片 ecr：BatchGetRepositoryScanning配置 ecr: CompleteLayer 上傳 ecr：CreatePullThroughCache規則 ECR: CreateRepository ECR: CreateRepository CreationTemplate ECR: 政策 DeleteLifecycle ecr：DeletePullThroughCache規則 ECR: 政策 DeleteRegistry ECR: DeleteRepository ECR: DeleteRepository CreationTemplate ECR: 政策 DeleteRepository ECR: DescribeImage ReplicationStatus ECR: DescribeImages ECR: DescribeImage ScanFindings ECR：DescribePullThroughCache規則 ECR: DescribeRegistry ECR: DescribeRepositories ecr：令牌 GetAuthorization

服務前綴	動作
	ECR: GetDownload UrlFor 圖層 ECR: 政策 GetLifecycle ECR: GetLifecycle PolicyPreview ECR: 政策 GetRegistry ECR: GetRegistry ScanningConfiguration ECR: 政策 GetRepository ecr: InitiateLayer 上傳 ECR: ListImages ECR: PutImage ECR: PutImage ScanningConfiguration ECR: 政策 PutRegistry ECR: PutRegistry ScanningConfiguration ecr：PutReplication配置 ecr: 掃描 StartImage ECR: StartLifecycle PolicyPreview ecr：UpdatePullThroughCache規則 ecr: UploadLayer 零件 ecr：ValidatePullThroughCache規則

服務前綴	動作
ecr-public	ECR-公眾號碼:BatchCheckLayerAvailability
	ecr-公共：圖片 BatchDelete
	ecr-公共：上傳 CompleteLayer
	ECR-公眾號碼:CreateRepository
	ECR-公眾號碼:DeleteRepository
	ECR-公眾：政策 DeleteRepository
	ECR-公眾號碼:DescribeImages
	ECR-公眾號碼:DescribeRegistries
	ECR-公共：DescribeRepositories
	ecr-公共：令牌 GetAuthorization
	ECR-公眾號碼:GetRegistryCatalogData
	ECR-公眾號碼:GetRepositoryCatalogData
	ECR-公眾：政策 GetRepository
	ecr-公共：上傳 InitiateLayer
	ECR-公共：PutImage
	ECR-公眾號碼:PutRegistryCatalogData
	ECR-公眾號碼:PutRepositoryCatalogData
	ECR-公眾：政策 SetRepository
	ecr-公眾：部分 UploadLayer

服務前綴	動作
ecs	ecs: CreateCapacity 供應商 ECS: CreateCluster ECS: CreateService ECS: CreateTask 設置 ECS: DeleteAccount 設定 ECS: DeleteAttributes ecs: DeleteCapacity 供應商 ECS: DeleteCluster ECS: DeleteService ECS: DeleteTask 定義 ECS: DeleteTask 設置 ECS: DeregisterContainer 實例 ECS: DeregisterTask 定義 ecs: DescribeCapacity 供應商 ECS: DescribeClusters ECS: DescribeContainer 執行個體 ECS: DescribeServices ECS: DescribeTask 定義 ECS: DescribeTasks ECS: DescribeTask 套裝 ECS: 端點 DiscoverPoll

服務前綴	動作
	ECS: ExecuteCommand ECS: GetTask 保護 ECS: ListAccount 設置 ECS: ListAttributes ECS: ListClusters ECS: ListContainer 執行個體 ECS: ListServices ECS: ListServices ByNamespace ECS: ListTask DefinitionFamilies ECS: ListTask 定義 ECS: ListTasks ECS: PutAccount 設定 ECS: PutAccount SettingDefault ECS: PutAttributes ECS: PutCluster CapacityProviders ECS: RegisterContainer 實例 ECS: RegisterTask 定義 ECS: RunTask ECS: StartTask ECS: StopTask ECS: SubmitAttachment StateChanges

服務前綴	動作
	ECS: SubmitContainer StateChange ECS: SubmitTask StateChange ecs: UpdateCapacity 供應商 ECS: UpdateCluster ECS: UpdateCluster 設置 ECS: UpdateContainer 代理 ECS: UpdateContainer InstancesState ECS: UpdateService ECS: UpdateService PrimaryTask 設置 ECS: UpdateTask 保護 ECS: UpdateTask 設置

服務前綴	動作
eks	埃克斯：政策 AssociateAccess 埃克斯：Config AssociateEncryption 是:AssociateIdentityProviderConfig 埃克斯：CreateAccess條目 是:CreateAddon 是:CreateCluster 是:CreateEksAnywhereSubscription EKS：CreateFargate配置文件 是:CreateNodegroup 埃克斯：DeleteAccess條目 是:DeleteAddon 是:DeleteCluster 是:DeleteEksAnywhereSubscription EKS：DeleteFargate配置文件 是:DeleteNodegroup 是:DeletePodIdentityAssociation 是:DeregisterCluster 埃克斯：DescribeAccess條目 是:DescribeAddon 例如：DescribeAddon配置 類別：DescribeAddon版本

服務前綴	動作
	是:DescribeCluster 是:DescribeEksAnywhereSubscription EKS：DescribeFargate配置文件 是:DescribeIdentityProviderConfig 是:DescribeInsight 是:DescribeNodegroup 是:DescribePodIdentityAssociation 是:DescribeUpdate 埃克斯：政策 DisassociateAccess 是:DisassociateIdentityProviderConfig 埃克斯：ListAccess條目 作為:政策 ListAccess 是:ListAddons 是:ListAssociatedAccessPolicies 是:ListClusters 是:ListEksAnywhereSubscriptions 一個：ListFargate配置文件 是:ListIdentityProviderConfigs 是:ListInsights 是:ListNodegroups 是:ListPodIdentityAssociations

服務前綴	動作
	是:ListUpdates 是:RegisterCluster 埃克斯：UpdateAccess條目 是:UpdateAddon 埃克斯：Config UpdateCluster 類別：UpdateCluster版本 是:UpdateEksAnywhereSubscription 埃克斯：Config UpdateNodegroup 類別：UpdateNodegroup版本 是:UpdatePodIdentityAssociation
elastic-inference	彈性推論：供應項目 DescribeAccelerator 彈性推斷：DescribeAccelerators 彈性推斷：類型 DescribeAccelerator

服務前綴	動作
elasticache	彈性痛:入口 AuthorizeCache SecurityGroup
	彈性痛：BatchApplyUpdateAction
	彈性痛：BatchStopUpdateAction
	彈性痛：CompleteMigration
	彈性痛：CopyServerlessCacheSnapshot
	彈性痛：CopySnapshot
	彈性痛：集群 CreateCache
	彈性痛：CreateCacheParameterGroup
	彈性痛：CreateCacheSecurityGroup
	彈性痛：CreateCacheSubnetGroup
	彈性痛：CreateGlobalReplicationGroup
	彈性痛:組 CreateReplication
	彈性疼痛：緩存 CreateServerless
	彈性痛：CreateServerlessCacheSnapshot
	彈性痛：CreateSnapshot
	彈性痛：CreateUser
	彈性痛:組 CreateUser
	彈性痛:組 DecreaseNode GroupsIn GlobalReplication
	彈性痛：計數 DecreaseReplica
	彈性痛：集群 DeleteCache
	彈性痛：DeleteCacheParameterGroup

服務前綴	動作
	彈性痛：DeleteCacheSecurityGroup
	彈性痛：DeleteCacheSubnetGroup
	彈性痛：DeleteGlobalReplicationGroup
	彈性痛:組 DeleteReplication
	彈性疼痛：緩存 DeleteServerless
	彈性痛：DeleteServerlessCacheSnapshot
	彈性痛：DeleteSnapshot
	彈性痛：DeleteUser
	彈性痛:組 DeleteUser
	彈性痛：簇 DescribeCache
	彈性痛：DescribeCacheEngineVersions
	彈性痛：DescribeCacheParameterGroups
	彈性痛：參數 DescribeCache
	彈性痛：DescribeCacheSecurityGroups
	彈性痛：DescribeCacheSubnetGroups
	彈性痛：DescribeEngineDefaultParameters
	彈性痛：DescribeEvents
	彈性痛：DescribeGlobalReplicationGroups
	彈性痛:群體 DescribeReplication
	彈性痛：DescribeReservedCacheNodes
	彈性痛:產品 DescribeReserved CacheNodes

服務前綴	動作
	彈性痛：緩存 DescribeServerless
	彈性痛：DescribeServerlessCacheSnapshots
	彈性痛：更新 DescribeService
	彈性痛：DescribeSnapshots
	彈性痛：動作 DescribeUpdate
	彈性痛:群體 DescribeUser
	彈性痛：DescribeUsers
	彈性痛：DisassociateGlobalReplicationGroup
	彈性痛：ExportServerlessCacheSnapshot
	彈性痛：FailoverGlobalReplicationGroup
	彈性痛:組 IncreaseNode GroupsIn GlobalReplication
	彈性痛：計數 IncreaseReplica
	彈性痛：修改 ListAllowed NodeType
	彈性痛：集群 ModifyCache
	彈性痛：ModifyCacheParameterGroup
	彈性痛：ModifyCacheSubnetGroup
	彈性痛：ModifyGlobalReplicationGroup
	彈性痛:組 ModifyReplication
	彈性疼痛：配置 ModifyReplication GroupShard
	彈性疼痛：緩存 ModifyServerless
	彈性痛：ModifyUser

服務前綴	動作
	彈性痛:組 ModifyUser
	彈性痛:提供 PurchaseReserved CacheNodes
	彈性痛：RebalanceSlotsInGlobalReplicationGroup
	彈性痛：集群 RebootCache
	彈性痛：ResetCacheParameterGroup
	彈性痛:入口 RevokeCache SecurityGroup
	彈性痛：StartMigration
	彈性痛：TestFailover
	彈性痛：TestMigration

服務前綴	動作
elasticbeanstalk	彈性鏈接：更新 AbortEnvironment 彈性豆莖：ApplyEnvironmentManagedAction 彈性豆莖：AssociateEnvironmentOperationsRole elasticbeanstalk:CheckDNSAvailability 彈性豆莖：ComposeEnvironments 彈性豆莖：CreateApplication 彈性鏈條：版本 CreateApplication 彈性鏈條：模板 CreateConfiguration 彈性豆莖：CreateEnvironment 彈性鏈條：版本 CreatePlatform 彈性桿：位置 CreateStorage 彈性豆莖：DeleteApplication 彈性鏈條：版本 DeleteApplication 彈性鏈條：模板 DeleteConfiguration 彈性鏈條：配置 DeleteEnvironment 彈性鏈條：版本 DeletePlatform 彈性狀態：屬性 DescribeAccount 彈性豆莖：DescribeApplications 彈性鏈條：版本 DescribeApplication 彈性鏈條：選項 DescribeConfiguration 彈性鏈接：設置 DescribeConfiguration

服務前綴	動作
	彈性豆莖：Health DescribeEnvironment 彈性鏈條：歷史 DescribeEnvironment ManagedAction 彈性豆莖：DescribeEnvironmentManagedActions 彈性資源：資源 DescribeEnvironment 彈性豆莖：DescribeEnvironments 彈性豆莖：DescribeEvents 彈性豆莖：Health DescribeInstances 彈性鏈條：版本 DescribePlatform 彈性豆莖：DisassociateEnvironmentOperationsRole 彈性豆莖：ListAvailableSolutionStacks 彈性鏈條：分支機構 ListPlatform 彈性鏈條：版本 ListPlatform 彈性豆莖：RebuildEnvironment 彈性跟踪：信息 RequestEnvironment 彈性：服務器 RestartApp 彈性跟踪：信息 RetrieveEnvironment 彈性鏈條：CNAME SwapEnvironment 彈性豆莖：TerminateEnvironment 彈性豆莖：UpdateApplication 彈性豆莖：UpdateApplicationResourceLifecycle 彈性鏈條：版本 UpdateApplication

服務前綴	動作
	彈性鏈條：模板 UpdateConfiguration 彈性豆莖：UpdateEnvironment 彈性鏈接：設置 ValidateConfiguration

服務前綴	動作
elasticfilesystem	彈性文件系統：點 CreateAccess
	彈性文件系統：系統 CreateFile
	彈性文件系統：目標 CreateMount
	彈性文件系統：配置 CreateReplication
	彈性文件系統：點 DeleteAccess
	彈性文件系統：系統 DeleteFile
	彈性文件系統：DeleteFileSystemPolicy
	彈性文件系統：目標 DeleteMount
	彈性文件系統：配置 DeleteReplication
	彈性文件系統：點 DescribeAccess
	彈性檔案系統：偏好設定 DescribeAccount
	彈性文件系統：策略 DescribeBackup
	彈性文件系統：DescribeFileSystemPolicy
	彈性文件系統：系統 DescribeFile
	彈性文件系統：配置 DescribeLifecycle
	彈性文件系統：目標 DescribeMount
	彈性檔案系統:群組 DescribeMount TargetSecurity
	彈性文件系統：配置 DescribeReplication
	彈性檔案系統:群組 ModifyMount TargetSecurity
	彈性檔案系統：偏好設定 PutAccount
	彈性文件系統：策略 PutBackup

服務前綴	動作
	彈性文件系統：PutFileSystemPolicy
	彈性文件系統：配置 PutLifecycle
	彈性文件系統：系統 UpdateFile
	彈性文件系統：UpdateFileSystemProtection

服務前綴	動作
elasticloadbalancing	彈性負載平衡：憑證 AddListener 彈性負載平衡：AddTrustStoreRevocations 彈性負載平衡：ApplySecurityGroupsToLoadBalancer 彈性負載平衡：子網路 AttachLoad BalancerTo 彈性負載平衡：檢查 ConfigureHealth 彈性負載平衡：政策 CreateApp CookieStickiness 彈性負載CookieStickiness平衡：建立原則 彈性負載平衡：CreateListener 彈性負載平衡：平衡器 CreateLoad 彈性負載平衡：CreateLoadBalancerListeners 彈性負載平衡：CreateLoadBalancerPolicy 彈性負載平衡：CreateRule 彈性負載平衡:群組 CreateTarget 彈性負載平衡：存儲 CreateTrust 彈性負載平衡：DeleteListener 彈性負載平衡：平衡器 DeleteLoad 彈性負載平衡：DeleteLoadBalancerListeners 彈性負載平衡：DeleteLoadBalancerPolicy 彈性負載平衡：DeleteRule 彈性負載平衡:群組 DeleteTarget 彈性負載平衡：存儲 DeleteTrust

服務前綴	動作
	彈性負載平衡：平衡器 DeregisterInstances FromLoad
	彈性負載平衡：DeregisterTargets
	彈性負載平衡：限制 DescribeAccount
	彈性負載平衡：Health DescribeInstance
	彈性負載平衡：憑證 DescribeListener
	彈性負載平衡：DescribeListeners
	彈性負載平衡：DescribeLoadBalancerAttributes
	彈性負載平衡：DescribeLoadBalancerPolicies
	彈性負載平衡：類型 DescribeLoad BalancerPolicy
	彈性負載平衡：平衡器 DescribeLoad
	彈性負載平衡：DescribeRules
	elasticloadbalancing:DescribeSSLPolicies
	彈性負載平衡：DescribeTargetGroupAttributes
	彈性負載平衡：群組 DescribeTarget
	彈性負載平衡：Health DescribeTarget
	彈性負載平衡：DescribeTrustStoreAssociations
	彈性負載平衡：DescribeTrustStoreRevocations
	彈性負載平衡：商店 DescribeTrust
	彈性負載平衡：子網路 DetachLoad BalancerFrom
	彈性負載平衡：DisableAvailabilityZonesForLoadBalancer
	彈性負載平衡：EnableAvailabilityZonesForLoadBalancer

服務前綴	動作
	彈性負載平衡：GetTrustStoreCaCertificatesBundle
	彈性負載平衡：內容 GetTrust StoreRevocation
	彈性負載平衡：ModifyListener
	彈性負載平衡：ModifyLoadBalancerAttributes
	彈性負載平衡：ModifyRule
	彈性負載平衡:群組 ModifyTarget
	彈性負載平衡：ModifyTargetGroupAttributes
	彈性負載平衡：存儲 ModifyTrust
	彈性負載平衡：平衡器 RegisterInstances WithLoad
	彈性負載平衡：RegisterTargets
	彈性負載平衡：憑證 RemoveListener
	彈性負載平衡：RemoveTrustStoreRevocations
	彈性負載平衡：SetIpAddressType
	彈性負載平衡：SSL 證書 SetLoad BalancerListener
	彈性負載平衡：伺服器 SetLoad BalancerPolicies ForBackend
	彈性負載平衡：SetLoadBalancerPoliciesOfListener
	彈性負載平衡：優先事項 SetRule
	彈性負載平衡：群組 SetSecurity
	彈性負載平衡：SetSubnets

服務前綴	動作
elastictranscoder	彈性轉碼器：CancelJob
	彈性轉碼器：CreateJob
	彈性轉碼器：CreatePipeline
	彈性轉碼器：CreatePreset
	彈性轉碼器：DeletePipeline
	彈性轉碼器：DeletePreset
	彈性轉碼器：ListJobsByPipeline
	彈性轉碼器：ListJobsByStatus
	彈性轉碼器：ListPipelines
	彈性轉碼器：ListPresets
	彈性轉碼器：ReadJob
	彈性轉碼器：ReadPipeline
	彈性轉碼器：ReadPreset
	彈性轉碼器：TestRole
	彈性轉碼器：UpdatePipeline
	彈性轉碼器：通知 UpdatePipeline
	彈性轉碼器：狀態 UpdatePipeline

服務前綴	動作
emr-containers	emr 容器：執行 CancelJob EMR 容器：模板 CreateJob emr 容器：端點 CreateManaged EMR 容器：配置 CreateSecurity emr 容器：叢集 CreateVirtual EMR 容器：模板 DeleteJob emr 容器：端點 DeleteManaged emr 容器：叢集 DeleteVirtual emr 容器：執行 DescribeJob EMR 容器：模板 DescribeJob emr 容器：端點 DescribeManaged EMR 容器：配置 DescribeSecurity emr 容器：叢集 DescribeVirtual emr 容器：認證 GetManaged EndpointSession emr 容器：執行 ListJob emr 容器：模板 ListJob emr 容器：端點 ListManaged emr 容器：配置 ListSecurity emr 容器：叢集 ListVirtual emr 容器：執行 StartJob

服務前綴	動作
emr-serverless	無伺服器：執行 CancelJob
	無伺服器：CreateApplication
	無伺服器：DeleteApplication
	無伺服器：GetApplication
	無伺服器：執行 GetDashboard ForJob
	無伺服器：執行 GetJob
	無伺服器：ListApplications
	無伺服器：執行 ListJob
	無伺服器：StartApplication
	無伺服器：執行 StartJob
	無伺服器：StopApplication
	無伺服器：UpdateApplication

服務前綴	動作
es	es: AcceptInbound 連接 是:AcceptInboundCrossClusterSearchConnection 是:AssociatePackage 是:AuthorizeVpcEndpointAccess es: CancelElasticsearch ServiceSoftware 更新 是:CancelServiceSoftwareUpdate 是:CreateDomain es: CreateElasticsearch 網域名稱 es: CreateOutbound 連接 是:CreateOutboundCrossClusterSearchConnection 是:CreatePackage es: CreateVpc 端點 是:DeleteDomain es: DeleteElasticsearch 網域名稱 是:DeleteElasticsearchServiceRole es: DeleteInbound 連接 是:DeleteInboundCrossClusterSearchConnection es: DeleteOutbound 連接 是:DeleteOutboundCrossClusterSearchConnection 是:DeletePackage es: DeleteVpc 端點

服務前綴	動作
	是:DescribeDomain 是:DescribeDomainAutoTunes 是:DescribeDomainChangeProgress ES: DescribeDomain Config 是:DescribeDomainHealth es: DescribeDomain 節點 是:DescribeDomains 是:DescribeDryRunProgress es: DescribeElasticsearch 網域名稱 是:DescribeElasticsearchDomainConfig es: DescribeElasticsearch 域名 es: DescribeElasticsearch InstanceType 限制 es: DescribeInbound 連接 是:DescribeInboundCrossClusterSearchConnections 是:DescribeInstanceTypeLimits es: DescribeOutbound 連接 是:DescribeOutboundCrossClusterSearchConnections 是:DescribePackages es: DescribeReserved ElasticsearchInstance 提供項目 是:DescribeReservedElasticsearchInstances 是:DescribeReservedInstanceOfferings

服務前綴	動作
	es: DescribeReserved 執行個體 es: DescribeVpc 端點 是:DissociatePackage 是:GetCompatibleElasticsearchVersions es: GetCompatible 版本 es: GetData 資料來源 是:GetDomainMaintenanceStatus 是:GetPackageVersionHistory es: GetUpgrade 歷史 es: GetUpgrade 狀態 es: ListData 資料來源 es: ListDomain 姓名 是:ListDomainsForPackage 是:ListElasticsearchInstanceTypes es: ListElasticsearch 版本 是:ListInstanceTypeDetails 是:ListPackagesForDomain 是:ListScheduled動作 是:ListVersions 是:ListVpcEndpointAccess es: ListVpc 端點

服務前綴	動作
	es: ListVpc EndpointsFor 網域名稱 es: PurchaseReserved ElasticsearchInstance 提供項目 是:PurchaseReservedInstanceOffering es: RejectInbound 連接 是:RejectInboundCrossClusterSearchConnection 是:RevokeVpcEndpointAccess es: StartDomain 維護 es: StartElasticsearch ServiceSoftware 更新 是:StartServiceSoftwareUpdate es: UpdateData 資料來源 ES: UpdateDomain Config 是:UpdateElasticsearchDomainConfig 是:UpdatePackage es: UpdateScheduled 行動 es: UpdateVpc 端點 是:UpgradeDomain es: UpgradeElasticsearch 網域名稱

服務前綴	動作
事件	事件:ActivateEvent來源 事件：CancelReplay 事件:CreateApi目的地 事件：CreateArchive 事件：CreateConnection 事件：CreateEndpoint 事件:CreateEvent公共汽車 事件：CreatePartnerEventSource 事件:DeactivateEvent來源 事件：DeauthorizeConnection 事件:DeleteApi目的地 事件：DeleteArchive 事件：DeleteConnection 事件：DeleteEndpoint 事件:DeleteEvent公共汽車 事件：DeletePartnerEventSource 事件：DeleteRule 事件:DescribeApi目的地 事件：DescribeArchive 事件：DescribeConnection 事件：DescribeEndpoint

服務前綴	動作
	事件:DescribeEvent公共汽車
	事件:DescribeEvent來源
	事件：DescribePartnerEventSource
	事件：DescribeReplay
	事件：DescribeRule
	事件：DisableRule
	事件：EnableRule
	活動:ListApi目的地
	事件：ListArchives
	事件：ListConnections
	事件：ListEndpoints
	事件:ListEvent巴士
	事件:ListEvent來源
	事件:ListPartnerEventSource帳戶
	事件：ListPartnerEventSources
	事件：ListReplays
	事件:ListRuleNamesBy目標
	事件：ListRules
	事件：ListTargetsByRule
	事件：PutPermission
	事件：PutRule

服務前綴	動作
	事件：PutTargets
	事件：RemovePermission
	事件：RemoveTargets
	事件：StartReplay
	事件:TestEvent模式
	事件:UpdateApi目的地
	事件：UpdateArchive
	事件：UpdateConnection
	事件：UpdateEndpoint

服務前綴	動作
evidently	顯然：CreateExperiment 顯然：CreateFeature 顯然：CreateLaunch 顯然：CreateProject 顯然：CreateSegment 顯然：DeleteExperiment 顯然：DeleteFeature 顯然：DeleteLaunch 顯然：DeleteProject 顯然：DeleteSegment 顯然：GetExperiment 顯然：GetExperiment結果 顯然：GetFeature 顯然：GetLaunch 顯然：GetProject 顯然：GetSegment 顯然：ListExperiments 顯然：ListFeatures 顯然：ListLaunches 顯然：ListProjects 顯然：ListSegment參考

服務前綴	動作
	顯然：ListSegments 顯然：StartExperiment 顯然：StartLaunch 顯然：StopExperiment 顯然：StopLaunch 顯然：TestSegment模式 顯然：UpdateExperiment 顯然：UpdateFeature 顯然：UpdateLaunch 顯然：UpdateProject 顯然：UpdateProjectDataDelivery

服務前綴	動作
finspace	最終空間:CreateEnvironment
	空間:CreateKx變更集
	最終空間:叢CreateKx集
	尋找空間:CreateKx資料庫
	尋找空間:CreateKx資料檢視
	空間:環境 CreateKx
	最終空間:CreateKxScalingGroup
	非空間:CreateKx使用者
	尋找空間:磁碟CreateKx區
	最終空間:CreateUser
	最終空間:DeleteEnvironment
	最終空間:叢DeleteKx集
	最終空間:DeleteKxClusterNode
	尋找空間:DeleteKx資料庫
	尋找空間:DeleteKx資料檢視
	空間:環境 DeleteKx
	最終空間:DeleteKxScalingGroup
	非空間:DeleteKx使用者
	尋找空間:磁碟DeleteKx區
	最終空間:GetEnvironment
	空間:GetKx變更集

服務前綴	動作
	最終空間:叢GetKx集 最終空間:GetKxConnectionString 尋找空間:GetKx資料庫 尋找空間:GetKx資料檢視 空間:環境 GetKx 最終空間:GetKxScalingGroup 非空間:GetKx使用者 尋找空間:磁碟GetKx區 尋找空間:狀態 GetLoad SampleData SetGroup IntoEnvironment 最終空間:GetUser 最終空間:ListEnvironments FINSPACE: ListKx 變更集 最終空間:ListKxClusterNodes 最終空間:叢ListKx集 非空間:ListKx資料庫 Finspace: ListKx 資料檢視 最終空間:環境 ListKx 最終空間:ListKxScalingGroups 金融空間:ListKx使用者 尋找空間:磁碟ListKx區 最終空間:ListUsers

服務前綴	動作
	空間:環境 LoadSample DataSet GroupInto 空間:密碼 ResetUser 最終空間:UpdateEnvironment 尋找空間:UpdateKxClusterCode配置 最終空間:UpdateKxClusterDatabases 尋找空間:UpdateKx資料庫 尋找空間:UpdateKx資料檢視 空間:環境 UpdateKx 最終空間:UpdateKxEnvironmentNetwork 非空間:UpdateKx使用者 尋找空間:磁碟UpdateKx區 最終空間:UpdateUser
firehose	防火喉:溪 CreateDelivery 防火喉:溪 DeleteDelivery 防火喉:溪 DescribeDelivery 火喉:ListDelivery溪流 火喉：StartDeliveryStreamEncryption 火喉：StopDeliveryStreamEncryption 火喉：UpdateDestination

服務前綴	動作
fis	FIS：CreateExperiment模板
	FIS：CreateTargetAccountConfiguration
	FIS：DeleteExperiment模板
	FIS：DeleteTargetAccountConfiguration
	FIS：GetAction
	FIS：GetExperiment
	FIS：GetExperimentTargetAccount配置
	FIS：GetExperiment模板
	FIS：GetTargetAccountConfiguration
	FIS：GetTargetResourceType
	FIS：ListActions
	FIS：ListExperimentResolvedTargets
	FIS：ListExperiments
	FIS：ListExperimentTargetAccount配置
	FIS：ListExperiment模板
	FIS：ListTargetAccountConfigurations
	FIS：ListTargetResourceTypes
	FIS：StartExperiment
	FIS：StopExperiment
	FIS：UpdateExperiment模板
	FIS：UpdateTargetAccountConfiguration

服務前綴	動作
fms	fms: AssociateAdmin 帳戶 FMS：AssociateThirdPartyFirewall fms: BatchAssociate 資源 fms: BatchDisassociate 資源 fms: 列DeleteApps表 fms: DeleteNotification 頻道 FMS：DeletePolicy fms: 列DeleteProtocols表 fms: DeleteResource 套裝 fms: DisassociateAdmin 帳戶 FMS：DisassociateThirdPartyFirewall fms: GetAdmin 帳戶 fms: 適用範圍 GetAdmin fms: 列GetApps表 fms: GetCompliance 詳細資料 fms: GetNotification 頻道 FMS：GetPolicy fms: 狀態 GetProtection fms: 列GetProtocols表 fms: GetResource 套裝 FMS：GetThirdPartyFirewallAssociationStatus

服務前綴	動作
	fms: GetViolation 詳細資料 fms: 組織機構 ListAdmin AccountsFor FMS：ListAdminsManagingAccount fms: 列ListApps表 fms: 狀態 ListCompliance fms: ListDiscovered 資源 fms: ListMember 帳戶 FMS：ListPolicies fms: 列ListProtocols表 FMS：ListResourceSetResources fms: ListResource 套裝 FMS：ListThirdPartyFirewallFirewallPolicies fms: PutAdmin 帳戶 fms: 列PutApps表 fms: PutNotification 頻道 FMS：PutPolicy fms: 列PutProtocols表 fms: PutResource 套裝

服務前綴	動作
frauddetector	欺詐探測器：可變 BatchCreate
	欺詐探測器：可變 BatchGet
	欺詐探測器：CancelBatchImportJob
	欺詐探測器：CancelBatchPredictionJob
	欺詐探測器：CreateBatchImportJob
	欺詐探測器：CreateBatchPredictionJob
	欺詐探測器：版本 CreateDetector
	欺詐探測器：CreateList
	欺詐探測器：CreateModel
	欺詐探測器：版本 CreateModel
	欺詐探測器：CreateRule
	欺詐探測器：CreateVariable
	欺詐探測器：DeleteBatchImportJob
	欺詐探測器：DeleteBatchPredictionJob
	欺詐探測器：DeleteDetector
	欺詐探測器：版本 DeleteDetector
	欺詐探測器：類型 DeleteEntity
	欺詐探測器：DeleteEvent
	欺詐探測器：類型 DeleteEvents ByEvent
	欺詐探測器：類型 DeleteEvent
	欺詐探測器：模型 DeleteExternal

服務前綴	動作
	欺詐探測器：DeleteLabel
	欺詐探測器：DeleteList
	欺詐探測器：DeleteModel
	欺詐探測器：版本 DeleteModel
	欺詐探測器：DeleteOutcome
	欺詐探測器：DeleteRule
	欺詐探測器：DeleteVariable
	欺詐探測器：DescribeDetector
	欺詐探測器：版本 DescribeModel
	欺詐探測器：GetBatchImportJobs
	欺詐探測器：GetBatchPredictionJobs
	欺詐探測器：狀態 GetDelete EventsBy EventType
	欺詐探測器：GetDetectors
	欺詐探測器：版本 GetDetector
	欺詐探測器：類型 GetEntity
	欺詐探測器：GetEvent
	欺詐探測器：預測 GetEvent
	欺詐探測器：GetEventPredictionMetadata
	欺詐探測器：類型 GetEvent
	欺詐探測器：模型 GetExternal
	詐騙偵測器:GetKMS EncryptionKey

服務前綴	動作
	欺詐探測器：GetLabels
	欺詐探測器：元素 GetList
	欺詐探測器：元數據 GetLists
	欺詐探測器：GetModels
	欺詐探測器：版本 GetModel
	欺詐探測器：GetOutcomes
	欺詐探測器：GetRules
	欺詐探測器：GetVariables
	欺詐探測器：預測 ListEvent
	欺詐探測器：PutDetector
	欺詐探測器：類型 PutEntity
	欺詐探測器：類型 PutEvent
	欺詐探測器：模型 PutExternal
	欺詐檢測器：公司 EncryptionKey
	欺詐探測器：PutLabel
	欺詐探測器：PutOutcome
	欺詐探測器：SendEvent
	欺詐探測器：版本 UpdateDetector
	欺詐探測器：UpdateDetectorVersionMetadata
	欺詐探測器：UpdateDetectorVersionStatus
	欺詐探測器：標籤 UpdateEvent

服務前綴	動作
	欺詐探測器：UpdateList
	欺詐探測器：UpdateModel
	欺詐探測器：版本 UpdateModel
	欺詐探測器：UpdateModelVersionStatus
	欺詐探測器：元數據 UpdateRule
	欺詐探測器：版本 UpdateRule
	欺詐探測器：UpdateVariable

服務前綴	動作
fsx	FSX：AssociateFileSystemAliases FSX：CancelDataRepositoryTask FSX：CopyBackup FSX：CreateDataRepositoryTask FSX: CreateFile 快取記憶體 FSX: CreateFile 系統 FSX: Backup CreateFile SystemFrom FSX：CreateSnapshot FSX：CreateStorageVirtualMachine FSX：CreateVolume FSX：CreateVolumeFromBackup FSX：DeleteBackup FSX: DeleteFile 快取記憶體 FSX: DeleteFile 系統 FSX：DeleteSnapshot FSX：DeleteStorageVirtualMachine FSX：DeleteVolume FSX：DescribeBackups FSX：DescribeDataRepositoryAssociations FSX：DescribeDataRepositoryTasks FSX: DescribeFile 快取記憶體

服務前綴	動作
	FSX：DescribeFileSystemAliases FSX: DescribeFile 系統 FSX：DescribeSharedVpcConfiguration FSX：DescribeSnapshots FSX：DescribeStorageVirtualMachines FSX：DescribeVolumes FSX：DisassociateFileSystemAliases FSX: 3 鎖 ReleaseFile SystemNfs FSX：RestoreVolumeFromSnapshot FSX：StartMisconfiguredStateRecovery FSX：UpdateDataRepositoryAssociation FSX: UpdateFile 快取記憶體 FSX: UpdateFile 系統 FSX：UpdateSharedVpcConfiguration FSX：UpdateSnapshot FSX：UpdateStorageVirtualMachine FSX：UpdateVolume

服務前綴	動作
gamelift	遊戲：AcceptMatch
	遊戲:伺服器 ClaimGame
	遊戲：CreateAlias
	遊戲：CreateBuild
	遊戲：CreateContainerGroupDefinition
	遊戲：CreateFleet
	遊戲:位置 CreateFleet
	遊戲：CreateGameServerGroup
	遊戲:工作階段 CreateGame
	遊戲：CreateGameSessionQueue
	遊戲：CreateLocation
	遊戲:配置 CreateMatchmaking
	遊戲：CreateMatchmakingRuleSet
	遊戲:工作階段 CreatePlayer
	遊戲:工作階段 CreatePlayer
	遊戲：CreateScript
	遊戲：CreateVpcPeeringAuthorization
	遊戲：CreateVpcPeeringConnection
	遊戲：DeleteAlias
	遊戲：DeleteBuild
	遊戲：DeleteContainerGroupDefinition

服務前綴	動作
	遊戲：DeleteFleet
	遊戲:位置 DeleteFleet
	遊戲：DeleteGameServerGroup
	遊戲：DeleteGameSessionQueue
	遊戲：DeleteLocation
	遊戲:配置 DeleteMatchmaking
	遊戲：DeleteMatchmakingRuleSet
	遊戲:政策 DeleteScaling
	遊戲：DeleteScript
	遊戲：DeleteVpcPeeringAuthorization
	遊戲：DeleteVpcPeeringConnection
	遊戲：DeregisterCompute
	遊戲:伺服器 DeregisterGame
	遊戲：DescribeAlias
	遊戲：DescribeBuild
	遊戲：DescribeCompute
	遊戲：DescribeContainerGroupDefinition
	遊戲化:說明 EC2 InstanceLimits
	遊戲:屬性 DescribeFleet
	遊戲:容量 DescribeFleet
	遊戲:事件 DescribeFleet

服務前綴	動作
	遊戲：DescribeFleetLocationAttributes
	遊戲：DescribeFleetLocationCapacity
	遊戲：DescribeFleetLocationUtilization
	遊戲：DescribeFleetPortSettings
	遊戲:利用率 DescribeFleet
	遊戲:伺服器 DescribeGame
	遊戲：DescribeGameServerGroup
	遊戲：DescribeGameServerInstances
	遊戲：DescribeGameSessionDetails
	遊戲：DescribeGameSessionPlacement
	遊戲：DescribeGameSessionQueues
	遊戲:工作階段 DescribeGame
	遊戲：DescribeInstances
	遊戲：DescribeMatchmaking
	遊戲:配置 DescribeMatchmaking
	遊戲：DescribeMatchmakingRuleSets
	遊戲:工作階段 DescribePlayer
	遊戲:配置 DescribeRuntime
	遊戲:政策 DescribeScaling
	遊戲：DescribeScript
	遊戲：DescribeVpcPeeringAuthorizations

服務前綴	動作
	遊戲：DescribeVpcPeeringConnections
	遊戲:訪問 GetCompute
	遊戲：GetComputeAuthToken
	遊戲：網址 GetGame SessionLog
	遊戲:訪問 GetInstance
	遊戲：ListAliases
	遊戲：ListBuilds
	遊戲：ListCompute
	遊戲：ListContainerGroupDefinitions
	遊戲：ListFleets
	遊戲：ListGameServerGroups
	遊戲:伺服器 ListGame
	遊戲：ListLocations
	遊戲：ListScripts
	遊戲:政策 PutScaling
	遊戲：RegisterCompute
	遊戲:伺服器 RegisterGame
	遊戲:憑證 RequestUpload
	遊戲：ResolveAlias
	遊戲：ResumeGameServerGroup
	遊戲:工作階段 SearchGame

服務前綴	動作
	遊戲:動作 StartFleet
	遊戲：StartGameSessionPlacement
	遊戲:回填 StartMatch
	遊戲：StartMatchmaking
	遊戲:動作 StopFleet
	遊戲：StopGameSessionPlacement
	遊戲：StopMatchmaking
	遊戲：SuspendGameServerGroup
	遊戲：UpdateAlias
	遊戲：UpdateBuild
	遊戲:屬性 UpdateFleet
	遊戲:容量 UpdateFleet
	遊戲：UpdateFleetPortSettings
	遊戲:伺服器 UpdateGame
	遊戲：UpdateGameServerGroup
	遊戲:工作階段 UpdateGame
	遊戲：UpdateGameSessionQueue
	遊戲:配置 UpdateMatchmaking
	遊戲:配置 UpdateRuntime
	遊戲：UpdateScript
	遊戲：ValidateMatchmakingRuleSet

服務前綴	動作
geo	地理:AssociateTracker消費者 地理:BatchDeleteDevicePosition歷史 地理:地理BatchDelete圍欄 地理:地理BatchEvaluate圍欄 地理：BatchGetDevicePosition 地理:地理BatchPut圍欄 地理：BatchUpdateDevicePosition 地理：CalculateRoute 地理:CalculateRoute矩陣 地理:CreateGeofence集合 地理：CreateMap 地理:CreatePlace索引 地理:CreateRoute計算器 地理：CreateTracker 地理:DeleteGeofence集合 地理：DeleteKey 地理：DeleteMap 地理:DeletePlace索引 地理:DeleteRoute計算器 地理：DeleteTracker 地理:DescribeGeofence集合

服務前綴	動作
	地理：DescribeKey 地理：DescribeMap 地理:DescribePlace索引 地理:DescribeRoute計算器 地理：DescribeTracker 地理:DisassociateTracker消費者 地理GetDevice位置：位置 地理：GetDevicePositionHistory 地理：GetGeofence 地理:GetMap字形 地理:GetMap精靈 地理：GetMapStyleDescriptor 地理:GetMap瓷磚 地理：GetPlace 地理ListDevice位置：位置 地理:ListGeofence集合 地理：ListGeofences 地理：ListKeys 地理：ListMaps 地理:ListPlace索引 地理:ListRoute計算器

服務前綴	動作
	地理:ListTracker消費者 地理：ListTrackers 地理：PutGeofence 地理SearchPlaceIndexFor位置：位置 地理:SearchPlaceIndexFor建議 地理：SearchPlaceIndexFor文本 地理:UpdateGeofence集合 地理：UpdateKey 地理：UpdateMap 地理:UpdatePlace索引 地理:UpdateRoute計算器 地理：UpdateTracker

服務前綴	動作
glacier	冰川：AbortMultipart上傳 冰川:AbortVault鎖 冰川：CompleteMultipart上傳 冰川:CompleteVault鎖 冰川：CreateVault 冰川：DeleteArchive 冰川：DeleteVault 冰川：DeleteVaultAccessPolicy 冰川:DeleteVault通知 冰川：DescribeJob 冰川：DescribeVault 冰川：GetDataRetrievalPolicy 冰川：GetJob輸出 冰川：GetVaultAccessPolicy 冰川:GetVault鎖 冰川:GetVault通知 冰川：InitiateJob 冰川：InitiateMultipart上傳 冰川:InitiateVault鎖 冰川：ListJobs 冰川：ListMultipart上傳

服務前綴	動作
	冰川：ListParts 冰川：ListProvisioned容量 冰川：ListVaults 冰川：PurchaseProvisioned容量 冰川：SetDataRetrievalPolicy 冰川：SetVaultAccessPolicy 冰川:SetVault通知 冰川：UploadArchive 冰川:UploadMultipart部分

服務前綴	動作
grafana	說:AssociateLicense 說:CreateWorkspace 說:CreateWorkspaceApiKey 說:DeleteWorkspace 說:DeleteWorkspaceApiKey 說:DescribeWorkspace 描述：身份驗證 DescribeWorkspace 描述：配置 DescribeWorkspace 說:DisassociateLicense 說:ListPermissions 說:ListVersions 說:ListWorkspaces 說:UpdatePermissions 說:UpdateWorkspace 描述：身份驗證 UpdateWorkspace 描述：配置 UpdateWorkspace

服務前綴	動作
greengrass	格雷格拉斯：AssociateRoleToGroup
	希望:帳戶 AssociateService RoleTo
	格雷格拉斯:設備 BatchAssociate ClientDevice WithCore
	格雷格拉斯:設備 BatchDisassociate ClientDevice FromCore
	格雷格拉斯：CancelDeployment
	格雷格拉斯：版本 CreateComponent
	格雷格拉斯：定義 CreateConnector
	格雷格拉斯：CreateConnectorDefinitionVersion
	格雷格拉斯：定義 CreateCore
	格雷格拉斯：CreateCoreDefinitionVersion
	格雷格拉斯：CreateDeployment
	格雷格拉斯：定義 CreateDevice
	格雷格拉斯：CreateDeviceDefinitionVersion
	格雷格拉斯：定義 CreateFunction
	格雷格拉斯：CreateFunctionDefinitionVersion
	格雷格拉斯：CreateGroup
	格雷格拉斯：CreateGroupCertificateAuthority
	格雷格拉斯：版本 CreateGroup
	格雷格拉斯：定義 CreateLogger
	格雷格拉斯：CreateLoggerDefinitionVersion
	格雷格拉斯：定義 CreateResource

服務前綴	動作
	格雷格拉斯：CreateResourceDefinitionVersion
	格雷格拉斯：CreateSoftwareUpdateJob
	格雷格拉斯：定義 CreateSubscription
	格雷格拉斯：CreateSubscriptionDefinitionVersion
	格雷格拉斯：DeleteComponent
	格雷格拉斯：定義 DeleteConnector
	格雷格拉斯：定義 DeleteCore
	格雷格拉斯:設備 DeleteCore
	格雷格拉斯：DeleteDeployment
	格雷格拉斯：定義 DeleteDevice
	格雷格拉斯：定義 DeleteFunction
	格雷格拉斯：DeleteGroup
	格雷格拉斯：定義 DeleteLogger
	格雷格拉斯：定義 DeleteResource
	格雷格拉斯：定義 DeleteSubscription
	格雷格拉斯：DescribeComponent
	格雷格拉斯：DisassociateRoleFromGroup
	希望:帳戶 DisassociateService RoleFrom
	格雷格拉斯:角色 GetAssociated
	格雷格拉斯：GetBulkDeploymentStatus
	格雷格拉斯：GetComponent

服務前綴	動作
	格雷格拉斯：GetComponentVersionArtifact
	格雷格拉斯：信息 GetConnectivity
	格雷格拉斯：定義 GetConnector
	格雷格拉斯：GetConnectorDefinitionVersion
	格雷格拉斯：定義 GetCore
	格雷格拉斯：GetCoreDefinitionVersion
	格雷格拉斯:設備 GetCore
	格雷格拉斯：GetDeployment
	格雷格拉斯：狀態 GetDeployment
	格雷格拉斯：定義 GetDevice
	格雷格拉斯：GetDeviceDefinitionVersion
	格雷格拉斯：定義 GetFunction
	格雷格拉斯：GetFunctionDefinitionVersion
	格雷格拉斯：GetGroup
	格雷格拉斯：GetGroupCertificateAuthority
	格雷格拉斯：GetGroupCertificateConfiguration
	格雷格拉斯：版本 GetGroup
	格雷格拉斯：定義 GetLogger
	格雷格拉斯：GetLoggerDefinitionVersion
	格雷格拉斯：定義 GetResource
	格雷格拉斯：GetResourceDefinitionVersion

服務前綴	動作
	希望:帳戶 GetService RoleFor
	格雷格拉斯：定義 GetSubscription
	格雷格拉斯：GetSubscriptionDefinitionVersion
	格雷格拉斯：GetThingRuntimeConfiguration
	格雷格拉斯:報告 ListBulk DeploymentDetailed
	環境:部署 ListBulk
	格雷格拉斯:設備 ListClient DevicesAssociated WithCore
	格雷格拉斯：ListComponents
	格雷格拉斯：版本 ListComponent
	格雷格拉斯：定義 ListConnector
	格雷格拉斯：ListConnectorDefinitionVersions
	格雷格拉斯：定義 ListCore
	格雷格拉斯：ListCoreDefinitionVersions
	格雷格拉斯:設備 ListCore
	格雷格拉斯：ListDeployments
	格雷格拉斯：定義 ListDevice
	格雷格拉斯：ListDeviceDefinitionVersions
	環境:部署 ListEffective
	格雷格拉斯：定義 ListFunction
	格雷格拉斯：ListFunctionDefinitionVersions
	格雷格拉斯：ListGroupCertificateAuthorities

服務前綴	動作
	格雷格拉斯：ListGroups
	格雷格拉斯：版本 ListGroup
	綠色:元件 ListInstalled
	格雷格拉斯：定義 ListLogger
	格雷格拉斯：ListLoggerDefinitionVersions
	格雷格拉斯：定義 ListResource
	格雷格拉斯：ListResourceDefinitionVersions
	格雷格拉斯：定義 ListSubscription
	格雷格拉斯：ListSubscriptionDefinitionVersions
	格雷格拉斯：ResetDeployments
	格雷格拉斯:部署 StartBulk
	格雷格拉斯:部署 StopBulk
	格雷格拉斯：信息 UpdateConnectivity
	格雷格拉斯：定義 UpdateConnector
	格雷格拉斯：定義 UpdateCore
	格雷格拉斯：定義 UpdateDevice
	格雷格拉斯：定義 UpdateFunction
	格雷格拉斯：UpdateGroup
	格雷格拉斯：UpdateGroupCertificateConfiguration
	格雷格拉斯：定義 UpdateLogger
	格雷格拉斯：定義 UpdateResource

服務前綴	動作
	格雷格拉斯：定義 UpdateSubscription 格雷格拉斯：UpdateThingRuntimeConfiguration

服務前綴	動作
groundstation	地面站：CancelContact 地面站：CreateConfig 地面站：CreateDataflowEndpointGroup 地面站：CreateEphemeris 地面站：CreateMission配置文件 地面站：DeleteConfig 地面站：DeleteDataflowEndpointGroup 地面站：DeleteEphemeris 地面站：DeleteMission配置文件 地面站：DescribeContact 地面站：DescribeEphemeris 地面站：GetConfig 地面站：GetDataflowEndpointGroup 發電站:GetMinute用法 地面站：GetMission配置文件 地面站：GetSatellite 地面站：ListConfigs 地面站：ListContacts 地面站：ListDataflowEndpointGroups 地面站：ListEphemerides 發地站:ListGround車站

服務前綴	動作
	地面站:ListMission配置文件 地面站：ListSatellites 地面站：RegisterAgent 地面站：ReserveContact 地基站：狀態 UpdateAgent 地面站：UpdateConfig 地面站：UpdateEphemeris 地面站：UpdateMission配置文件

服務前綴	動作
guardduty	守衛：邀請 AcceptAdministrator
	守衛：AcceptInvitation
	守衛：ArchiveFindings
	守衛：CreateDetector
	守衛：CreateFilter
	guardduty:CreateIPSet
	守衛：CreateMembers
	警衛：目的地 CreatePublishing
	守衛：發現 CreateSample
	守衛：CreateThreatIntelSet
	守衛：DeclineInvitations
	守衛：DeleteDetector
	守衛：DeleteFilter
	守衛：DeleteInvitations
	guardduty:DeleteIPSet
	守衛：DeleteMembers
	警衛：目的地 DeletePublishing
	守衛：DeleteThreatIntelSet
	守衛：掃描 DescribeMalware
	守衛：配置 DescribeOrganization
	警衛：目的地 DescribePublishing

服務前綴	動作
	守衛：DisableOrganizationAdminAccount
	守衛：DisassociateFromAdministratorAccount
	守衛：DisassociateFromMasterAccount
	守衛：DisassociateMembers
	守衛：EnableOrganizationAdminAccount
	保安：帳戶 GetAdministrator
	衛兵：統計 GetCoverage
	守衛：GetDetector
	守衛：GetFilter
	守衛：GetFindings
	衛兵：統計 GetFindings
	守衛：計數 GetInvitations
	guardduty:GetIPSet
	守衛：GetMalwareScanSettings
	保安：帳戶 GetMaster
	守衛：探測器 GetMember
	守衛：GetMembers
	衛兵：統計 GetOrganization
	守衛：天 GetRemaining FreeTrial
	守衛：GetThreatIntelSet
	衛兵：統計 GetUsage

服務前綴	動作
	守衛：InviteMembers
	守衛：ListCoverage
	守衛：ListDetectors
	守衛：ListFilters
	守衛：ListFindings
	守衛：ListInvitations
	guardduty:ListIPSets
	守衛：ListMembers
	守衛：ListOrganizationAdminAccounts
	警衛：目的地 ListPublishing
	守衛：ListThreatIntelSets
	守衛：遙測 SendSecurity
	守衛：掃描 StartMalware
	保安：成員 StartMonitoring
	保安：成員 StopMonitoring
	守衛：UnarchiveFindings
	守衛：UpdateDetector
	守衛：UpdateFilter
	守衛：反饋 UpdateFindings
	guardduty:UpdateIPSet
	守衛：UpdateMalwareScanSettings

服務前綴	動作
	守衛：探測器 UpdateMember 守衛：配置 UpdateOrganization 警衛：目的地 UpdatePublishing 守衛：UpdateThreatIntelSet
healthlake	healthlake:CreateFHIRDatastore 健康湖：CreateResource healthlake:DeleteFHIRDatastore 健康湖：DeleteResource healthlake:DescribeFHIRDatastore 健康湖:描述 ExportJob 健康湖:描述 ImportJob 健康湖：GetCapabilities healthlake:ListFHIRDatastores 健康湖:列表法希尔 ExportJobs 健康湖:列表法希尔 ImportJobs 健康湖：ReadResource 健康湖:SearchWith獲取 健康湖:郵政 SearchWith 健康湖:星期六 ExportJob 健康湖:斯塔法希爾 ImportJob 健康湖：UpdateResource

服務前綴	動作
honeycode	蜂蜜碼：BatchCreateTableRows
	蜂蜜碼：BatchDeleteTableRows
	蜂蜜碼：BatchUpdateTableRows
	蜂蜜碼：BatchUpsertTableRows
	蜂蜜碼：Job DescribeTable DataImport
	蜂蜜碼:GetScreen資料
	蜂蜜碼：InvokeScreen自動化
	蜂蜜碼：列 ListTable
	蜂蜜碼：ListTable行
	蜂蜜碼：ListTables
	蜂蜜碼：QueryTable行
	蜂蜜碼：Job StartTable DataImport

服務前綴	動作
iam	IAM：AddClient身份證識別ToOpen碼 ConnectProvider IAM：AddRoleToInstance設定檔 IAM：AddUserToGroup IAM：AttachGroup政策 IAM：AttachRole政策 IAM：AttachUser政策 IAM：ChangePassword IAM: CreateAccess 金鑰 IAM: CreateAccount 別名 IAM：CreateGroup IAM：CreateInstance設定檔 IAM：CreateLogin設定檔 IAM：CreateOpen身份證 ConnectProvider IAM：CreatePolicy IAM: CreatePolicy 版本 IAM：CreateRole iam:CreateSAMLProvider IAM：CreateServiceLinkedRole IAM：CreateServiceSpecificCredential IAM：CreateUser 我：CreateVirtualMFA 設備

服務前綴	動作
	iam:DeactivateMFADevice IAM: DeleteAccess 金鑰 IAM: DeleteAccount 別名 IAM：DeleteAccountPasswordPolicy IAM：DeleteCloudFrontPublic關鍵 IAM：DeleteGroup IAM：DeleteGroup政策 IAM：DeleteInstance設定檔 IAM：DeleteLogin設定檔 IAM：DeleteOpen身份證 ConnectProvider IAM：DeletePolicy IAM: DeletePolicy 版本 IAM：DeleteRole IAM：DeleteRolePermissionsBoundary IAM：DeleteRole政策 iam:DeleteSAMLProvider IAM：DeleteServer證書 IAM：DeleteServiceLinkedRole IAM：DeleteServiceSpecificCredential IAM：DeleteSigning證書 介面:刪除安全殼層 PublicKey

服務前綴	動作
	IAM：DeleteUser IAM：DeleteUserPermissionsBoundary IAM：DeleteUser政策 我：DeleteVirtualMFA 設備 IAM：DetachGroup政策 IAM：DetachRole政策 IAM：DetachUser政策 iam:EnableMFADevice IAM: GenerateCredential 報告 IAM：GenerateOrganizationsAccessReport IAM: GenerateService LastAccessed 詳細資訊 IAM：GetAccessKeyLast用過 IAM：GetAccountAuthorizationDetails IAM：GetAccountEmailAddress IAM：GetAccount名稱 IAM：GetAccountPasswordPolicy IAM：GetAccount摘要 IAM：GetCloudFrontPublic關鍵 IAM：GetContextKeysForCustomPolicy IAM：GetContextKeysForPrincipalPolicy IAM: GetCredential 報告

服務前綴	動作
	IAM：GetGroup IAM：GetGroup政策 IAM：GetInstance設定檔 IAM：GetLogin設定檔 iam:GetMFADevice IAM：GetOpen身份證 ConnectProvider IAM：GetOrganizationsAccessReport IAM：GetPolicy IAM: GetPolicy 版本 IAM：GetRole IAM：GetRole政策 iam:GetSAMLProvider IAM：GetServer證書 IAM: GetService LastAccessed 詳細資訊 IAM：GetServiceLastAccessedDetailsWith實體 IAM：GetServiceLinkedRoleDeletionStatus 家庭服務:獲取 PublicKey IAM：GetUser IAM：GetUser政策 IAM: ListAccess 金鑰 IAM：ListAccount別名

服務前綴	動作
	IAM：ListAttachedGroupPolicies
	IAM：ListAttachedRolePolicies
	IAM：ListAttachedUserPolicies
	IAM: ListCloud FrontPublic 金鑰
	IAM：ListEntitiesForPolicy
	IAM：ListGroup政策
	IAM：ListGroups
	IAM：ListGroupsForUser
	IAM：ListInstance配置文件
	IAM：ListInstanceProfilesFor角色
	iam:ListMFADevices
	IAM：ListOpen身份證 ConnectProviders
	IAM：ListPolicies
	IAM：ListPoliciesGrantingService訪問
	IAM：ListPolicy版本
	IAM：ListRole政策
	IAM：ListRoles
	iam:ListSAMLProviders
	IAM：ListServer證書
	IAM：ListServiceSpecificCredentials
	IAM：ListSigning證書

服務前綴	動作
	介面:列出安全殼層 PublicKeys IAMRegionalEndpoints: 列出狀態 IAM：ListUser政策 IAM：ListUsers 我：ListVirtualMFA 設備 IAM：PutGroup政策 IAM：PutRolePermissionsBoundary IAM：PutRole政策 IAM：PutUserPermissionsBoundary IAM：PutUser政策 IAM：RemoveClient身份證識別FromOpen碼 ConnectProvider IAM：RemoveRoleFromInstance設定檔 IAM：RemoveUserFromGroup IAM：ResetServiceSpecificCredential iam:ResyncMFADevice IAM：SetDefaultPolicyVersion IAM：SetSecurityTokenService偏好設定 IAMRegionalEndpoint: 設定狀態 IAM：SimulateCustom政策 IAM：SimulatePrincipal政策 IAM: UpdateAccess 金鑰

服務前綴	動作
	IAM：UpdateAccountEmailAddress IAM：UpdateAccount名稱 IAM：UpdateAccountPasswordPolicy IAM：UpdateAssumeRolePolicy IAM：UpdateCloudFrontPublic關鍵 IAM：UpdateGroup IAM：UpdateLogin設定檔 IAM: UpdateOpen ID ConnectProvider 指紋 IAM：UpdateRole IAM：UpdateRole說明 iam:UpdateSAMLProvider IAM：UpdateServer證書 IAM：UpdateServiceSpecificCredential IAM：UpdateSigning證書 介面:更新 SSH PublicKey IAM：UpdateUser IAM：UploadCloudFrontPublic關鍵 IAM：UploadServer證書 IAM：UploadSigning證書 介面:上傳安全殼層 PublicKey

服務前綴	動作
identitystore	身份存儲：CreateGroup 身分儲存庫：成員資格 CreateGroup 身份存儲：CreateUser 身份存儲：DeleteGroup 身分儲存庫：成員資格 DeleteGroup 身份存儲：DeleteUser 身份存儲：DescribeGroup 身分儲存庫：成員資格 DescribeGroup 身份存儲：DescribeUser 身份存儲：Id GetGroup 身份存儲：GetGroupMembershipId 身份存儲：Id GetUser 身份存儲：IsMemberInGroups 身份存儲：成員資格 ListGroup 身份存儲:成員 ListGroup MembershipsFor 身份存儲：ListGroups 身份存儲：ListUsers 身份存儲：UpdateGroup 身份存儲：UpdateUser

服務前綴	動作
imagebuilder	圖像生成器：CancelImage創建 圖像生成器：執CancelLifecycle行 圖像生成器：CreateComponent 圖像生成器：食譜 CreateContainer 圖像生成器：CreateDistribution配置 圖像生成器：CreateImage 圖像生成器：CreateImage管道 圖像生成器：食譜 CreateImage 圖像生成器：CreateInfrastructure配置 圖像生成器：策略 CreateLifecycle 圖像生成器：CreateWorkflow 圖像生成器：DeleteComponent 圖像生成器：食譜 DeleteContainer 圖像生成器：DeleteDistribution配置 圖像生成器：DeleteImage 圖像生成器：DeleteImage管道 圖像生成器：食譜 DeleteImage 圖像生成器：DeleteInfrastructure配置 圖像生成器：策略 DeleteLifecycle 圖像生成器：DeleteWorkflow 圖像生成器：策略 GetComponent

服務前綴	動作
	圖像生成器：GetContainerRecipePolicy
	圖像生成器：策略 GetImage
	圖像生成器：GetImageRecipePolicy
	圖像生成器：執GetLifecycle行
	圖像生成器：策略 GetLifecycle
	圖像生成器：執GetWorkflow行
	圖像生成器：GetWorkflowStepExecution
	圖像生成器：ImportComponent
	圖像生成器：ImportVm圖像
	圖像生成器：ListComponentBuildVersions
	圖像生成器：ListComponents
	圖像生成器：食譜 ListContainer
	圖像生成器：ListDistribution配置
	圖像生成器：ListImageBuildVersions
	圖像生成器：ListImage軟件包
	圖像生成器：ListImagePipelineImages
	圖像生成器：ListImage管道
	圖像生成器：食譜 ListImage
	圖像生成器：ListImages
	圖像生成器：聚ListImageScanFinding合
	圖像生成器：ListImageScanFindings

服務前綴	動作
	圖像生成器：ListInfrastructure配置 圖像生成器：ListLifecycleExecutionResources 圖像生成器：執ListLifecycle行 圖像生成器：策略 ListLifecycle 圖像生成器：ListWaitingWorkflowSteps 圖像生成器：執ListWorkflow行 圖像生成器：ListWorkflows 圖像生成器：ListWorkflowStepExecutions 圖像生成器：策略 PutComponent 圖像生成器：PutContainerRecipePolicy 圖像生成器：策略 PutImage 圖像生成器：PutImageRecipePolicy 圖像生成器：SendWorkflowStepAction 圖像生成器：StartImagePipelineExecution 圖像生成器：StartResourceStateUpdate 圖像生成器：UpdateDistribution配置 圖像生成器：UpdateImage管道 圖像生成器：UpdateInfrastructure配置

服務前綴	動作
inspector	督察：AddAttributesToFindings
	檢查員：CreateAssessment目標
	檢查員:CreateAssessment範本
	檢查器：CreateExclusions預覽
	檢查員：CreateResource組
	檢查員：DeleteAssessment運行
	檢查員：DeleteAssessment目標
	檢查員:DeleteAssessment範本
	檢查器：DescribeAssessment運行
	檢查員：DescribeAssessment目標
	檢查員：DescribeAssessment模板
	檢查員：DescribeCrossAccountAccess角色
	督察：DescribeExclusions
	督察：DescribeFindings
	檢查員：DescribeResource群組
	檢查員：DescribeRules套件
	檢查員：GetAssessment報告
	檢查器：GetExclusions預覽
	檢查器:GetTelemetry中繼資
	督察：ListAssessmentRunAgents
	檢查器：ListAssessment運行

服務前綴	動作
	檢查員：ListAssessment目標
	檢查員：ListAssessment模板
	檢查員：ListEvent訂閱
	督察：ListExclusions
	督察：ListFindings
	檢查員：ListRules套件
	督察：PreviewAgents
	檢查員：RegisterCrossAccountAccess角色
	督察：RemoveAttributesFromFindings
	檢查員：StartAssessment運行
	檢查員：StopAssessment運行
	檢查員：SubscribeTo事件
	檢查員：UnsubscribeFrom事件
	檢查員：UpdateAssessment目標

服務前綴	動作
inspector2	檢查員 2：AssociateMember 檢查員 2：BatchGetAccountStatus 檢查員 2：BatchGetCodeSnippet 檢查員 2：BatchGetFindingDetails 檢查員 2：信息 BatchGet FreeTrial 檢查員 2：2 狀態 BatchGet MemberEc DeepInspection 檢查員 2：2 狀態 BatchUpdate MemberEc DeepInspection 檢查員 2：報告 CancelFindings 檢查員 2：出口 CancelSbom 檢查員 2：CreateCisScanConfiguration 檢查員 2：CreateFilter 檢查員 2：報告 CreateFindings 檢查員 2：出口 CreateSbom 檢查員 2：DeleteCisScanConfiguration 檢查員 2：DeleteFilter 檢查器 2：配置 DescribeOrganization inspector2:Disable 檢查員 2：DisableDelegatedAdminAccount 檢查員 2：DisassociateMember inspector2:Enable 檢查員 2：EnableDelegatedAdminAccount

服務前綴	動作
	檢查員 2：GetCisScanReport 檢查員 2：詳細信息 GetCis ScanResult 檢查員 2：GetConfiguration 檢查員 2：GetDelegatedAdminAccount 檢查器 2：GetEc2 配置 DeepInspection 檢查員 2：關鍵 GetEncryption 檢查員 2：GetFindingsReportStatus 檢查員 2：GetMember 檢查員 2：出口 GetSbom 檢查員 2：權限 ListAccount 檢查員 2：ListCisScanConfigurations 檢查員 2：檢查 ListCis ScanResults AggregatedBy 檢查員 2：ListCisScanResultsAggregatedByTargetResource 檢查員 2：掃描 ListCis 檢查員 2：ListCoverage 督察 2：統計 ListCoverage 檢查員 2：ListDelegatedAdminAccounts 檢查員 2：ListFilters 檢查員 2：聚合 ListFinding 檢查員 2：ListFindings 檢查員 2：ListMembers

服務前綴	動作
	檢查員 2：總計 ListUsage 檢查員 2：關鍵 ResetEncryption 檢查員 2：SearchVulnerabilities 檢查員 2：SendCisSessionHealth 檢查員 2：SendCisSessionTelemetry 檢查員 2：會話 StartCis 檢查員 2：會話 StopCis 檢查員 2：UpdateCisScanConfiguration 檢查員 2：UpdateConfiguration 檢查器 2：UpdateEc2 配置 DeepInspection 檢查員 2：關鍵 UpdateEncryption 檢查員 2：UpdateFilter 檢查器 2：配置 UpdateOrganization 檢查器 2：UpdateOrgEC2 配置 DeepInspection

服務前綴	動作
iot	物聯網:AcceptCertificate轉移
	物聯網:AddThingToBilling集團
	物聯網:AddThingToThing集團
	物聯網:AssociateTargetsWithJob
	物聯網:AttachPolicy
	物聯網:AttachPrincipal政策
	物聯網：AttachSecurity設定檔
	物聯網:AttachThing首席
	物聯網:CancelAuditMitigationActions任務
	物聯網:CancelAudit任務
	物聯網:CancelCertificate轉移
	物聯網:CancelDetectMitigationActions任務
	物聯網:CancelJob
	物聯網:CancelJob執行
	物聯網:ClearDefault授權者
	物聯網:ConfirmTopicRuleDestination
	物聯網:CreateAudit抑制
	物聯網:CreateAuthorizer
	物聯網:CreateBilling集團
	物聯網:CreateCertificateFromCsr
	物聯網:CreateCertificate供應商

服務前綴	動作
	物聯網:CreateCustom公制
	物聯網:CreateDimension
	物聯網:CreateDomain配置
	物聯網:CreateDynamicThingGroup
	物聯網:CreateFleet公制
	物聯網:CreateJob
	物聯網:CreateJob模板
	物聯網:CreateKeysAndCertificate
	物聯網:CreateMitigation行動
	iot:CreateOTAUpdate
	物聯網:CreatePackage
	物聯網:CreatePackage版本
	物聯網:CreatePolicy
	物聯網:CreatePolicy版本
	物聯網:CreateProvisioning索賠
	物聯網:CreateProvisioning模板
	物聯網:CreateProvisioningTemplateVersion
	物聯網:CreateRole別名
	物聯網:CreateScheduled審計
	物聯網：CreateSecurity設定檔
	物聯網:CreateStream

服務前綴	動作
	物聯網:CreateThing
	物聯網:CreateThing集團
	物聯網:CreateThing類型
	物聯網:CreateTopic規則
	物聯網:CreateTopicRuleDestination
	物聯網:DeleteAccountAuditConfiguration
	物聯網:DeleteAudit抑制
	物聯網:DeleteAuthorizer
	物聯網:DeleteBilling集團
	iot:DeleteCACertificate
	物聯網:DeleteCertificate
	物聯網:DeleteCertificate供應商
	物聯網:DeleteCustom公制
	物聯網:DeleteDimension
	物聯網:DeleteDomain配置
	物聯網:DeleteDynamicThingGroup
	物聯網:DeleteFleet公制
	物聯網:DeleteJob
	物聯網:DeleteJob執行
	物聯網:DeleteJob模板
	物聯網:DeleteMitigation行動

服務前綴	動作
	iot:DeleteOTAUpdate
	物聯網:DeletePackage
	物聯網:DeletePackage版本
	物聯網:DeletePolicy
	物聯網:DeletePolicy版本
	物聯網:DeleteProvisioning模板
	物聯網:DeleteProvisioningTemplateVersion
	物聯網:DeleteRegistration代碼
	物聯網:DeleteRole別名
	物聯網:DeleteScheduled審計
	物聯網：DeleteSecurity設定檔
	物聯網:DeleteStream
	物聯網:DeleteThing
	物聯網:DeleteThing集團
	物聯網:DeleteThing類型
	物聯網:DeleteTopic規則
	物聯網:DeleteTopicRuleDestination
	物聯網:刪除 V2 LoggingLevel
	物聯網:DeprecateThing類型
	物聯網:DescribeAccountAuditConfiguration
	物聯網:DescribeAudit尋找

服務前綴	動作
	物聯網:DescribeAuditMitigationActions任務
	物聯網:DescribeAudit抑制
	物聯網:DescribeAudit任務
	物聯網:DescribeAuthorizer
	物聯網:DescribeBilling集團
	iot:DescribeCACertificate
	物聯網:DescribeCertificate
	物聯網:DescribeCertificate供應商
	物聯網:DescribeCustom公制
	物聯網:DescribeDefault授權者
	物聯網:DescribeDetectMitigationActions任務
	物聯網:DescribeDimension
	物聯網:DescribeDomain配置
	物聯網:DescribeEndpoint
	物聯網：DescribeEvent配置
	物聯網:DescribeFleet公制
	物聯網:DescribeIndex
	物聯網:DescribeJob
	物聯網:DescribeJob執行
	物聯網:DescribeJob模板
	物聯網:DescribeManagedJobTemplate

服務前綴	動作
	物聯網:DescribeMitigation行動
	物聯網:DescribeProvisioning模板
	物聯網:DescribeProvisioningTemplateVersion
	物聯網:DescribeRole別名
	物聯網:DescribeScheduled審計
	物聯網：DescribeSecurity設定檔
	物聯網:DescribeStream
	物聯網:DescribeThing
	物聯網:DescribeThing集團
	物聯網:DescribeThingRegistrationTask
	物聯網:DescribeThing類型
	物聯網:DetachPolicy
	物聯網:DetachPrincipal政策
	物聯網：DetachSecurity設定檔
	物聯網:DetachThing首席
	物聯網:DisableTopic規則
	物聯網:EnableTopic規則
	物聯網:GetBehaviorModelTraining摘要
	物聯網:GetBuckets聚合
	物聯網:GetCardinality
	物聯網:GetEffective政策

服務前綴	動作
	物聯網:GetJob文件
	物聯網:GetLogging選項
	iot:GetOTAUpdate
	物聯網:GetPackage
	物聯網:GetPackage配置
	物聯網:GetPackage版本
	物聯網:GetPercentiles
	物聯網:GetPolicy
	物聯網:GetPolicy版本
	物聯網:GetRegistration代碼
	物聯網:GetStatistics
	物聯網:GetTopic規則
	物聯網:GetTopicRuleDestination
	物聯網：獲取 V2 LoggingOptions
	物聯網:ListActive違規
	物聯網:ListAttached政策
	物聯網:ListAudit發現
	物聯網:ListAuditMitigationActions執行
	物聯網:ListAuditMitigationActions任務
	物聯網:ListAudit抑制
	物聯網:ListAudit任務

服務前綴	動作
	物聯網:ListAuthorizers
	物聯網:ListBilling群組
	iot:ListCACertificates
	物聯網:ListCertificate供應商
	物聯網:ListCertificates
	物聯網:ListCertificatesBYCA
	物聯網:ListCustom度量
	物聯網:ListDetectMitigationActions執行
	物聯網:ListDetectMitigationActions任務
	物聯網:ListDimensions
	物聯網：ListDomain配置
	物聯網:ListFleet度量
	物聯網:ListIndices
	物聯網:ListJobExecutionsForJob
	物聯網:ListJobExecutionsFor事
	物聯網:ListJobs
	物聯網:ListJob範本
	物聯網:ListManagedJobTemplates
	物聯網:ListMetric價值觀
	物聯網:ListMitigation行動
	iot:ListOTAUpdates

服務前綴	動作
	物聯網:ListOutgoing證書
	物聯網:ListPackages
	物聯網：ListPackage版本
	物聯網:ListPolicies
	物聯網:ListPolicy校長
	物聯網：ListPolicy版本
	物聯網:ListPrincipal政策
	物聯網:ListPrincipal事
	物聯網:ListProvisioning範本
	物聯網:ListProvisioningTemplateVersions
	物聯網:ListRelatedResourcesForAuditFinding
	物聯網:ListRole別名
	物聯網:ListScheduled審計
	物聯網:ListSecurity設定檔
	物聯網:ListSecurityProfilesFor目標
	物聯網:ListStreams
	物聯網:ListTargetsForPolicy
	物聯網：ListTargetsForSecurity設定檔
	物聯網:ListThing群組
	物聯網:ListThingGroupsFor事
	物聯網:ListThing校長

服務前綴	動作
	物聯網:ListThingRegistrationTask報告
	物聯網:ListThingRegistrationTasks
	物聯網:ListThings
	物聯網:ListThingsInBilling集團
	物聯網:ListThingsInThing集團
	物聯網:ListThing類型
	物聯網:ListTopicRuleDestinations
	物聯網:ListTopic規則
	物聯網:清單 V2 LoggingLevels
	物聯網:ListViolation活動
	物聯網:PutVerificationStateOn違規
	iot:RegisterCACertificate
	物聯網:RegisterCertificate
	物聯網:RegisterCertificate沒有
	物聯網:RegisterThing
	物聯網:RejectCertificate轉移
	物聯網:RemoveThingFromBilling集團
	物聯網:RemoveThingFromThing集團
	物聯網:ReplaceTopic規則
	物聯網:SearchIndex
	物聯網:SetDefault授權者

服務前綴	動作
	物聯網:SetDefaultPolicyVersion
	物聯網:SetLogging選項
	物聯網:設定 V2 LoggingLevel
	物聯網:設定 V2 LoggingOptions
	物聯網:StartAuditMitigationActions任務
	物聯網:StartDetectMitigationActions任務
	物聯網:StartOnDemandAudit任務
	物聯網:StartThingRegistrationTask
	物聯網:StopThingRegistrationTask
	物聯網:TestAuthorization
	物聯網:TestInvoke授權者
	物聯網:TransferCertificate
	物聯網:UpdateAccountAuditConfiguration
	物聯網:UpdateAudit抑制
	物聯網:UpdateAuthorizer
	物聯網:UpdateBilling集團
	iot:UpdateCACertificate
	物聯網:UpdateCertificate
	物聯網:UpdateCertificate供應商
	物聯網:UpdateCustom公制
	物聯網:UpdateDimension

服務前綴	動作
	物聯網:UpdateDomain配置
	物聯網:UpdateDynamicThingGroup
	物聯網：UpdateEvent配置
	物聯網:UpdateFleet公制
	物聯網:UpdateIndexing配置
	物聯網:UpdateJob
	物聯網:UpdateMitigation行動
	物聯網:UpdatePackage
	物聯網:UpdatePackage配置
	物聯網:UpdatePackage版本
	物聯網:UpdateProvisioning模板
	物聯網:UpdateRole別名
	物聯網:UpdateScheduled審計
	物聯網：UpdateSecurity設定檔
	物聯網:UpdateStream
	物聯網:UpdateThing
	物聯網:UpdateThing集團
	物聯網:UpdateThingGroupsFor事
	物聯網:UpdateTopicRuleDestination
	物聯網:ValidateSecurityProfileBehaviors

服務前綴	動作
iotanalytics	物聯網分析：再處理 CancelPipeline
	物聯網分析：CreateChannel
	物聯網分析：CreateDataset
	物聯網分析：內容 CreateDataset
	物聯網分析：CreateDatastore
	物聯網分析：CreatePipeline
	物聯網分析：DeleteChannel
	物聯網分析：DeleteDataset
	物聯網分析：內容 DeleteDataset
	物聯網分析：DeleteDatastore
	物聯網分析：DeletePipeline
	物聯網分析：DescribeChannel
	物聯網分析：DescribeDataset
	物聯網分析：DescribeDatastore
	物聯網分析：選項 DescribeLogging
	物聯網分析：DescribePipeline
	物聯網分析：內容 GetDataset
	物聯網分析：ListChannels
	物聯網分析：內容 ListDataset
	物聯網分析：ListDatasets
	物聯網分析：ListDatastores

服務前綴	動作
	物聯網分析：ListPipelines
	物聯網分析：選項 PutLogging
	物聯網分析：活動 RunPipeline
	物聯網分析：資料 SampleChannel
	物聯網分析：再處理 StartPipeline
	物聯網分析：UpdateChannel
	物聯網分析：UpdateDataset
	物聯網分析：UpdateDatastore
	物聯網分析：UpdatePipeline
iotdeviceadvisor	物聯網設備顧問：定義 CreateSuite
	物聯網設備顧問：定義 DeleteSuite
	物聯網設備顧問：GetEndpoint
	物聯網設備顧問：定義 GetSuite
	物聯網設備顧問：執行 GetSuite
	物聯網設備顧問：GetSuiteRunReport
	物聯網設備顧問：定義 ListSuite
	物聯網設備顧問：執行 ListSuite
	物聯網設備顧問：執行 StartSuite
	物聯網設備顧問：執行 StopSuite
	物聯網設備顧問：定義 UpdateSuite

服務前綴	動作
iotevents	注意事項:警報 BatchAcknowledge 物質:探測器 BatchDelete 注意事項:警報 BatchDisable 注意事項:警報 BatchEnable 注意事項:警報 BatchReset 注意事項:警報 BatchSnooze 物質:探測器 BatchUpdate 替代品:模型 CreateAlarm 替代品:模型 CreateDetector 碘酸鹽：CreateInput 替代品:模型 DeleteAlarm 替代品:模型 DeleteDetector 碘酸鹽：DeleteInput 碘酸鹽：DescribeAlarm 替代品:模型 DescribeAlarm 碘酸鹽：DescribeDetector 替代品:模型 DescribeDetector 碘酸鹽：DescribeDetectorModelAnalysis 碘酸鹽：DescribeInput 替代品:選項 DescribeLogging 注射劑：結果 GetDetector ModelAnalysis

服務前綴	動作
	替代品:模型 ListAlarm 碘酸鹽：ListAlarmModelVersions 碘酸鹽：ListAlarms 替代品:模型 ListDetector 碘酸鹽：ListDetectorModelVersions 碘酸鹽：ListDetectors 含量:路由 ListInput 碘酸鹽：ListInputs 替代品:選項 PutLogging 碘酸鹽：StartDetectorModelAnalysis 替代品:模型 UpdateAlarm 替代品:模型 UpdateDetector 碘酸鹽：UpdateInput
iotfleethub	微波器:CreateApplication 微波器:DeleteApplication 微波器:DescribeApplication 微波器:ListApplications 微波器:UpdateApplication

服務前綴	動作
iotsitewise	異位方向：AssociateAssets 異位方向：AssociateTimeSeriesToAssetProperty 異位方向：BatchAssociateProjectAssets 異位方向：BatchDisassociateProjectAssets 異位：價值 BatchGet AssetProperty 異位方向：BatchGetAssetPropertyValueHistory 異位：價值 BatchPut AssetProperty 異位：政策 CreateAccess 異位方向：CreateAsset 異位：模型 CreateAsset 異位：模型 CreateAsset ModelComposite 異位方向：CreateBulkImportJob 異位方向：CreateDashboard 異位方向：CreateGateway 異位方向：CreatePortal 異位方向：CreateProject 異位：政策 DeleteAccess 異位方向：DeleteAsset 異位：模型 DeleteAsset 異位：模型 DeleteAsset ModelComposite 異位方向：DeleteDashboard

服務前綴	動作
	異位方向：DeleteGateway 異位方向：DeletePortal 異位方向：DeleteProject 異位：系列 DeleteTime 異位：政策 DescribeAccess 異位方向：DescribeAsset 異位方向：DescribeAssetCompositeModel 異位：模型 DescribeAsset 異位：模型 DescribeAsset ModelComposite 異位：屬性 DescribeAsset 異位方向：DescribeBulkImportJob 異位方向：DescribeDashboard 異位方向：DescribeDefaultEncryptionConfiguration 異位方向：DescribeGateway 異位方向：DescribeGatewayCapabilityConfiguration 異位：選項 DescribeLogging 異位方向：DescribePortal 異位方向：DescribeProject 物聯網：配置 DescribeStorage 異位：系列 DescribeTime 異位方向：DisassociateAssets

服務前綴	動作
	異位方向：DisassociateTimeSeriesFromAssetProperty 異位方向：ExecuteAction 異位方向：ExecuteQuery 物聯網：政策 ListAccess 異位方向：ListActions 異位：模型 ListAsset ModelComposite 異位方向：ListAssetModelProperties 異位：模型 ListAsset 異位：屬性 ListAsset 異位：關係 ListAsset 異位方向：ListAssets 物聯網位置：資產 ListAssociated 異位方向：ListBulkImportJobs 異位：關係 ListComposition 異位方向：ListDashboards 異位方向：ListGateways 異位方向：ListPortals 物聯網位置：資產 ListProject 異位方向：ListProjects 異位：系列 ListTime 異位方向：PutDefaultEncryptionConfiguration

服務前綴	動作
	異位：選項 PutLogging 物聯網：配置 PutStorage 異位：政策 UpdateAccess 異位方向：UpdateAsset 異位：模型 UpdateAsset 異位：模型 UpdateAsset ModelComposite 異位：屬性 UpdateAsset 異位方向：UpdateDashboard 異位方向：UpdateGateway 異位方向：UpdateGatewayCapabilityConfiguration 異位方向：UpdatePortal 異位：UpdateProject

服務前綴	動作
iottwinmaker	物聯網製造商：CancelMetadataTransferJob
	物聯網製造商：類型 CreateComponent
	物聯網製造商：CreateEntity
	物聯網製造商：CreateMetadataTransferJob
	物聯網製造商：CreateScene
	物聯網創造者：Job CreateSync
	物聯網製造商：CreateWorkspace
	物聯網製造商：類型 DeleteComponent
	物聯網製造商：DeleteEntity
	物聯網製造商：DeleteScene
	物聯網創造者：Job DeleteSync
	物聯網製造商：DeleteWorkspace
	物聯網製造商：ExecuteQuery
	物聯網製造商：GetMetadataTransferJob
	物聯網製造者：計劃 GetPricing
	物聯網製造商：GetScene
	物聯網創造者：Job GetSync
	物聯網製造商：ListComponents
	物聯網製造商：類型 ListComponent
	物聯網製造商：ListEntities
	物聯網製造商：ListMetadataTransferJobs

服務前綴	動作
	物聯網製造商：ListProperties
	物聯網製造商：ListScenes
	物聯網製造商：工作 ListSync
	物聯網製造者：資源 ListSync
	物聯網製造商：ListWorkspaces
	物聯網製造商：類型 UpdateComponent
	物聯網製造商：UpdateEntity
	物聯網製造者：計劃 UpdatePricing
	物聯網製造商：UpdateScene
	物聯網製造商：UpdateWorkspace

服務前綴	動作
iotwireless	物聯無線:AssociateAwsAccountWithPartnerAccount 物聯無線:AssociateMulticastGroupWithFuotaTask 物聯無線:AssociateWirelessDeviceWithFuotaTask 物聯無線:AssociateWirelessDeviceWithMulticastGroup 物聯無線：事 AssociateWireless DeviceWith 物聯網無線：證書 AssociateWireless GatewayWith 物聯無線：事 AssociateWireless GatewayWith 物聯無線:CancelMulticastGroupSession 物聯無線:CreateDestination 物聯無線：設定檔 CreateDevice 物聯無線：任務 CreateFuota 物聯無線:群組 CreateMulticast 物聯無線:CreateNetworkAnalyzerConfiguration 物聯無線：設定檔 CreateService 物聯網無線：設備 CreateWireless 物聯網無線：閘道 CreateWireless 物聯無線:CreateWirelessGatewayTask 物聯無線：定義 CreateWireless GatewayTask 物聯無線:DeleteDestination 物聯無線：設定檔 DeleteDevice 物聯無線：任務 DeleteFuota

服務前綴	動作
	物聯無線:群組 DeleteMulticast 物聯無線:DeleteNetworkAnalyzerConfiguration 物聯網無線：訊息 DeleteQueued 物聯無線：設定檔 DeleteService 物聯網無線：設備 DeleteWireless 物聯無線：任務 DeleteWireless DeviceImport 物聯網無線：閘道 DeleteWireless 物聯無線:DeleteWirelessGatewayTask 物聯無線：定義 DeleteWireless GatewayTask 物聯網無線：設備 DeregisterWireless 物聯無線:DisassociateAwsAccountFromPartnerAccount 物聯無線:DisassociateMulticastGroupFromFuotaTask 物聯無線:DisassociateWirelessDeviceFromFuotaTask 物聯無線:DisassociateWirelessDeviceFromMulticastGroup 物聯無線：事 DisassociateWireless DeviceFrom 物聯網無線：證書 DisassociateWireless GatewayFrom 物聯無線：事 DisassociateWireless GatewayFrom 物聯無線:GetDestination 物聯無線：設定檔 GetDevice 物聯無線:GetEventConfigurationByResourceTypes 物聯無線：任務 GetFuota

服務前綴	動作
	物聯無線:GetLogLevelsByResourceTypes 物聯無線：配置 GetMetric 物聯無線:GetMetrics 物聯無線:群組 GetMulticast 物聯無線:GetMulticastGroupSession 物聯無線:GetNetworkAnalyzerConfiguration 無線網路：帳戶 GetPartner 物聯無線:GetPosition 物聯無線：配置 GetPosition 物聯無線：估計 GetPosition 物聯無線:GetResourceEventConfiguration 物聯無線:GetResourceLogLevel 物聯無線：位置 GetResource 物聯網無線：端點 GetService 物聯無線：設定檔 GetService 物聯網無線：設備 GetWireless 物聯無線：任務 GetWireless DeviceImport 物聯無線:GetWirelessDeviceStatistics 物聯網無線：閘道 GetWireless 物聯無線:GetWirelessGatewayCertificate 物聯無線：資訊 GetWireless GatewayFirmware

服務前綴	動作
	物聯無線:GetWirelessGatewayStatistics 物聯無線:GetWirelessGatewayTask 物聯無線：定義 GetWireless GatewayTask 物聯無線:ListDestinations 物聯無線：設定檔 ListDevice 物聯無線：任務 ListDevices ForWireless DeviceImport 物聯無線：配置 ListEvent 物聯網無線：任務 ListFuota 物聯無線：群組 ListMulticast 物聯無線:ListMulticastGroupsByFuotaTask 物聯無線:ListNetworkAnalyzerConfigurations 無線網路：帳戶 ListPartner 物聯無線：配置 ListPosition 物聯網無線：訊息 ListQueued 物聯無線：設定檔 ListService 物聯網無線：任務 ListWireless DeviceImport 物聯無線:裝置 ListWireless 物聯網無線：閘道 ListWireless 物聯網無線：定義 ListWireless GatewayTask 物聯無線：配置 PutPosition 物聯無線:PutResourceLogLevel

服務前綴	動作
	物聯無線：水平 ResetAll ResourceLog 物聯無線:ResetResourceLogLevel 物聯無線:群組 SendData ToMulticast 物聯網無線：設備 SendData ToWireless 物聯無線:StartBulkAssociateWirelessDeviceWithMulticastGroup 物聯無線:StartBulkDisassociateWirelessDeviceFromMulticastGroup 物聯無線：任務 StartFuota 物聯無線:StartMulticastGroupSession 物聯無線:StartNetworkAnalyzerStream 物聯無線:StartSingleWirelessDeviceImportTask 物聯無線：任務 StartWireless DeviceImport 物聯網無線：設備 TestWireless 物聯無線:UpdateDestination 物聯無線:UpdateEventConfigurationByResourceTypes 物聯無線：任務 UpdateFuota 物聯無線:UpdateLogLevelsByResourceTypes 物聯無線：配置 UpdateMetric 物聯無線:群組 UpdateMulticast 物聯無線:UpdateNetworkAnalyzerConfiguration 無線網路：帳戶 UpdatePartner 物聯無線:UpdatePosition

服務前綴	動作
	物聯無線:UpdateResourceEventConfiguration 物聯無線：位置 UpdateResource 物聯網無線：設備 UpdateWireless 物聯無線：任務 UpdateWireless DeviceImport 物聯網無線：閘道 UpdateWireless

服務前綴	動作
ivs	ivs: BatchGet 頻道 智能：BatchGetStreamKey ivs: 撤銷 BatchStart ViewerSession 智能：CreateChannel ivs: CreateEncoder 組態 ivs：令牌 CreateParticipant 智能：CreatePlaybackRestrictionPolicy ivs: CreateRecording 組態 ivs: CreateStorage 組態 ivs: 鑰匙 CreateStream 智能：DeleteChannel ivs: DeleteEncoder 組態 智能：DeletePlaybackKeyPair 智能：DeletePlaybackRestrictionPolicy ivs: DeleteRecording 組態 ivs: DeleteStorage 組態 ivs: 鑰匙 DeleteStream 智能：DisconnectParticipant 智能：GetChannel 智能：GetComposition ivs: GetEncoder 組態

服務前綴	動作
	智能：GetParticipant
	智能：GetPlaybackKeyPair
	智能：GetPlaybackRestrictionPolicy
	ivs: GetRecording 組態
	ivs: GetStorage 組態
	智能：GetStream
	ivs: 鑰匙 GetStream
	ivs: GetStream 工作階段
	智能：ImportPlaybackKeyPair
	智能：ListChannels
	智能：ListCompositions
	ivs: ListEncoder 組態
	ivs: ListParticipant 活動
	智能：ListParticipants
	智能：ListPlaybackKeyPairs
	智能：ListPlaybackRestrictionPolicies
	ivs: ListRecording 組態
	ivs: ListStorage 組態
	ivs: ListStream 按鍵
	智能：ListStreams
	ivs: ListStream 工作階段

服務前綴	動作
	智能：PutMetadata 智能：StartComposition 智能：StartViewerSessionRevocation 智能：StopComposition 智能：StopStream 智能：UpdateChannel 智能：UpdatePlaybackRestrictionPolicy
ivschat	伊夫斯聊天:令牌 CreateChat 互聯:配置 CreateLogging 伊夫斯查特：CreateRoom 互聯:配置 DeleteLogging 伊夫斯查特：DeleteMessage 伊夫斯查特：DeleteRoom 伊夫斯查特：DisconnectUser 互聯:配置 GetLogging 伊夫斯查特：GetRoom 查看:配置 ListLogging 伊夫斯查特：ListRooms 伊夫斯查特：SendEvent 互聯:配置 UpdateLogging 伊夫斯查特：UpdateRoom

服務前綴	動作
kafka	卡夫卡:BatchAssociateScramSecret
	卡夫卡:BatchDisassociateScramSecret
	卡夫卡:CreateCluster
	卡夫卡:V2 CreateCluster
	卡夫卡:CreateConfiguration
	卡夫卡:CreateReplicator
	卡夫卡:連接 CreateVpc
	卡夫卡:DeleteCluster
	卡夫卡:政策 DeleteCluster
	卡夫卡:DeleteConfiguration
	卡夫卡:DeleteReplicator
	卡夫卡:連接 DeleteVpc
	卡夫卡:DescribeCluster
	卡夫卡：操作 DescribeCluster
	卡夫卡:操作 V2 DescribeCluster
	卡夫卡:V2 DescribeCluster
	卡夫卡:DescribeConfiguration
	卡夫卡:修訂 DescribeConfiguration
	卡夫卡:連接 DescribeVpc
	卡夫卡:經紀人 GetBootstrap
	卡夫卡:政策 GetCluster

服務前綴	動作
	卡夫卡:GetCompatibleKafkaVersions
	卡夫卡:ListClientVpcConnections
	卡夫卡:操作 ListCluster
	卡夫卡：操作 SV2 ListCluster
	卡夫卡:ListClusters
	卡夫卡:V2 ListClusters
	卡夫卡:修訂 ListConfiguration
	卡夫卡:ListConfigurations
	卡夫卡:版本 ListKafka
	卡夫卡:ListNodes
	卡夫卡:ListReplicators
	卡夫卡：秘密 ListScram
	卡夫卡:連接 ListVpc
	卡夫卡:政策 PutCluster
	卡夫卡:RebootBroker
	卡夫卡:RejectClientVpcConnection
	卡夫卡：計數 UpdateBroker
	卡夫卡:存儲 UpdateBroker
	卡夫卡：類型 UpdateBroker
	卡夫卡:配置 UpdateCluster
	卡夫卡:UpdateClusterKafkaVersion

服務前綴	動作
	卡夫卡:UpdateConfiguration 卡夫卡:UpdateConnectivity 卡夫卡:UpdateMonitoring 卡夫卡:信息 UpdateReplication 卡夫卡:UpdateSecurity 卡夫卡:UpdateStorage
kafkaconnect	卡夫卡內特:CreateConnector 插件：插件 CreateCustom 功能：配置 CreateWorker 卡夫卡內特:DeleteConnector 插件：插件 DeleteCustom 功能：配置 DeleteWorker 卡夫卡內特:DescribeConnector 插件：插件 DescribeCustom 功能：配置 DescribeWorker 卡夫卡內特:ListConnectors 功能：插件 ListCustom 功能：配置 ListWorker 卡夫卡內特:UpdateConnector

服務前綴	動作
kendra	肯德拉：AssociateEntitiesToExperience
	肯德拉：AssociatePersonasToEntities
	肯德拉:BatchDelete文件
	肯德拉：BatchDeleteFeaturedResults設置
	肯德拉：BatchGetDocumentStatus
	肯德拉:BatchPut文件
	肯德拉：ClearQuery建議
	肯德拉：CreateAccessControlConfiguration
	肯德拉：CreateData來源
	肯德拉：CreateExperience
	肯德拉：CreateFaq
	肯德拉：CreateFeaturedResultsSet
	肯德拉：CreateIndex
	肯德拉：列CreateQuerySuggestionsBlock表
	肯德拉：CreateThesaurus
	肯德拉：DeleteData來源
	肯德拉：DeleteExperience
	肯德拉：DeleteFaq
	肯德拉：DeleteIndex
	肯德拉：映射 DeletePrincipal
	肯德拉：列DeleteQuerySuggestionsBlock表

服務前綴	動作
	肯德拉：DeleteThesaurus
	肯德拉：DescribeAccessControlConfiguration
	肯德拉：DescribeData來源
	肯德拉：DescribeExperience
	肯德拉：DescribeFaq
	肯德拉：DescribeFeaturedResultsSet
	肯德拉：DescribeIndex
	肯德拉：映射 DescribePrincipal
	肯德拉：列DescribeQuerySuggestionsBlock表
	肯德拉：DescribeQuerySuggestionsConfig
	肯德拉：DescribeThesaurus
	肯德拉：DisassociateEntitiesFromExperience
	肯德拉：DisassociatePersonasFromEntities
	肯德拉：GetQuery建議
	肯德拉：GetSnapshots
	肯德拉：ListAccessControlConfigurations
	肯德拉：ListData來源
	肯德拉：ListDataSourceSync工作
	肯德拉：人物角色 ListEntity
	肯德拉：ListExperience實體
	肯德拉：ListExperiences

服務前綴	動作
	肯德拉：ListFaqs
	肯德拉：ListFeaturedResultsSets
	肯德拉：ListGroupsOlderThanOrderingId
	肯德拉：ListIndices
	肯德拉：列ListQuerySuggestionsBlock表
	肯德拉：ListThesauri
	肯德拉：映射 PutPrincipal
	kendra:Query
	kendra:Retrieve
	肯德拉：Job StartData SourceSync
	肯德拉：Job StopData SourceSync
	肯德拉：SubmitFeedback
	肯德拉：UpdateData來源
	肯德拉：UpdateExperience
	肯德拉：UpdateFeaturedResultsSet
	肯德拉：UpdateIndex
	肯德拉：列UpdateQuerySuggestionsBlock表
	肯德拉：UpdateQuerySuggestionsConfig
	肯德拉：UpdateThesaurus

服務前綴	動作
kinesis	室壁運動:CreateStream 室壁運動:DecreaseStreamRetentionPeriod 室壁運動:DeleteStream 室壁運動:DeregisterStream消費者 室壁運動:DescribeLimits 室壁運動:DescribeStream 室壁運動:DescribeStream消費者 室壁運動:總DescribeStream結 室壁運動:監測 DisableEnhanced 室壁運動:監測 EnableEnhanced 室壁運動:IncreaseStreamRetentionPeriod 室壁運動:ListShards 室壁運動:ListStream消費者 室壁運動:ListStreams 室壁運動:MergeShards 室壁運動:RegisterStream消費者 室壁運動:SplitShard 室壁運動:StartStream加密 室壁運動:StopStream加密 室壁運動:UpdateShard計數 室壁運動:UpdateStream模式

服務前綴	動作
kinesisanalytics	運動分析：AddApplicationCloudWatchLoggingOption 動力分析:輸入 AddApplication 動力分析:配置 AddApplication InputProcessing 動力分析:輸出 AddApplication 動力分析：來源 AddApplication ReferenceData 運動分析：AddApplicationVpcConfiguration 運動分析：CreateApplication 運動分析：CreateApplicationPresignedUrl 動力分析:快照 CreateApplication 運動分析：DeleteApplication 運動分析：DeleteApplicationCloudWatchLoggingOption 動力分析:配置 DeleteApplication InputProcessing 動力分析:輸出 DeleteApplication 動力分析：來源 DeleteApplication ReferenceData 動力分析:快照 DeleteApplication 運動分析：DeleteApplicationVpcConfiguration 運動分析：DescribeApplication 動力分析:快照 DescribeApplication 運動分析:版本 DescribeApplication 動力分析:綱要 DiscoverInput 運動分析：ListApplications

服務前綴	動作
	動力分析：快照 ListApplication
	動力分析:版本 ListApplication
	運動分析：RollbackApplication
	運動分析：StartApplication
	運動分析：StopApplication
	運動分析：UpdateApplication
	運動分析：UpdateApplicationMaintenanceConfiguration

服務前綴	動作
kms	公里:CancelKey刪除 公里：ConnectCustomKeyStore 公里：CreateAlias 公里：CreateCustomKeyStore 公里：CreateGrant 公里：CreateKey kms:解密 公里：DeleteAlias 公里：DeleteCustomKeyStore 公里：DeleteImportedKeyMaterial 公里：DescribeCustomKeyStores 公里：DescribeKey 公里：DisableKey 公里:DisableKey旋轉 公里：DisconnectCustomKeyStore 公里：EnableKey 公里:EnableKey旋轉 kms:Encrypt 公里：GenerateData鑰匙 公里：GenerateDataKeyPair 公里：GenerateDataKeyPairWithoutPlaintext

服務前綴	動作
	公里:GenerateDataKeyWithout明文 公里：GenerateMac 公里：GenerateRandom 公里：GetKey政策 公里：GetKeyRotationStatus 公里：GetParametersForImport 公里：GetPublic鑰匙 公里:ImportKey材料 公里：ListAliases 公里：ListGrants 公里：ListKey政策 公里：ListKeys 公里：ListRetirable贈款 公里：ReplicateKey 公里：RetireGrant 公里：RevokeGrant 公里:ScheduleKey刪除 kms:Sign 公里：UpdateAlias 公里：UpdateCustomKeyStore 公里：UpdateKey說明

服務前綴	動作
	公里:UpdatePrimary區域 kms:Verify 公里：VerifyMac

服務前綴	動作
lambda	拉姆達：AddLayerVersionPermission
	拉姆達：AddLayerVersionPermission
	拉姆達：AddPermission
	拉姆達：AddPermission
	拉姆達：AddPermission
	拉姆達：CreateAlias
	拉姆達：CreateAlias
	拉姆達：CreateCodeSigningConfig
	拉姆達：CreateEventSourceMapping
	拉姆達：CreateEventSourceMapping
	拉姆達：CreateFunction
	拉姆達：CreateFunction
	拉姆達：CreateFunctionUrlConfig
	拉姆達：DeleteAlias
	拉姆達：DeleteAlias
	拉姆達：DeleteCodeSigningConfig
	拉姆達：DeleteEventSourceMapping
	拉姆達：DeleteEventSourceMapping
	拉姆達：DeleteFunction
	拉姆達：DeleteFunction
	拉姆達：DeleteFunctionCodeSigningConfig

服務前綴	動作
	拉姆達：DeleteFunction並發
	拉姆達：DeleteFunction並發
	拉姆達：DeleteFunctionEventInvokeConfig
	拉姆達：DeleteFunctionUrlConfig
	拉姆達：DeleteLayer版本
	拉姆達：DeleteLayer版本
	拉姆達：DeleteProvisionedConcurrencyConfig
	拉姆達：GetAccount設置
	拉姆達：GetAccount設置
	拉姆達：GetAlias
	拉姆達：GetAlias
	拉姆達：GetCodeSigningConfig
	拉姆達：GetEventSourceMapping
	拉姆達：GetEventSourceMapping
	拉姆達：GetFunction
	拉姆達：GetFunction
	拉姆達：GetFunction
	拉姆達：GetFunctionCodeSigningConfig
	拉姆達：GetFunction並發
	拉姆達：GetFunction配置
	拉姆達：GetFunction配置

服務前綴	動作
	拉姆達：GetFunction配置
	拉姆達：GetFunctionEventInvokeConfig
	拉姆達：GetFunctionUrlConfig
	拉姆達：GetLayer版本
	拉姆達：GetLayer版本
	拉姆達：GetLayer版本
	拉姆達：GetLayer版本
	拉姆達：GetLayerVersionPolicy
	拉姆達：GetLayerVersionPolicy
	拉姆達：GetPolicy
	拉姆達：GetPolicy
	拉姆達：GetPolicy
	拉姆達：GetProvisionedConcurrencyConfig
	拉姆達：GetRuntimeManagementConfig
	拉姆達：ListAliases
	拉姆達：ListAliases
	拉姆達：ListCodeSigningConfigs
	拉姆達：ListEventSourceMappings
	拉姆達：ListEventSourceMappings
	拉姆達：ListFunctionEventInvoke配置
	拉姆達：ListFunctions

服務前綴	動作
	拉姆達：ListFunctions
	拉姆達：ListFunctionsByCodeSigningConfig
	拉姆達：ListFunctionUrlConfigs
	拉姆達：ListLayers
	拉姆達：ListLayers
	拉姆達：ListLayer版本
	拉姆達：ListLayer版本
	拉姆達：ListProvisionedConcurrencyConfigs
	拉姆達：ListVersionsByFunction
	拉姆達：ListVersionsByFunction
	拉姆達：PublishLayer版本
	拉姆達：PublishLayer版本
	拉姆達：PublishVersion
	拉姆達：PublishVersion
	拉姆達：PutFunctionCodeSigningConfig
	拉姆達：PutFunction並發
	拉姆達：PutFunction並發
	拉姆達：PutFunctionEventInvokeConfig
	拉姆達：PutProvisionedConcurrencyConfig
	拉姆達：PutRuntimeManagementConfig
	拉姆達：RemoveLayerVersionPermission

服務前綴	動作
	拉姆達：RemoveLayerVersionPermission
	拉姆達：RemovePermission
	拉姆達：RemovePermission
	拉姆達：RemovePermission
	拉姆達：UpdateAlias
	拉姆達：UpdateAlias
	拉姆達：UpdateCodeSigningConfig
	拉姆達：UpdateEventSourceMapping
	拉姆達：UpdateEventSourceMapping
	拉姆達：UpdateFunction代碼
	拉姆達：UpdateFunction代碼
	拉姆達：UpdateFunction代碼
	拉姆達：UpdateFunction配置
	拉姆達：UpdateFunction配置
	拉姆達：UpdateFunction配置
	拉姆達：UpdateFunctionEventInvokeConfig
	拉姆達：UpdateFunctionUrlConfig

服務前綴	動作
lex	萊克斯：BatchCreateCustomVocabulary項目 萊克斯：BatchDeleteCustomVocabulary項目 萊克斯：BatchUpdateCustomVocabulary項目 LEX：BuildBot語言環境 萊克斯：CreateBot別名 萊克斯:CreateBot版本 萊克斯：CreateExport 萊克斯:CreateIntent版本 萊克斯:CreateResource政策 萊克斯：CreateSlotTypeVersion 萊克斯:CreateTestSetDiscrepancy報告 萊克斯：CreateUpload網址 萊克斯：DeleteBot 萊克斯：DeleteBotChannelAssociation 萊克斯：DeleteExport 萊克斯：DeleteImport 萊克斯:DeleteIntent版本 萊克斯:DeleteResource政策 萊克斯：DeleteSlotTypeVersion 萊克斯：DeleteTest設置 萊克斯：DeleteUtterances

服務前綴	動作
	萊克斯：DescribeBot別名 萊克斯：DescribeBot推薦 萊克斯：DescribeBotResourceGeneration 萊克斯:DescribeBot版本 萊克斯：DescribeCustomVocabularyMetadata 萊克斯：DescribeExport 萊克斯：DescribeImport 萊克斯:DescribeResource政策 萊克斯：DescribeTest執行 萊克斯：DescribeTest設置 萊克斯:DescribeTestSetDiscrepancy報告 萊克斯：DescribeTestSetGeneration 萊克斯：GenerateBot元素 萊克斯：GetBot 萊克斯：GetBot別名 別GetBot名 萊克斯：GetBotChannelAssociation 萊克斯：GetBotChannelAssociations 萊克斯：GetBots 萊克斯：GetBot版本 萊克斯：GetBuiltin意圖

服務前綴	動作
	萊克斯：GetBuiltin意圖 萊克斯：GetBuiltinSlotTypes 萊克斯：GetExport 萊克斯：GetImport 萊克斯：GetIntent 萊克斯：GetIntents 萊克斯：GetIntent版本 萊克斯：GetMigration 萊克斯：GetMigrations 萊克斯：GetSlot類型 萊克斯：GetSlot類型 萊克斯：GetSlotTypeVersions 萊克斯：GetTestExecutionArtifacts網址 萊克斯:GetUtterances檢視 別ListBot名 萊克斯:ListBot建議 萊克斯：ListBotResourceGenerations 萊克斯：ListBots 萊克斯：ListBot版本 萊克斯：ListBuiltInIntents 萊克斯：ListBuiltInSlot類型

服務前綴	動作
	萊克斯：ListCustomVocabularyItems 萊克斯：ListExports 萊克斯：ListImports 萊克斯:ListIntent度量 萊克斯：ListIntent路徑 萊克斯：ListRecommended意圖 萊克斯：ListSessionAnalyticsData 萊克斯:ListSession度量 萊克斯：ListTestExecutionResult項目 萊克斯：ListTest執行 萊克斯：ListTest套 萊克斯：PutBot 萊克斯：PutBot別名 萊克斯：PutIntent 萊克斯：PutSlot類型 萊克斯:SearchAssociated成績單 萊克斯：StartBot推薦 萊克斯：StartImport 萊克斯：StartMigration 萊克斯：StartTest執行 萊克斯：StartTestSetGeneration

服務前綴	動作
	萊克斯：StopBot推薦 萊克斯：UpdateBot別名 萊克斯：UpdateBot推薦 萊克斯：UpdateExport 萊克斯:UpdateResource政策
license-manager-linux-subscriptions	授權管理員-Linux 訂閱：設定 GetService 許可證管理器-鏈接訂閱：ListLinuxSubscriptionInstances 授權管理員-Linux 訂閱：訂閱 ListLinux 授權管理員-Linux 訂閱：設定 UpdateService

服務前綴	動作
lightsail	閃帆:IP AllocateStatic 光帆：AttachCertificateToDistribution 光帆：AttachDisk 光帆：AttachInstancesToLoad平衡器 光帆：證書 AttachLoad BalancerTls 閃帆:IP AttachStatic 光帆：CloseInstancePublicPorts 光帆：CopySnapshot 光帆：CreateBucket 光帆：CreateBucketAccessKey 光帆：CreateCertificate 光帆：CreateCloudFormationStack 光帆：CreateContact方法 光帆：CreateContainer服務 光帆：CreateContainerServiceDeployment 光帆:登錄 CreateContainer ServiceRegistry 光帆：CreateDisk 光帆：CreateDiskFromSnapshot 光帆:CreateDisk快照 光帆：CreateDistribution 光帆：CreateDomain

服務前綴	動作
	光帆SessionAccess:創建圖形詳細信息 光帆：CreateInstances 光帆：CreateInstancesFromSnapshot 光帆:CreateInstance快照 光帆：CreateKey對 光帆：CreateLoad平衡器 光帆：證書 CreateLoad BalancerTls 光帆:CreateRelational資料庫 光帆:CreateRelationalDatabaseFrom快照 光帆：CreateRelationalDatabaseSnapshot 光帆：DeleteAlarm 光帆:DeleteAuto快照 光帆：DeleteBucket 光帆：DeleteBucketAccessKey 光帆：DeleteCertificate 光帆：DeleteContact方法 光帆：DeleteContainer影像 光帆：DeleteContainer服務 光帆：DeleteDisk 光帆:DeleteDisk快照 光帆：DeleteDistribution

服務前綴	動作
	光帆：DeleteDomain 光帆：DeleteDomain入口 光帆：DeleteInstance 光帆:DeleteInstance快照 光帆：DeleteKey對 光帆：DeleteKnownHostKeys 光帆：DeleteLoad平衡器 光帆：證書 DeleteLoad BalancerTls 光帆:DeleteRelational資料庫 光帆：DeleteRelationalDatabaseSnapshot 光帆：DetachCertificateFromDistribution 光帆：DetachDisk 光帆：DetachInstancesFromLoad平衡器 閃帆:IP DetachStatic 閃電帆：DisableAdd開啟 光帆：DownloadDefaultKeyPair 閃電帆：EnableAdd開啟 光帆：ExportSnapshot 光帆：GetActive名稱 光帆：GetAlarms 光帆：GetAuto快照

服務前綴	動作
	光帆：GetBlueprints 光帆：GetBucketAccessKeys 光帆：捆綁 GetBucket 光帆：GetBucketMetricData 光帆：GetBuckets 光帆：GetBundles 光帆：GetCertificates 光帆：GetCloudFormationStack記錄 光帆：GetContact方法 光帆:API 元數GetContainer據 光帆：圖GetContainer片 光帆:GetContainer日誌 光帆：GetContainerServiceDeployments 光帆:GetContainerServiceMetric資料 光帆：GetContainerServicePowers 光帆：GetContainer服務 光帆：GetCost估計 光帆：GetDisk 光帆：GetDisks 光帆:GetDisk快照 光帆：GetDisk快照

服務前綴	動作
	光帆：捆綁 GetDistribution 光帆:GetDistributionLatestCache重設 光帆：GetDistributionMetricData 光帆：GetDistributions 光帆：GetDomain 光帆：GetExportSnapshotRecords 光帆：GetInstance 光帆：GetInstanceMetricData 光帆：GetInstancePortStates 光帆：GetInstances 光帆:GetInstance快照 光帆：GetInstance快照 光帆:狀態 GetInstance 光帆：GetKey對 光帆：GetKey對 光帆：GetLoad平衡器 光帆:GetLoadBalancerMetric資料 光帆：GetLoad平衡器 光帆:證書 GetLoad BalancerTls 光帆:政策 GetLoad BalancerTls 光帆：GetOperation

服務前綴	動作
	光帆：GetOperations 光帆：GetOperationsForResource 光帆：GetRegions 光帆:GetRelational資料庫 光帆：GetRelationalDatabaseBlueprints 光帆：GetRelationalDatabaseBundles 光帆：GetRelationalDatabaseEvents 光帆:GetRelationalDatabaseLog活動 光帆:GetRelationalDatabaseLog流 光帆：GetRelationalDatabaseMasterUserPassword 光帆:GetRelationalDatabaseMetric資料 光帆：GetRelationalDatabaseParameters 光帆:GetRelational資料庫 光帆：GetRelationalDatabaseSnapshot 光帆：GetRelationalDatabaseSnapshots 光帆：歷史 GetSetup 閃帆:IP GetStatic 光帆:Ips GetStatic 光帆：ImportKey對 光帆:IsVpc對等 光帆：OpenInstancePublicPorts

服務前綴	動作
	光帆：PeerVpc 光帆：PutAlarm 光帆：PutInstancePublicPorts 光帆：RebootInstance 光帆:RebootRelational資料庫 光帆：RegisterContainer影像 閃帆:IP ReleaseStatic 光帆：ResetDistribution緩存 光帆：SendContactMethodVerification 光帆：SetIpAddressType 光帆：桶 SetResource AccessFor 閃電帆:HTTPS SetupInstance lightsail:StartGUISession 光帆：StartInstance 光帆:StartRelational資料庫 lightsail:StopGUISession 光帆：StopInstance 光帆:StopRelational資料庫 光帆：TestAlarm 光帆：UnpeerVpc 光帆：UpdateBucket

服務前綴	動作
	光帆:捆綁 UpdateBucket 光帆：UpdateContainer服務 光帆：UpdateDistribution 光帆:捆綁 UpdateDistribution 光帆：UpdateDomain入口 光帆：UpdateInstanceMetadataOptions 光帆：UpdateLoadBalancerAttribute 光帆:UpdateRelational資料庫 光帆：UpdateRelationalDatabaseParameters

服務前綴	動作
日誌	記錄檔:AssociateKms索引鍵
	記錄檔:CancelExport工作
	記錄檔:CreateExport工作
	日誌：CreateLogAnomalyDetector
	記錄檔:CreateLog群組
	記錄檔:CreateLog串流
	日誌：DeleteDataProtectionPolicy
	日誌：DeleteDelivery
	記錄檔:DeleteDelivery目的地
	日誌：DeleteDeliveryDestinationPolicy
	記錄檔:DeleteDelivery來源
	日誌：DeleteDestination
	記錄檔:DeleteLog群組
	記錄檔:DeleteLog串流
	記錄檔:DeleteMetric篩選
	記錄檔:DeleteQuery定義
	記錄檔:DeleteResource策略
	記錄檔:DeleteRetention策略
	記錄檔:DeleteSubscription篩選
	記錄檔:DescribeAccount策略
	日誌：DescribeDeliveries

服務前綴	動作
	記錄檔:DescribeDelivery目的地
	記錄檔:DescribeDelivery來源
	日誌：DescribeDestinations
	記錄檔:DescribeExport工作
	記錄檔:DescribeLog群組
	記錄檔:DescribeLog串流
	記錄檔:DescribeMetric篩選器
	日誌：DescribeQueries
	記錄檔:DescribeQuery定義
	記錄檔:DescribeResource策略
	記錄檔:DescribeSubscription篩選器
	記錄檔:DisassociateKms索引鍵
	日誌：GetDataProtectionPolicy
	日誌：GetDelivery
	記錄檔:GetDelivery目的地
	日誌：GetDeliveryDestinationPolicy
	記錄檔:GetDelivery來源
	日誌：GetLogGroupFields
	記錄檔:GetLog記錄
	記錄檔:GetQuery結果
	日誌：ListAnomalies

服務前綴	動作
	日誌：ListLogAnomalyDetectors
	日誌：PutDataProtectionPolicy
	記錄檔:PutDelivery目的地
	日誌：PutDeliveryDestinationPolicy
	記錄檔:PutDelivery來源
	日誌：PutDestination
	記錄檔:PutDestination策略
	記錄檔:PutMetric篩選
	記錄檔:PutQuery定義
	記錄檔:PutResource策略
	記錄檔:PutRetention策略
	記錄檔:PutSubscription篩選
	日誌：StartLive尾巴
	日誌：StartQuery
	日誌：StopQuery
	記錄檔:TestMetric篩選

服務前綴	動作
lookoutequipment	查找設備：CreateDataset 查找設備：排程器 CreateInference 查找設備：CreateLabel 尋找設備：集團 CreateLabel 查找設備：CreateModel 查找設備：DeleteDataset 查找設備：排程器 DeleteInference 查找設備：DeleteLabel 尋找設備：集團 DeleteLabel 查找設備：DeleteModel 查找設備:政策 DeleteResource 查找設備：排程器 DeleteRetraining 查找設備：DescribeDataIngestionJob 查找設備：DescribeDataset 查找設備：排程器 DescribeInference lookoutequipment:Describelabel 尋找設備：集團 DescribeLabel 查找設備：DescribeModel 查找設備：版本 DescribeModel 查找設備:政策 DescribeResource 查找設備：排程器 DescribeRetraining

服務前綴	動作
	查找設備：ImportDataset 查找設備：版本 ImportModel 查找設備：ListDataIngestionJobs 查找設備：ListDatasets 查找設備：活動 ListInference 查找設備：執行 ListInference 查找設備：排程器 ListInference 查找設備：群組 ListLabel 查找設備：ListLabels 查找設備：ListModels 查找設備：版本 ListModel 查找設備：排程器 ListRetraining 查找設備：統計 ListSensor 查找設備:政策 PutResource 查找設備：StartDataIngestionJob 查找設備：排程器 StartInference 查找設備：排程器 StartRetraining 查找設備：排程器 StopInference 查找設備：排程器 StopRetraining 查找設備：UpdateActiveModelVersion 查找設備：排程器 UpdateInference

服務前綴	動作
	尋找設備：集團 UpdateLabel 查找設備：UpdateModel 查找設備：排程器 UpdateRetraining

服務前綴	動作
lookoutmetrics	查找指標：檢測器 ActivateAnomaly 查看指標：BackTestAnomalyDetector 查看指標：CreateAlert 查找指標：檢測器 CreateAnomaly 查看指標:設定 CreateMetric 查找指標：檢測器 DeactivateAnomaly 查看指標：DeleteAlert 查找指標：檢測器 DeleteAnomaly 查看指標：DescribeAlert 查看指標：DescribeAnomalyDetectionExecutions 查找指標：檢測器 DescribeAnomaly 查看指標:設定 DescribeMetric 查看指標：DetectMetricSetConfig 查看指標:群組 GetAnomaly 查看指標：GetDataQualityMetrics 查看指標：GetFeedback 查看指標:資料 GetSample 查看指標：ListAlerts 查找指標：探測器 ListAnomaly 查看指標:測量結果 ListAnomaly GroupRelated 查看指標：ListAnomalyGroupSummaries

服務前綴	動作
	查找指標：系列 ListAnomaly GroupTime 查看指標：集 ListMetric 查看指標：PutFeedback 查看指標：UpdateAlert 查找指標：檢測器 UpdateAnomaly 查看指標:設定 UpdateMetric

服務前綴	動作
lookoutvision	展望視野：CreateDataset 展望視野：CreateModel 展望視野：CreateProject 展望視野：DeleteDataset 展望視野：DeleteModel 展望視野：DeleteProject 展望視野：DescribeDataset 展望視野：DescribeModel 展望視野：DescribeModelPackagingJob 展望視野：DescribeProject 展望視野：DetectAnomalies 查看視野：條目 ListDataset 展望視野：ListModelPackagingJobs 展望視野：ListModels 展望視野：ListProjects 展望視野：StartModel 展望視野：StartModelPackagingJob 展望視野：StopModel 查看視野：條目 UpdateDataset

服務前綴	動作
m2	平方米：CancelBatchJobExecution
	平方米：CreateApplication
	平方米：CreateDataSetImport任務
	平方米：CreateDeployment
	平方米：CreateEnvironment
	平方米：DeleteApplication
	平方米：DeleteApplicationFromEnvironment
	平方米：DeleteEnvironment
	平方米：GetApplication
	平方米：GetApplication版本
	平方米：GetBatchJobExecution
	平方米：GetDataSetDetails
	平方米：GetDataSetImport任務
	平方米：GetDeployment
	平方米：GetEnvironment
	平方米：GetSignedBluinsightsUrl
	平方米：ListApplications
	平方米：ListApplication版本
	平方米：ListBatchJobDefinitions
	平方米：ListBatchJobExecutions
	平方米：ListBatchJobRestart點

服務前綴	動作
	m2：ListDataSetImport歷史 平方米：ListData套 平方米：ListDeployments 平方米：ListEngine版本 平方米：ListEnvironments 平方米：StartApplication 平方米：StartBatchJob 平方米：StopApplication 平方米：UpdateApplication 平方米：UpdateEnvironment

服務前綴	動作
managedblockchain	管理區塊鏈：CreateAccessor 管理區塊鏈：CreateMember 管理區塊鏈：CreateNetwork 管理區塊鏈：CreateNode 管理區塊鏈：CreateProposal 管理區塊鏈：DeleteAccessor 管理區塊鏈：DeleteMember 管理區塊鏈：DeleteNode 管理區塊鏈：GetAccessor 管理區塊鏈：GetMember 管理區塊鏈：GetNetwork 管理區塊鏈：GetNode 管理區塊鏈：GetProposal 管理區塊鏈：InvokeRpcPolygonMainnet 管理區塊鏈：測試網 InvokeRpc PolygonMumbai 管理區塊鏈：ListAccessors 管理區塊鏈：ListInvitations 管理區塊鏈：ListMembers 管理區塊鏈：ListNetworks 管理區塊鏈：ListNodes 管理區塊鏈：ListProposals

服務前綴	動作
	管理區塊鏈：投票 ListProposal 管理區塊鏈：RejectInvitation 管理區塊鏈：UpdateMember 管理區塊鏈：UpdateNode 管理區塊鏈：提案 VoteOn

服務前綴	動作
mediaconnect	媒體輸出：輸出 AddBridge 媒體：來源 AddBridge 媒體內容：AddFlowMediaStreams 媒體輸出：輸出 AddFlow 媒體：來源 AddFlow 媒體內容：AddFlowVpcInterfaces 媒體內容：CreateBridge 媒體內容：CreateFlow 媒體內容：CreateGateway 媒體內容：DeleteBridge 媒體內容：DeleteFlow 媒體內容：DeleteGateway 媒體連接：實例 DeregisterGateway 媒體內容：DescribeBridge 媒體內容：DescribeFlow 媒體內容：DescribeFlowSourceMetadata 媒體內容：DescribeGateway 媒體連接：實例 DescribeGateway 媒體內容：DescribeOffering 媒體內容：DescribeReservation 媒體內容：權利 GrantFlow

服務前綴	動作
	媒體內容：ListBridges
	媒體內容：ListEntitlements
	媒體內容：ListFlows
	媒體連接：實例 ListGateway
	媒體內容：ListGateways
	媒體內容：ListOfferings
	媒體內容：ListReservations
	媒體內容：PurchaseOffering
	媒體輸出：輸出 RemoveBridge
	媒體內容：來源 RemoveBridge
	媒體內容：RemoveFlowMediaStream
	媒體輸出：輸出 RemoveFlow
	媒體內容：來源 RemoveFlow
	媒體內容：RemoveFlowVpcInterface
	媒體信息:權利 RevokeFlow
	媒體內容：StartFlow
	媒體內容：StopFlow
	媒體內容：UpdateBridge
	媒體輸出：輸出 UpdateBridge
	媒體內容：來源 UpdateBridge
	媒體中心:狀態 UpdateBridge

服務前綴	動作
	媒體內容：UpdateFlow 媒體信息:權利 UpdateFlow 媒體內容：UpdateFlowMediaStream 媒體輸出：輸出 UpdateFlow 媒體內容：來源 UpdateFlow 媒體連接：實例 UpdateGateway

服務前綴	動作
mediaconvert	媒介：AssociateCertificate 媒介：CancelJob 媒介：CreateJob 媒體連接:模板 CreateJob 媒介：CreatePreset 媒介：CreateQueue 媒體連接:模板 DeleteJob 媒介：DeletePolicy 媒介：DeletePreset 媒介：DeleteQueue 媒介：DescribeEndpoints 媒介：DisassociateCertificate 媒介：GetJob 媒體連接:模板 GetJob 媒介：GetPolicy 媒介：GetPreset 媒介：GetQueue 媒介：ListJobs 媒體移動:模板 ListJob 媒介：ListPresets 媒介：ListQueues

服務前綴	動作
	媒介：PutPolicy 媒體連接:模板 UpdateJob 媒介：UpdatePreset 媒介：UpdateQueue

服務前綴	動作
medialive	媒體活著：AcceptInputDeviceTransfer
	媒體：BatchDelete
	媒體：BatchStart
	媒體：BatchStop
	媒體活躍:排BatchUpdate程
	媒體活著：CancelInputDeviceTransfer
	媒體：ClaimDevice
	媒體：CreateChannel
	媒體:CreateCloudWatchAlarm模板
	媒體活著：CreateCloudWatchAlarmTemplateGroup
	媒體:CreateEventBridgeRule模板
	媒體活著：CreateEventBridgeRuleTemplateGroup
	媒體：CreateInput
	媒體活著：CreateInputSecurityGroup
	媒體：CreateMultiplex
	媒體活著:CreateMultiplex程序
	媒體活著：輸CreatePartner入
	媒體活著:CreateSignal地圖
	媒體：DeleteChannel
	媒體:DeleteCloudWatchAlarm模板
	媒體活著：DeleteCloudWatchAlarmTemplateGroup

服務前綴	動作
	媒體:DeleteEventBridgeRule模板
	媒體活著：DeleteEventBridgeRuleTemplateGroup
	媒體：DeleteInput
	媒體活著：DeleteInputSecurityGroup
	媒體活著：DeleteMultiplex
	媒體活著:DeleteMultiplex程序
	媒體活著：DeleteReservation
	媒體活著：DeleteSchedule
	媒體活著:DeleteSignal地圖
	媒體活著:DescribeAccount配置
	媒體活著：DescribeChannel
	媒體活著：DescribeInput
	媒體活著:DescribeInput設備
	媒體活著：DescribeInputDeviceThumbnail
	媒體活著：DescribeInputSecurityGroup
	媒體活著：DescribeMultiplex
	媒體活著:DescribeMultiplex程序
	媒體活著：DescribeOffering
	媒體活著：DescribeReservation
	媒體活著：DescribeSchedule
	媒體活著：DescribeThumbnails

服務前綴	動作
	媒體:GetCloudWatchAlarm模板
	媒體活著：GetCloudWatchAlarmTemplateGroup
	媒體:GetEventBridgeRule模板
	媒體活著：GetEventBridgeRuleTemplateGroup
	媒體活著:GetSignal地圖
	媒體活著：ListChannels
	媒體活著：ListCloudWatchAlarmTemplateGroups
	媒體活著:ListCloudWatchAlarm模板
	媒體活著：ListEventBridgeRuleTemplateGroups
	媒體活著:ListEventBridgeRule模板
	媒體活著:ListInput設備
	媒體活著：ListInputDeviceTransfers
	媒體活著：ListInputs
	媒體活著：ListInputSecurityGroups
	媒體活著：ListMultiplexes
	媒體活著:ListMultiplex程序
	媒體活著：ListOfferings
	媒體活著：ListReservations
	媒體:地圖 ListSignal
	媒體活著：PurchaseOffering
	媒體活著:RebootInput設備

服務前綴	動作
	媒體活著：RejectInputDeviceTransfer
	媒體活著：RestartChannel管道
	媒體活著：StartChannel
	媒體活著：StartDeleteMonitorDeployment
	媒體活著:StartInput設備
	媒體活著:StartInputDeviceMaintenance視窗
	媒體活著:StartMonitor部署
	媒體活著：StartMultiplex
	媒體活著：StartUpdateSignalMap
	媒體活著：StopChannel
	媒體活著:StopInput設備
	媒體活著：StopMultiplex
	媒體活著:TransferInput設備
	媒體活著:UpdateAccount配置
	媒體活著：UpdateChannel
	媒體:類 UpdateChannel
	媒體:UpdateCloudWatchAlarm模板
	媒體活著：UpdateCloudWatchAlarmTemplateGroup
	媒體:UpdateEventBridgeRule模板
	媒體活著：UpdateEventBridgeRuleTemplateGroup
	媒體活著：UpdateInput

服務前綴	動作
	媒體活著:UpdateInput設備 媒體活著：UpdateInputSecurityGroup 媒體活著：UpdateMultiplex 媒體活著:UpdateMultiplex程序 媒體活著：UpdateReservation

服務前綴	動作
mediapackage	媒體包：ConfigureLogs 媒體包：CreateChannel 媒體包：Job CreateHarvest 媒體套件:端點 CreateOrigin 媒體包：DeleteChannel 媒體套件:端點 DeleteOrigin 媒體包：DescribeChannel 媒體包：Job DescribeHarvest 媒體套件:端點 DescribeOrigin 媒體包：ListChannels 媒體包：工作 ListHarvest 媒體套件:端點 ListOrigin 媒體包：ListTagsForResource 媒體套件：憑據 RotateChannel 媒體包：RotateIngestEndpointCredentials 媒體包：TagResource 媒體包：UntagResource 媒體包：UpdateChannel 媒體套件:端點 UpdateOrigin

服務前綴	動作
mediapackage-vod	媒體套件點播：ConfigureLogs 媒體套件點播：CreateAsset 媒體套件點播：配置 CreatePackaging 媒體套件點播:集團 CreatePackaging 媒體套件點播：DeleteAsset 媒體套件點播：配置 DeletePackaging 媒體套件點播:集團 DeletePackaging 媒體套件點播：DescribeAsset 媒體套件點播：配置 DescribePackaging 媒體套件點播:集團 DescribePackaging 媒體套件點播：ListAssets 媒體套件點播：配置 ListPackaging 媒體套件點播:群組 ListPackaging 媒體套件點播：ListTagsForResource 媒體套件點播：TagResource 媒體套件點播：UntagResource 媒體套件點播:集團 UpdatePackaging

服務前綴	動作
mediastore	媒體商店：CreateContainer 媒體商店：DeleteContainer 媒體商店:政策 DeleteContainer 媒體商店:政策 DeleteCors 媒體商店:政策 DeleteLifecycle 媒體商店:政策 DeleteMetric 媒體商店：DescribeContainer 媒體商店:政策 GetContainer 媒體商店:政策 GetCors 媒體商店:政策 GetLifecycle 媒體商店:政策 GetMetric 媒體商店：ListContainers 媒體商店:政策 PutContainer 媒體商店:政策 PutCors 媒體商店:政策 PutLifecycle 媒體商店:政策 PutMetric 媒體存儲:記錄 StartAccess 媒體存儲:記錄 StopAccess

服務前綴	動作
mediatailor	媒體:配置 ConfigureLogs ForPlayback 媒體:CreateChannel 媒體：來源 CreateLive 媒體:排程 CreatePrefetch 媒體:CreateProgram 媒體:位置 CreateSource 媒體：來源 CreateVod 媒體:DeleteChannel 媒體:政策 DeleteChannel 媒體：來源 DeleteLive 媒體:配置 DeletePlayback 媒體:排程 DeletePrefetch 媒體:DeleteProgram 媒體:位置 DeleteSource 媒體：來源 DeleteVod 媒體:DescribeChannel 媒體：來源 DescribeLive 媒體:DescribeProgram 媒體:位置 DescribeSource 媒體：來源 DescribeVod 媒體:政策 GetChannel

服務前綴	動作
	媒體:排程 GetChannel
	媒體:配置 GetPlayback
	媒體:排程 GetPrefetch
	媒體:ListAlerts
	媒體:ListChannels
	媒體:來源 ListLive
	媒體:配置 ListPlayback
	媒體：排程 ListPrefetch
	媒體:位置 ListSource
	媒體:來源 ListVod
	媒體:政策 PutChannel
	媒體:配置 PutPlayback
	媒體:StartChannel
	媒體:StopChannel
	媒體:UpdateChannel
	媒體：來源 UpdateLive
	媒體:UpdateProgram
	媒體:位置 UpdateSource
	媒體：來源 UpdateVod

服務前綴	動作
memorydb	內存數據庫：集群 BatchUpdate 內存數據庫：CopySnapshot 內存數據庫：CreateAcl 內存數據庫：CreateCluster 內存數據庫：組 CreateParameter 內存數據庫：CreateSnapshot 內存數據庫：組 CreateSubnet 內存數據庫：CreateUser 內存數據庫：DeleteAcl 內存數據庫：DeleteCluster 內存數據庫：組 DeleteParameter 內存數據庫：DeleteSnapshot 內存數據庫：組 DeleteSubnet 內存數據庫：DeleteUser 內存數據庫：DescribeAcls 內存數據庫：DescribeClusters 內存數據庫：版本 DescribeEngine 內存數據庫：DescribeEvents 內存數據庫：組 DescribeParameter 內存數據庫：DescribeParameters 內存數據庫：節點 DescribeReserved

服務前綴	動作
	內存數據庫：DescribeReservedNodesOfferings 內存數據庫：更新 DescribeService 內存數據庫：DescribeSnapshots 內存數據庫：組 DescribeSubnet 內存數據庫：DescribeUsers 內存數據庫：FailoverShard 內存數據庫：更新 ListAllowed NodeType 內存數據庫：PurchaseReservedNodesOffering 內存數據庫：組 ResetParameter 內存數據庫：UpdateAcl 內存數據庫：UpdateCluster 內存數據庫：組 UpdateParameter 內存數據庫：組 UpdateSubnet 內存數據庫：UpdateUser

服務前綴	動作
mgh	mgh: Artifact AssociateCreated mgh: AssociateDiscovered 資源 兆赫：CreateHomeRegionControl 兆赫：CreateProgressUpdateStream 兆赫：DeleteHomeRegionControl 兆赫：DeleteProgressUpdateStream mgh: 州 DescribeApplication 兆赫：DescribeHomeRegionControls MGH：DescribeMigration任務 mgh: Artifact DisassociateCreated mgh: DisassociateDiscovered 資源 mgh: GetHome 區域 MGH：ImportMigration任務 mgh: ListApplication 美國 mgh: ListCreated 文物 mgh: ListDiscovered 資源 mgh：ListMigration任務 兆赫：ListProgressUpdateStreams mgh: 州 NotifyApplication 兆赫：NotifyMigrationTaskState mgh：PutResource屬性

服務前綴	動作
mgn	毫克:ArchiveApplication 毫克:ArchiveWave 毫克:AssociateApplications 點AssociateSource:伺服器 位置:狀態 ChangeServer LifeCycle 毫克:CreateApplication 毫克:CreateConnector 毫克:CreateLaunchConfigurationTemplate 毫克:CreateReplicationConfigurationTemplate 毫克:CreateWave 毫克:DeleteApplication 毫克:DeleteConnector 毫克:DeleteJob 毫克:DeleteLaunchConfigurationTemplate 毫克:DeleteReplicationConfigurationTemplate 點DeleteSource:伺服器 點DeleteVcenter:用戶端 毫克:DeleteWave 毫克:DescribeJobLogItems 毫克:DescribeJobs 毫克:DescribeLaunchConfigurationTemplates

服務前綴	動作
	毫克:DescribeReplicationConfigurationTemplates 點DescribeVcenter:客戶端 毫克:DisassociateApplications 點DisassociateSource:伺服器 點:DisconnectFrom服務 毫克:FinalizeCutover 點:GetReplication配置 毫克:InitializeService 毫克:ListConnectors 錯誤：錯誤 ListExport 毫克:ListExports 錯誤：錯誤 ListImport 毫克:ListImports 使用者:ListManaged帳戶 毫克:ListSourceServerActions 一個:ListTemplate行動 點MarkAs:已存檔 毫克:PauseReplication 毫克:PutSourceServerAction 遊戲:PutTemplate行動 毫克:RemoveSourceServerAction

服務前綴	動作
	遊戲:RemoveTemplate行動 毫克:ResumeReplication 點:RetryData複製 毫克:StartCutover 毫克:StartExport 毫克:StartImport 毫克:StartReplication 毫克:StartTest 毫克:StopReplication 時間:TerminateTarget實例 毫克:UnarchiveApplication 毫克:UnarchiveWave 毫克:UpdateApplication 毫克:UpdateConnector 毫克:UpdateLaunchConfigurationTemplate 點:UpdateReplication配置 毫克:UpdateReplicationConfigurationTemplate 點UpdateSource:伺服器 類型：類型 UpdateSource ServerReplication 毫克:UpdateWave

服務前綴	動作
migrationhub-strategy	遷移集線器策略：模式 GetAnti
	遷移中心策略：GetApplicationComponentDetails
	遷移中心策略：GetApplicationComponentStrategies
	遷移中心策略：GetAssessment
	遷移中心策略：GetImportFileTask
	遷移中心策略：GetLatestAssessmentId
	遷移中心策略：GetMessage
	遷移中心策略：首選項 GetPortfolio
	遷移中心策略：摘要 GetPortfolio
	遷移中心策略：GetRecommendationReportDetails
	遷移中心策略：詳細信息 GetServer
	遷移中心策略：策略 GetServer
	遷移中心策略：服務器 ListAnalyzable
	遷移中心策略：模式 ListAnti
	遷移集線器策略：組件 ListApplication
	遷移中心策略：ListCollectors
	遷移中心策略：ListImportFileTask
	遷移中心策略：文物 ListJar
	遷移中心策略：ListServers
	遷移中心策略：首選項 PutPortfolio
	遷移中心策略：RegisterCollector

服務前綴	動作
	遷移中心策略：SendMessage
	遷移中心策略：StartAssessment
	遷移中心策略：StartImportFileTask
	遷移中心策略：StartRecommendationReportGeneration
	遷移中心策略：StopAssessment
	遷移中心策略：UpdateApplicationComponentConfig
	遷移中心策略：配置 UpdateCollector
	遷移中心策略：Config UpdateServer

服務前綴	動作
mobiletargeting	動員：CreateApp 動員：CreateCampaign 動員設計:模板 CreateEmail 移動:Job CreateExport 移動:Job CreateImport 動員：CreateInAppTemplate 動員：CreateJourney 動員設計:模板 CreatePush 移動設備：配置 CreateRecommender 動員：CreateSegment 動員設計:模板 CreateSms 動員設計:模板 CreateVoice 移動:頻道 DeleteAdm 移動:頻道 DeleteApns 動員：DeleteApnsSandboxChannel 動員：DeleteApnsVoipChannel 移動:頻道 DeleteApns VoipSandbox 動員：DeleteApp 移動:頻道 DeleteBaidu 動員：DeleteCampaign 移動:頻道 DeleteEmail

服務前綴	動作
	動員設計:模板 DeleteEmail 動員：DeleteEndpoint 移動:流 DeleteEvent 移動:頻道 DeleteGcm 動員：DeleteInAppTemplate 動員：DeleteJourney 動員設計:模板 DeletePush 移動設備：配置 DeleteRecommender 動員：DeleteSegment 移動:頻道 DeleteSms 動員設計:模板 DeleteSms 行動裝置:端點 DeleteUser 移動:頻道 DeleteVoice 動員設計:模板 DeleteVoice 移動:頻道 GetAdm 移動:頻道 GetApns 動員：GetApnsSandboxChannel 動員：GetApnsVoipChannel 移動:頻道 GetApns VoipSandbox 動員：GetApp 移動:關鍵績效指標 GetApplication DateRange

服務前綴	動作
	移動設置：設置 GetApplication 動員：GetApps 移動:頻道 GetBaidu 動員：GetCampaign 移動：活動 GetCampaign 移動:關鍵績效指標 GetCampaign DateRange 動員：GetCampaigns 移動:版本 GetCampaign 行動裝置:版本 GetCampaign 動員：GetChannels 移動:頻道 GetEmail 動員設計:模板 GetEmail 動員：GetEndpoint 移動:流 GetEvent 移動:Job GetExport 移動:工作 GetExport 移動:頻道 GetGcm 移動:Job GetImport 移動:工作 GetImport 動員：GetInAppMessages 動員：GetInAppTemplate

服務前綴	動作
	動員：GetJourney 移動:關鍵績效指標 GetJourney DateRange 行動裝置:度量 GetJourney ExecutionActivity 動員：GetJourneyExecutionMetrics 動員：GetJourneyRunExecutionActivityMetrics 行動裝置:度量 GetJourney RunExecution 移動:運行 GetJourney 動員設計:模板 GetPush 移動設備：配置 GetRecommender 移動設備：配置 GetRecommender 動員：GetSegment 動員：GetSegmentExportJobs 動員：GetSegmentImportJobs 動員：GetSegments 移動:版本 GetSegment 行動裝置:版本 GetSegment 移動:頻道 GetSms 動員設計:模板 GetSms 行動裝置:端點 GetUser 移動:頻道 GetVoice 動員設計:模板 GetVoice

服務前綴	動作
	動員：ListJourneys 動員：ListTemplates 行動裝置:版本 ListTemplate 行動裝置:驗證 PhoneNumber 移動:流 PutEvent 動員：RemoveAttributes 移動:頻道 UpdateAdm 移動:頻道 UpdateApns 動員：UpdateApnsSandboxChannel 動員：UpdateApnsVoipChannel 移動:頻道 UpdateApns VoipSandbox 移動設置：設置 UpdateApplication 移動:頻道 UpdateBaidu 動員：UpdateCampaign 移動:頻道 UpdateEmail 動員設計:模板 UpdateEmail 動員：UpdateEndpoint 移動：Batch UpdateEndpoints 移動:頻道 UpdateGcm 動員：UpdateInAppTemplate 動員：UpdateJourney

服務前綴	動作
	動員：狀態 UpdateJourney 動員設計:模板 UpdatePush 移動設備：配置 UpdateRecommender 動員：UpdateSegment 移動:頻道 UpdateSms 動員設計:模板 UpdateSms 動員：UpdateTemplateActiveVersion 移動:頻道 UpdateVoice 動員設計:模板 UpdateVoice mobiletargeting:VerifyOTPMessage

服務前綴	動作
mq	mq：CreateBroker mq：CreateConfiguration mq：CreateUser mq：DeleteBroker mq：DeleteUser mq：DescribeBroker mq：DescribeBrokerEngineTypes mq：DescribeBrokerInstanceOptions mq：DescribeConfiguration mq：DescribeConfiguration修訂版本 mq：DescribeUser mq：ListBrokers mq：ListConfiguration修訂版本 mq：ListConfigurations mq：ListUsers mq:Promote mq：RebootBroker mq：UpdateBroker mq：UpdateConfiguration mq：UpdateUser

服務前綴	動作
networkmanager	網路管理員：AcceptAttachment 網路管理員:AssociateConnect對等 網路管理員:閘AssociateCustomer道 網路管理員：AssociateLink 網路管理員:AssociateTransitGatewayConnect對等 網路管理員:附CreateConnect件 網路管理員：CreateConnection 網路管理員:CreateConnect對等 網路管理員:CreateCore網路 網路管理員：CreateDevice 網路管理員:CreateGlobal網路 網路管理員：CreateLink 網路管理員：CreateSite 網路管理員：CreateSiteToSiteVpnAttachment 網路管理員：CreateTransitGatewayPeering 網路管理員：CreateTransitGatewayRouteTableAttachment 網路管理員:附CreateVpc件 網路管理員：DeleteAttachment 網路管理員：DeleteConnection 網路管理員:DeleteConnect對等 網路管理員:DeleteCore網路

服務前綴	動作
	網路管理員:DeleteCoreNetworkPolicy版本 網路管理員：DeleteDevice 網路管理員:DeleteGlobal網路 網路管理員：DeleteLink 網路管理員：DeletePeering 網路管理員:政策 DeleteResource 網路管理員：DeleteSite 網路管理員:閘DeregisterTransit道 網路管理員:DescribeGlobal網路 網路管理員:DisassociateConnect對等 網路管理員:閘DisassociateCustomer道 網路管理員：DisassociateLink 網路管理員:DisassociateTransitGatewayConnect對等 網路管理員：ExecuteCoreNetworkChange設定 網路管理員:附GetConnect件 網路管理員：GetConnections 網路管理員:GetConnect對等 網路管理員：GetConnectPeerAssociations 網路管理員:GetCore網路 網路管理員:GetCoreNetworkChange事件 網路管理員：GetCoreNetworkChange設定

服務前綴	動作
	網路管理員：GetCoreNetworkPolicy 網路管理員：GetCustomerGatewayAssociations 網路管理員：GetDevices 網路管理員:GetLink關聯 網路管理員：GetLinks 網路管理員：GetNetworkResourceCounts 網路管理員：GetNetworkResourceRelationships 網路管理員:GetNetwork資源 網路管理員:GetNetwork路由 網路管理員:遙測 GetNetwork 網路管理員:政策 GetResource 網路管理員:GetRoute分析 網路管理員：GetSites 網路管理員：GetSiteToSiteVpnAttachment 網路管理員：GetTransitGatewayConnectPeerAssociations 網路管理員：GetTransitGatewayPeering 網路管理員：GetTransitGatewayRegistrations 網路管理員：GetTransitGatewayRouteTableAttachment 網路管理員:附GetVpc件 網路管理員：ListAttachments 網路管理員:ListConnect同行

服務前綴	動作
	網路管理員：ListCoreNetworkPolicy版本 網路管理員:ListCore網路 網路管理員:狀態 ListOrganization ServiceAccess 網路管理員：ListPeerings 網路管理員：PutCoreNetworkPolicy 網路管理員:政策 PutResource 網路管理員:閘RegisterTransit道 網路管理員：RejectAttachment 網路管理員:RestoreCoreNetworkPolicy版本 網路管理員:StartOrganizationServiceAccess更新 網路管理員:StartRoute分析 網路管理員：UpdateConnection 網路管理員:UpdateCore網路 網路管理員：UpdateDevice 網路管理員:UpdateGlobal網路 網路管理員：UpdateLink 網路管理員：UpdateNetworkResourceMetadata 網路管理員：UpdateSite 網路管理員:附UpdateVpc件

服務前綴	動作
nimble	靈活：AcceptEulas
	靈活：CreateLaunch配置文件
	靈活：圖CreateStreaming像
	靈活：CreateStreaming會話
	靈活：CreateStreamingSessionStream
	靈活：CreateStudio
	靈活：CreateStudio組件
	靈活：DeleteLaunch配置文件
	靈活：DeleteLaunchProfileMember
	靈活：圖DeleteStreaming像
	靈活：DeleteStreaming會話
	靈活：DeleteStudio
	靈活：DeleteStudio組件
	靈活：DeleteStudio成員
	靈活：GetEula
	靈活：GetLaunchProfileDetails
	靈活：圖GetStreaming像
	靈活：GetStreaming會話
	靈活：GetStreamingSessionBackup
	靈活：GetStreamingSessionStream
	靈活：GetStudio

服務前綴	動作
	靈活：GetStudio組件
	靈活：GetStudio成員
	靈活：ListEulas
	靈活：ListLaunchProfileMembers
	靈活：ListLaunch配置文件
	靈活：圖ListStreaming像
	靈活：ListStreamingSessionBackups
	靈活：ListStreaming會議
	靈活：ListStudio組件
	靈活：ListStudio成員
	靈活：ListStudios
	靈活：PutLaunchProfileMembers
	靈活：PutStudio成員
	靈活：StartStreaming會話
	靈活：SSO StartStudio ConfigurationRepair
	靈活：StopStreaming會話
	靈活：UpdateLaunch配置文件
	靈活：UpdateLaunchProfileMember
	靈活：圖UpdateStreaming像
	靈活：UpdateStudio
	靈活：UpdateStudio組件

服務前綴	動作
omics	組學:AbortMultipartReadSet上傳 組學：BatchDeleteReadSet 組學：CancelAnnotationImportJob 組學：CancelRun 組學：CancelVariantImportJob 組學:CompleteMultipartReadSet上傳 組學:CreateAnnotation商店 組學:CreateMultipartReadSet上傳 組學:CreateReference商店 組學:CreateRun組 組學:CreateSequence商店 組學:CreateVariant商店 組學：CreateWorkflow 組學:DeleteAnnotation商店 組學：DeleteReference 組學:DeleteReference商店 組學：DeleteRun 組學:DeleteRun組 組學:DeleteSequence商店 組學:DeleteVariant商店 組學：DeleteWorkflow

服務前綴	動作
	組學：GetAnnotationImportJob 組學:GetAnnotation商店 組學:GetRead設置 組學:Job GetRead SetActivation 組學:Job GetRead SetExport 組學:Job GetRead SetImport 組學：GetReadSetMetadata 組學：GetReference 組學：GetReferenceImportJob 組學:GetReference元數據 組學:GetReference商店 組學：GetRun 組學:GetRun組 組學：GetRun任務 組學:GetSequence商店 組學：GetVariantImportJob 組學:GetVariant商店 組學：GetWorkflow 組學：ListAnnotationImportJobs 組學:ListAnnotation商店 組學:ListMultipartReadSet上傳

服務前綴	動作
	組學:ListReadSetActivation工作 組學:ListReadSetExport工作 組學:ListReadSetImport工作 組學:ListRead集 組學:ListReadSetUpload零件 組學：ListReferenceImportJobs 組學：ListReferences 組學:ListReference商店 組學:ListRun群組 組學：ListRuns 組學：ListRun任務 組學:ListSequence商店 組學：ListVariantImportJobs 組學:ListVariant商店 組學：ListWorkflows 組學：StartAnnotationImportJob 組學:Job StartRead SetActivation 組學:Job StartRead SetExport 組學:Job StartRead SetImport 組學：StartReferenceImportJob 組學：StartRun

服務前綴	動作
	組學：StartVariantImportJob 組學:UpdateAnnotation商店 組學:UpdateRun組 組學:UpdateVariant商店 組學：UpdateWorkflow 組學：UploadReadSetPart

服務前綴	動作
opsworks	作品:AssignInstance
	作品:AssignVolume
	作品:IP AssociateElastic
	作品:AttachElasticLoadBalancer
	作品:CloneStack
	作品:CreateApp
	作品:CreateDeployment
	作品:CreateInstance
	作品:CreateLayer
	作品:CreateStack
	作品:設定檔 CreateUser
	作品:DeleteApp
	作品:DeleteInstance
	作品:DeleteLayer
	作品:DeleteStack
	作品:設定檔 DeleteUser
	作業:叢集 DeregisterEcs
	作品:IP DeregisterElastic
	作品:DeregisterInstance
	作品:DeregisterRdsDbInstance
	作品:DeregisterVolume

服務前綴	動作
	作品:版本 DescribeAgent
	作品:DescribeApps
	作品:DescribeCommands
	作品:DescribeDeployments
	作業：叢集 DescribeEcs
	操作:Ips DescribeElastic
	作品:DescribeElasticLoadBalancers
	作品:DescribeInstances
	作品:DescribeLayers
	作業:縮放 DescribeLoad BasedAuto
	作品:DescribeMyUserProfile
	作業：系統 DescribeOperating
	作品:DescribePermissions
	作業:陣列 DescribeRaid
	作品:DescribeRdsDbInstances
	操作：錯誤 DescribeService
	作品:DescribeStackProvisioningParameters
	作品:DescribeStacks
	作品：摘要 DescribeStack
	作業:縮放 DescribeTime BasedAuto
	作業:設定檔 DescribeUser

服務前綴	動作
	作品:DescribeVolumes
	作品:DetachElasticLoadBalancer
	作品:IP DisassociateElastic
	作品：建議 GetHostname
	作品:GrantAccess
	作品:RebootInstance
	作業:叢集 RegisterEcs
	作品:IP RegisterElastic
	作品:RegisterInstance
	作品:RegisterRdsDbInstance
	作品:RegisterVolume
	作業:縮放 SetLoad BasedAuto
	作品:SetPermission
	作業:縮放 SetTime BasedAuto
	作品:StartInstance
	作品:StartStack
	作品:StopInstance
	作品:StopStack
	作品:UnassignInstance
	作品:UnassignVolume
	作品:UpdateApp

服務前綴	動作
	作品:IP UpdateElastic 作品:UpdateInstance 作品:UpdateLayer 作品:UpdateMyUserProfile 作品:UpdateRdsDbInstance 作品:UpdateStack 作品:設定檔 UpdateUser 作品:UpdateVolume

服務前綴	動作
opsworks-cm	尺寸-厘米:AssociateNode 尺寸-厘米:CreateBackup 尺寸-厘米:CreateServer 尺寸-厘米:DeleteBackup 尺寸-厘米:DeleteServer 選項-厘米：屬性 DescribeAccount 尺寸-厘米:DescribeBackups 尺寸-厘米:DescribeEvents 尺寸-厘米:DescribeNodeAssociationStatus 尺寸-厘米:DescribeServers 尺寸-厘米:DisassociateNode 尺寸-厘米:ExportServerEngineAttribute 尺寸-厘米:RestoreServer 尺寸-厘米:StartMaintenance 尺寸-厘米:UpdateServer 尺寸-厘米:UpdateServerEngineAttributes

服務前綴	動作
組織	組織：AcceptHandshake
	組織：AttachPolicy
	組織：CancelHandshake
	組織：CloseAccount
	組織：CreateAccount
	組織：CreateGovCloudAccount
	組織：CreateOrganization
	組織:CreateOrganizational單位
	組織：CreatePolicy
	組織：DeclineHandshake
	組織：DeleteOrganization
	組織:DeleteOrganizational單位
	組織：DeletePolicy
	組織:DeleteResource政策
	組織:DeregisterDelegated管理員
	組織：DescribeAccount
	組織：DescribeCreateAccountStatus
	組織:DescribeEffective政策
	組織：DescribeHandshake
	組織：DescribeOrganization
	組織:DescribeOrganizational單位

服務前綴	動作
	組織：DescribePolicy
	組織:DescribeResource政策
	組織：DetachPolicy
	組織:停用 AWSServiceAccess
	組織:DisablePolicy類型
	組織:EnableAll功能
	組織:啟用 AWSServiceAccess
	組織:EnablePolicy類型
	組織：InviteAccountToOrganization
	組織：LeaveOrganization
	組織：ListAccounts
	組織：ListAccountsForParent
	組織:清單 AWSServiceAccessForOrganization
	組織：ListChildren
	組織：ListCreateAccountStatus
	組織:ListDelegated管理員
	組織:ListDelegatedServicesFor帳戶
	組織：ListHandshakesForAccount
	組織：ListHandshakesForOrganization
	組織:ListOrganizationalUnitsFor父系
	組織：ListParents

服務前綴	動作
	組織：ListPolicies 組織：ListPoliciesForTarget 組織：ListRoots 組織：ListTargetsForPolicy 組織：MoveAccount 組織:PutResource政策 組織:RegisterDelegated管理員 組織：RemoveAccountFromOrganization 組織:UpdateOrganizational單位 組織：UpdatePolicy

服務前綴	動作
outposts	前哨：CancelCapacity任務 前哨：CancelOrder 前哨：CreateOrder 前哨：CreateOutpost 前哨：CreatePrivateConnectivityConfig 前哨：CreateSite 前哨：DeleteOutpost 前哨：DeleteSite 前哨：GetCapacity任務 前哨：項目 GetCatalog 前哨：GetConnection 前哨：GetOrder 前哨：GetOutpost 前哨：GetOutpostInstanceTypes 前哨：類型 GetOutpost SupportedInstance 前哨：GetPrivateConnectivityConfig 前哨：GetSite 前哨：地址 GetSite 前哨：ListAssets 前哨：ListCapacity任務 前哨：項目 ListCatalog

服務前綴	動作
	前哨：ListOrders 前哨：ListOutposts 前哨：ListSites 前哨：StartCapacity任務 前哨：StartConnection 前哨：UpdateOutpost 前哨：UpdateSite 前哨：地址 UpdateSite 前哨：UpdateSiteRackPhysical屬性

服務前綴	動作
panorama	全景：CreateApplication實例
	全景：CreateJobForDevices
	全景:CreateNodeFromTemplateJob
	全景：CreatePackage
	全景：CreatePackageImportJob
	全景：DeleteDevice
	全景：DeletePackage
	全景:DeregisterPackage版本
	全景：DescribeApplication實例
	全景：DescribeApplicationInstanceDetails
	全景：DescribeDevice
	全景:DescribeDeviceJob
	全景：DescribeNode
	全景:DescribeNodeFromTemplateJob
	全景：DescribePackage
	全景：DescribePackageImportJob
	全景:DescribePackage版本
	全景：ListApplicationInstanceDependencies
	全景:ListApplicationInstanceNode例證
	全景:ListApplication例證
	全景：ListDevices

服務前綴	動作
	全景:ListDevices工作 全景:ListNodeFromTemplate工作 全景：ListNodes 全景：ListPackageImportJobs 全景：ListPackages 全景：ProvisionDevice 全景:RegisterPackage版本 全景：RemoveApplication實例 全景:SignalApplicationInstanceNode例證 全景:UpdateDevice中繼資料
pi	圓周率:CreatePerformanceAnalysisReport 圓周率:DeletePerformanceAnalysisReport 圓周率：DescribeDimension鑰匙 圓周率:GetDimensionKeyDetails 圓周率:GetPerformanceAnalysisReport pi: GetResource 中繼資料 圓周率:GetResource度量 圓周率:ListAvailableResourceDimensions 圓周率:ListAvailableResourceMetrics 圓周率:ListPerformanceAnalysisReports

服務前綴	動作
pipes	管道：CreatePipe 管道：DeletePipe 管道：DescribePipe 管道：ListPipes 管道：StartPipe 管道：StopPipe 管道：UpdatePipe
polly	波莉：DeleteLexicon 波莉：DescribeVoices 波莉：GetLexicon 波莉：GetSpeechSynthesisTask 波莉：ListLexicons 波莉：ListSpeechSynthesisTasks 波莉：PutLexicon 波莉：StartSpeechSynthesisTask 波莉：SynthesizeSpeech

服務前綴	動作
profile	設定檔：AddProfile鍵
	設定檔：CreateCalculatedAttributeDefinition
	設定檔：CreateDomain
	設定檔：CreateEvent串流
	設定檔：CreateProfile
	設定檔：DeleteCalculatedAttributeDefinition
	設定檔：DeleteDomain
	設定檔：DeleteEvent串流
	設定檔：DeleteIntegration
	設定檔：DeleteProfile
	設定檔：DeleteProfile鍵
	設定檔:DeleteProfile物件
	設定檔：DeleteProfileObjectType
	設定檔：DeleteWorkflow
	設定檔：DetectProfileObjectType
	設定檔：GetAutoMergingPreview
	設定檔：GetCalculatedAttributeDefinition
	設定檔：GetCalculatedAttributeFor設定檔
	設定檔：GetDomain
	設定檔：GetEvent串流
	設定檔：GetIdentityResolutionJob

服務前綴	動作
	設定檔：GetIntegration
	設定檔：GetMatches
	設定檔：GetProfileObjectType
	設定檔:GetProfileObjectType範本
	設定檔:GetSimilar設定檔
	設定檔：GetWorkflow
	設定檔:GetWorkflow步驟
	設定檔:ListAccount整合
	設定檔：ListCalculatedAttributeDefinitions
	設定檔：ListCalculatedAttributesFor設定檔
	設定檔：ListDomains
	設定檔:ListEvent串流
	設定檔：ListIdentityResolutionJobs
	設定檔：ListIntegrations
	設定檔:ListProfile物件
	設定檔：ListProfileObjectTypes
	設定檔:ListProfileObjectType範本
	設定檔：ListRuleBasedMatches
	設定檔：ListWorkflows
	設定檔：MergeProfiles
	設定檔：PutIntegration

服務前綴	動作
	設定檔:PutProfile物件 設定檔：PutProfileObjectType 設定檔：SearchProfiles 設定檔：UpdateCalculatedAttributeDefinition 設定檔：UpdateDomain 設定檔：UpdateProfile

服務前綴	動作
qldb	qldb: CancelJournal KinesisStream qldb: CreateLedger qldb: DeleteLedger qldb: DescribeJournal KinesisStream 匯出 DescribeJournal qldb: DescribeLedger qldb: 至 3 ExportJournal qldb: GetBlock qldb: GetDigest qldb: GetRevision qldb: ListJournal KinesisStreams ForLedger 出口:第三季出口 ListJournal 分析庫：ListJournalS3 分類帳 ExportsFor qldb: ListLedgers qldb: StreamJournal ToKinesis qldb: UpdateLedger qldb: UpdateLedger PermissionsMode

服務前綴	動作
ram	公羊：AcceptResourceShareInvitation
	公羊：AssociateResource分享
	公羊：AssociateResourceSharePermission
	公羊：CreatePermission
	內存：CreatePermission版本
	公羊：CreateResource分享
	公羊：DeletePermission
	內存：DeletePermission版本
	公羊：DeleteResource分享
	公羊：DisassociateResource分享
	公羊：DisassociateResourceSharePermission
	內存：EnableSharingWithAws組織
	公羊：GetPermission
	內存：GetResource政策
	公羊：GetResourceShareAssociations
	公羊：GetResourceShareInvitations
	內存：GetResource股票
	公羊：ListPendingInvitationResources
	公羊：ListPermission協會
	公羊：ListPermissions
	內存：ListPermission版本

服務前綴	動作
	公羊：ListPrincipals 公羊：ListReplacePermissionAssociations工作 公羊：ListResources 公羊：ListResourceSharePermissions 內存：ListResource類型 內存：PromotePermissionCreatedFrom策略 公羊：PromoteResourceShareCreatedFromPolicy 公羊：RejectResourceShareInvitation 公羊：ReplacePermission協會 公羊：SetDefaultPermissionVersion 公羊：UpdateResource分享
rbin	資料夾：CreateRule 資料夾：DeleteRule 資料夾：GetRule 資料夾：ListRules 資料夾：LockRule 資料夾：UnlockRule 資料夾：UpdateRule

服務前綴	動作
rds	rds：目AddRole標資料庫叢集
	rds：AddRole待資料庫執行個體
	rds: AddSource IdentifierTo 訂閱
	RDS：ApplyPendingMaintenanceAction
	RDSSecurityGroup: 作者教育局入口
	rds:BacktrackDBCluster
	RDS：CancelExport任務
	RDS: 複ClusterParameter製資料庫群組
	RDS: 複製資料庫 ClusterSnapshot
	RDS: 複製資料庫 ParameterGroup
	rds:CopyDBSnapshot
	rds: CopyOption 群組
	RDS: CreateCustom 資料庫 EngineVersion
	RDS: ClusterParameter 建立資料庫群組
	RDS: 建立資料庫 ClusterSnapshot
	RDS: 建立資料庫 ParameterGroup
	rds:CreateDBProxy
	RDS: 建立資料庫 ProxyEndpoint
	RDS: 建立資料庫 SecurityGroup
	rds:CreateDBSnapshot
	RDS: 建立資料庫 SubnetGroup

服務前綴	動作
	rds: CreateEvent 訂閱
	rds: CreateGlobal 叢集
	rds: CreateOption 群組
	RDS：DeleteBlueGreenDeployment
	RDS: 刪除資ClusterAutomated料庫 Backup
	RDS: 刪除資ClusterParameter料庫群組
	RDS: 刪除資料庫 ClusterSnapshot
	RDS: 刪除資InstanceAutomated料庫 Backup
	RDS: 刪除資料庫 ParameterGroup
	rds:DeleteDBProxy
	RDS: 刪除資料庫 ProxyEndpoint
	RDS: 刪除資料庫 SecurityGroup
	rds:DeleteDBSnapshot
	RDS: 刪除資料庫 SubnetGroup
	rds: DeleteEvent 訂閱
	rds: DeleteGlobal 叢集
	rds: DeleteOption 群組
	RDS: 取消註冊資料庫 ProxyTargets
	rds：DescribeAccount屬性
	RDS：DescribeBlueGreenDeployments
	RDS：DescribeCertificates

服務前綴	動作
	RDSClusterAutomated: 描述 B 備份
	RDS: 描述 B ClusterBacktracks
	RDS: 描述 B ClusterEndpoints
	RDSClusterParameter: 描述 B 群組
	RDS: 描述 B ClusterParameters
	rds:DescribeDBClusters
	RDS: 描述屬性 ClusterSnapshot
	RDS: 描述 B ClusterSnapshots
	RDS: 描述 B EngineVersions
	RDSInstanceAutomated: 描述 B 備份
	rds:DescribeDBInstances
	RDS: 描述 B LogFiles
	RDS: 描述 B ParameterGroups
	rds:DescribeDBParameters
	rds:DescribeDBProxies
	RDS: 描述 B ProxyEndpoints
	RDSProxyTarget: 描述 B 群組
	RDS: 描述 B ProxyTargets
	RDS: 描述建議
	RDS: 描述 B SecurityGroups
	RDS: 描述 B SnapshotAttributes

服務前綴	動作
	rds:DescribeDBSnapshots
	rds: DescribeDb SnapshotTenant 資料庫
	RDS: 描述 B SubnetGroups
	rds：DescribeEngineDefaultCluster參數
	RDS：DescribeEngineDefaultParameters
	rds: DescribeEvent 類別
	RDS：DescribeEvents
	rds：DescribeEvent訂閱
	rds：DescribeExport工作
	rds：DescribeGlobal叢集
	RDS：DescribeIntegrations
	RDS：DescribeOptionGroupOptions
	rds: DescribeOption 群組
	RDS: DescribeOrderable 資料庫 InstanceOptions
	RDS：DescribePendingMaintenanceActions
	rds: DescribeReserved 資料庫執行個體
	RDS: DescribeReserved 資料庫 InstancesOfferings
	rds: DescribeSource 區域
	rds: DescribeTenant 資料庫
	RDS: DescribeValid 資料庫 InstanceModifications
	RDS: DownloadComplete 資料庫 LogFile

服務前綴	動作
	RDSLogFile: 下載資料庫部分
	rds:FailoverDBCluster
	rds: FailoverGlobal 叢集
	RDS: ModifyActivity 流
	RDS：ModifyCertificates
	RDS: ModifyCurrent 資料庫 ClusterCapacity
	RDS: 修改資料庫 ClusterEndpoint
	RDS: 修改資ClusterParameter料庫群組
	RDSClusterSnapshot: 修改資料庫屬性
	RDS: 修改資料庫 ParameterGroup
	rds:ModifyDBProxy
	RDS: 修改資料庫 ProxyEndpoint
	RDS: 修改資ProxyTarget料庫群組
	RDS: 修改註釋
	rds:ModifyDBSnapshot
	RDS: 修改資料庫 SnapshotAttribute
	RDS: 修改資料庫 SubnetGroup
	rds: ModifyEvent 訂閱
	rds: ModifyGlobal 叢集
	rds: ModifyOption 群組
	rds: ModifyTenant 資料庫

服務前綴	動作
	RDS: PurchaseReserved 資料庫 InstancesOffering
	rds:RebootDBCluster
	RDS: 註冊資料庫 ProxyTargets
	RDS：RemoveFromGlobalCluster
	rds: RemoveRole 從叢集
	rds：RemoveRole從實例
	rds: RemoveSource IdentifierFrom 訂閱
	RDS: ClusterParameter 重設資料庫群組
	RDS: 重設資料庫 ParameterGroup
	RDSClusterFrom: 恢復列表 S3
	RDSClusterFrom: 還原快照
	RDS ClusterToPointIn: 恢復時間
	RDSInstanceFrom: 還原資料庫快照
	RDSInstanceFrom: 恢復列表 S3
	RDS InstanceToPointIn: 恢復時間
	RDSSecurityGroup: 撤銷資料庫輸入
	RDS: StartActivity 流
	rds:StartDBCluster
	rds:StartDBInstance
	RT: 起始資料庫 InstanceAutomated BackupsReplication
	RDS：StartExport任務

服務前綴	動作
	RDS: StopActivity 流 rds:StopDBCluster rds:StopDBInstance RD: 停止資料庫 InstanceAutomated BackupsReplication RDS：SwitchoverBlueGreenDeployment rds: SwitchoverGlobal 叢集 rds: SwitchoverRead 複本

服務前綴	動作
redshift	紅移：AcceptReservedNodeExchange 紅移：AddPartner 紅移：AssociateDataShareConsumer 紅移：AuthorizeClusterSecurityGroup入口 紅移：AuthorizeData分享 紅移：AuthorizeEndpoint訪問 紅移：AuthorizeSnapshot訪問 紅移：BatchDeleteClusterSnapshots 紅移：BatchModifyClusterSnapshots 紅移：CancelResize 紅移：CopyCluster快照 紅移：CreateAuthentication配置文件 紅移：CreateCluster 紅移：CreateClusterParameterGroup 紅移：CreateClusterSecurityGroup 紅移：CreateCluster快照 紅移：CreateClusterSubnetGroup 紅移：CreateCustomDomainAssociation 紅移：CreateEndpoint訪問 紅移：訂閱 CreateEvent 紅移：CreateHsmClientCertificate

服務前綴	動作
	紅移：CreateHsm配置 紅移：CreateRedshiftIdcApplication 紅移：CreateScheduled動作 紅移：CreateSnapshotCopyGrant 紅移：CreateSnapshot排程 紅移：CreateUsage限制 紅移：DeauthorizeData分享 紅移：DeleteAuthentication配置文件 紅移：DeleteCluster 紅移：DeleteClusterParameterGroup 紅移：DeleteClusterSecurityGroup 紅移：DeleteCluster快照 紅移：DeleteClusterSubnetGroup 紅移：DeleteCustomDomainAssociation 紅移：DeleteEndpoint訪問 紅移：訂閱 DeleteEvent 紅移：DeleteHsmClientCertificate 紅移：DeleteHsm配置 紅移：DeletePartner 紅移：DeleteScheduled動作 紅移：DeleteSnapshotCopyGrant

服務前綴	動作
	紅移：DeleteSnapshot排程 紅移：DeleteUsage限制 紅移：DescribeAccount屬性 紅移：DescribeAuthentication設定檔 紅移：DescribeClusterDbRevisions 紅移：DescribeClusterParameterGroups 紅移：DescribeCluster參數 紅移：DescribeClusters 紅移：DescribeClusterSecurityGroups 紅移：DescribeCluster快照 紅移：DescribeClusterSubnetGroups 紅移：DescribeCluster軌道 紅移：DescribeCluster版本 紅移：DescribeCustomDomainAssociations 紅移：股票 DescribeData 紅移：DescribeDataSharesFor消費者 紅移：DescribeDataSharesFor製作人 紅移：DescribeDefaultClusterParameters 紅移：DescribeEndpoint訪問 紅移：授權 DescribeEndpoint 紅移：類DescribeEvent別

服務前綴	動作
	紅移：DescribeEvents 紅移：訂閱 DescribeEvent 紅移：DescribeHsmClientCertificates 紅移：DescribeHsm配置 紅移：DescribeInbound集成 紅移：狀態 DescribeLogging 紅移：DescribeNodeConfigurationOptions 紅移：DescribeOrderableClusterOptions 紅移：DescribePartners 紅移：DescribeRedshiftIdcApplications 紅移：狀態 DescribeReserved NodeExchange 紅移：DescribeReservedNodeOfferings 紅移：節點 DescribeReserved 紅移：DescribeResize 紅移：DescribeScheduled動作 紅移：DescribeSnapshotCopyGrants 紅移：DescribeSnapshot排程 紅移：DescribeStorage 紅移：DescribeTableRestoreStatus 紅移：DescribeUsage限制 紅移：DisableLogging

服務前綴	動作
	紅移：DisableSnapshot複製 紅移：DisassociateDataShareConsumer 紅移：EnableLogging 紅移：EnableSnapshot複製 紅移：FailoverPrimary計算 紅移：憑證 GetCluster 紅移：IAM GetCluster CredentialsWith 紅移：GetReservedNodeExchangeConfigurationOptions 紅移：提供項目 GetReserved NodeExchange 紅移：ListRecommendations 紅移：ModifyAqua配置 紅移：ModifyAuthentication配置文件 紅移：ModifyCluster 紅移：ModifyClusterDbRevision 紅移：ModifyClusterIamRoles 紅移：維護 ModifyCluster 紅移：ModifyClusterParameterGroup 紅移：ModifyCluster快照 紅移：ModifyClusterSnapshotSchedule 紅移：ModifyClusterSubnetGroup 紅移：ModifyCustomDomainAssociation

服務前綴	動作
	紅移：ModifyEndpoint訪問 紅移：訂閱 ModifyEvent 紅移：ModifyScheduled動作 紅移：ModifySnapshotCopyRetention週期 紅移：ModifySnapshot排程 紅移：ModifyUsage限制 紅移：PauseCluster 紅移：PurchaseReservedNodeOffering 紅移：RebootCluster 紅移：RejectData分享 紅移：ResetClusterParameterGroup 紅移：ResizeCluster 紅移：RestoreFromClusterSnapshot 紅移：RestoreTableFromCluster快照 紅移：ResumeCluster 紅移：RevokeClusterSecurityGroup入口 紅移：RevokeEndpoint訪問 紅移：RevokeSnapshot訪問 紅移：RotateEncryption鍵 紅移：狀態 UpdatePartner

服務前綴	動作
redshift-data	標記資料：聲明 BatchExecute
	標記數據：CancelStatement
	標記數據：DescribeStatement
	標記數據：DescribeTable
	標記數據：ExecuteStatement
	標記資料：結果 GetStatement
	標記數據：ListDatabases
	標記數據：ListSchemas
	標記數據：ListStatements
	標記數據：ListTables

服務前綴	動作
refactor-spaces	重構空間：CreateApplication 重構空間：CreateEnvironment 重構空間：CreateRoute 重構空間：CreateService 重構空間：DeleteApplication 重構空間：DeleteEnvironment 重構空間：政策 DeleteResource 重構空間：DeleteRoute 重構空間：DeleteService 重構空間：GetApplication 重構空間：GetEnvironment 重構空間：政策 GetResource 重構空間：GetRoute 重構空間：GetService 重構空間：ListApplications 重構空間：ListEnvironments 重構空間：Vpcs ListEnvironment 重構空間：ListRoutes 重構空間：ListServices 重構空間：政策 PutResource 重構空間：UpdateRoute

服務前綴	動作
rekognition	重新認知：AssociateFaces
	重新認知：CompareFaces
	重新認知:版本 CopyProject
	重新認知：CreateCollection
	重新認知：CreateDataset
	重新認知：CreateFaceLivenessSession
	重新認知：CreateProject
	重新認知:版本 CreateProject
	重新認知:處理器 CreateStream
	重新認知：CreateUser
	重新認知：DeleteCollection
	重新認知：DeleteDataset
	重新認知：DeleteFaces
	重新認知：DeleteProject
	重新認知：政策 DeleteProject
	重新認知:版本 DeleteProject
	重新認知:處理器 DeleteStream
	重新認知：DeleteUser
	重新認知：DescribeCollection
	重新認知：DescribeDataset
	重新認知：DescribeProjects

服務前綴	動作
	重新認知：版本 DescribeProject
	重新認知:處理器 DescribeStream
	重新認知：標籤 DetectCustom
	重新認知：DetectFaces
	重新認知：DetectLabels
	重新認知：標籤 DetectModeration
	重新認知:設備 DetectProtective
	重新認知：DetectText
	重新認知：DisassociateFaces
	重新認知:項目 DistributeDataset
	重新認知：資訊 GetCelebrity
	重新認知:識別 GetCelebrity
	重新認知:適度 GetContent
	重新認知:偵測 GetFace
	重新認知：結果 GetFace LivenessSession
	重新認知:搜尋 GetFace
	重新認知:偵測 GetLabel
	重新認知：GetMediaAnalysisJob
	重新認知：追蹤 GetPerson
	重新認知:偵測 GetSegment
	重新認知:偵測 GetText

服務前綴	動作
	重新認知：IndexFaces
	重新認知：ListCollections
	重新認知:項目 ListDataset
	重新認知：標籤 ListDataset
	重新認知：ListFaces
	重新認知：ListMediaAnalysisJobs
	重新認知：政策 ListProject
	重新認知:處理器 ListStream
	重新認知：ListUsers
	重新認知：政策 PutProject
	重新認知：RecognizeCelebrities
	重新認知：SearchFaces
	重新認知：SearchFacesByImage
	重新認知：SearchUsers
	重新認知：SearchUsersByImage
	重新認知:識別 StartCelebrity
	重新認知:適度 StartContent
	重新認知:偵測 StartFace
	重新認知：StartFaceLivenessSession
	重新認知:搜尋 StartFace
	重新認知:偵測 StartLabel

服務前綴	動作
	重新認知：StartMediaAnalysisJob
	重新認知：追蹤 StartPerson
	重新認知:版本 StartProject
	重新認知:偵測 StartSegment
	重新認知:處理器 StartStream
	重新認知:偵測 StartText
	重新認知:版本 StopProject
	重新認知:處理器 StopStream
	重新認知:項目 UpdateDataset
	重新認知:處理器 UpdateStream

服務前綴	動作
resiliencehub	恢復中心：AddDraftAppVersionResourceMappings 恢復中心：CreateApp 恢復中心:組件 CreateApp VersionApp 恢復中心：CreateAppVersionResource 恢復中心:模板 CreateRecommendation 恢復中心:政策 CreateResiliency 恢復中心：DeleteApp 復原中心:評估 DeleteApp 恢復中心：DeleteAppInputSource 恢復中心:組件 DeleteApp VersionApp 恢復中心：DeleteAppVersionResource 恢復中心:模板 DeleteRecommendation 恢復中心:政策 DeleteResiliency 恢復中心：DescribeApp 復原中心:評估 DescribeApp 恢復中心:版本 DescribeApp 恢復中心:組件 DescribeApp VersionApp 恢復中心：DescribeAppVersionResource 恢復中心：DescribeAppVersionResourcesResolutionStatus 恢復中心：DescribeAppVersionTemplate 恢復中心：狀態 DescribeDraft AppVersion ResourcesImport

服務前綴	動作
	恢復中心:政策 DescribeResiliency 恢復中心：ImportResourcesToDraftAppVersion 恢復中心：建議 ListAlarm 復原中心：評估 ListApp 恢復中心：ListAppComponentCompliances 恢復中心：ListAppComponentRecommendations 恢復中心：ListAppInputSources 恢復中心：ListApps 恢復中心:組件 ListApp VersionApp 復原中心:對應 ListApp VersionResource 恢復中心：ListAppVersionResources 恢復中心:版本 ListApp 恢復中心：模板 ListRecommendation 恢復中心:政策 ListResiliency 恢復中心：建議 ListSop 恢復中心：ListSuggestedResiliencyPolicies 恢復中心：建議 ListTest 彈性中心：資源 ListUnsupported AppVersion 恢復中心:版本 PublishApp 恢復中心:模板 PutDraft AppVersion 恢復中心：RemoveDraftAppVersionResourceMappings

服務前綴	動作
	恢復中心：ResolveAppVersionResources 復原中心:評估 StartApp 恢復中心：UpdateApp 恢復中心:版本 UpdateApp 恢復中心:組件 UpdateApp VersionApp 恢復中心：UpdateAppVersionResource 恢復中心:政策 UpdateResiliency

服務前綴	動作
resource-explorer-2	資源瀏覽器 -2：檢視 AssociateDefault
	資源瀏覽器 -2：檢視 BatchGet
	資源瀏覽器 -2：CreateIndex
	資源瀏覽器 -2：CreateView
	資源瀏覽器 -2：DeleteIndex
	資源瀏覽器 -2：DeleteView
	資源瀏覽器 -2：檢視 DisassociateDefault
	資源瀏覽器 -2：配置 GetAccount LevelService
	資源瀏覽器 -2：檢視 GetDefault
	資源瀏覽器 -2：GetIndex
	資源瀏覽器 -2：ListIndexes
	資源瀏覽器 -2：ListIndexesForMembers
	資源瀏覽器 -2：ListSupportedResourceTypes
	資源瀏覽器 -2：ListViews
	resource-explorer-2:Search
	資源瀏覽器 -2：類型 UpdateIndex
	資源瀏覽器 -2：UpdateView

服務前綴	動作
resource-groups	資源群組：CreateGroup 資源群組：DeleteGroup 資源群組:設定 GetAccount 資源群組：GetGroup 資源群組:配置 GetGroup 資源群組:查詢 GetGroup 資源群組：GroupResources 資源群組：資源 ListGroup 資源群組：ListGroups 資源群組:配置 PutGroup 資源群組：SearchResources 資源群組：UngroupResources 資源群組:設定 UpdateAccount 資源群組：UpdateGroup 資源群組:查詢 UpdateGroup

服務前綴	動作
robomaker	機器人製造商：世界 BatchDelete
	機器人製造商：BatchDescribeSimulationJob
	機器人製造商：Job CancelDeployment
	機器人製造商：Job CancelSimulation
	機器人製造商：CancelSimulationJobBatch
	機器人製造商：CancelWorldExportJob
	機器人製造商：CancelWorldGenerationJob
	機器人製造商：Job CreateDeployment
	機器人製造商：CreateFleet
	機器人製造商：CreateRobot
	機器人製造商：CreateRobot應用
	機器人製造商：CreateRobotApplicationVersion
	機器人製造商：CreateSimulation應用
	機器人製造商：CreateSimulationApplicationVersion
	機器人製造商：Job CreateSimulation
	機器人製造商：CreateWorldExportJob
	機器人製造商：CreateWorldGenerationJob
	機器人製造商:CreateWorld模板
	機器人製造商：DeleteFleet
	機器人製造商：DeleteRobot
	機器人製造商：DeleteRobot應用

服務前綴	動作
	機器人製造商：DeleteSimulation應用
	機器人製造商:DeleteWorld模板
	機器人製造商：DeregisterRobot
	機器人製造商：Job DescribeDeployment
	機器人製造商：DescribeFleet
	機器人製造商：DescribeRobot
	機器人製造商：DescribeRobot應用
	機器人製造商：DescribeSimulation應用
	機器人製造商：Job DescribeSimulation
	機器人製造商：DescribeSimulationJobBatch
	機器人製造商：DescribeWorld
	機器人製造商：DescribeWorldExportJob
	機器人製造商：DescribeWorldGenerationJob
	機器人製造商:DescribeWorld模板
	機器人製造商：GetWorldTemplateBody
	機器人製造商:ListDeployment工作
	機器人製造商：ListFleets
	機器人製造商：ListRobot應用
	機器人製造商：ListRobots
	機器人製造商：ListSimulation應用
	機器人製造商：ListSimulationJobBatches

服務前綴	動作
	機器人製造商:ListSimulation工作 機器人製造商：ListWorldExportJobs 機器人製造商：ListWorldGenerationJobs 機器人製造商：ListWorlds 機器人製造商：ListWorld模板 機器人製造商：RegisterRobot 機器人製造商：Job RestartSimulation 機器人製造商：StartSimulationJobBatch 機器人製造商：Job SyncDeployment 機器人製造商：UpdateRobot應用 機器人製造商：UpdateSimulation應用 機器人製造商:UpdateWorld模板

服務前綴	動作
rolesanywhere	角色任何地方：CreateProfile 角色任意位置:錨 CreateTrust 角色任何地方:映射 DeleteAttribute 角色任何地方：DeleteCrl 角色任何地方：DeleteProfile 角色任意位置:錨 DeleteTrust 角色任何地方：DisableCrl 角色任何地方：DisableProfile 角色任意位置:錨 DisableTrust 角色任何地方：EnableCrl 角色任何地方：EnableProfile 角色任意位置:錨 EnableTrust 角色任何地方：GetCrl 角色任何地方：GetProfile 角色任何地方：GetSubject 角色任意位置:錨 GetTrust 角色任何地方：ImportCrl 角色任何地方：ListCrls 角色任何地方：ListProfiles 角色任何地方：ListSubjects 角色任何地方：錨 ListTrust

服務前綴	動作
	角色任何地方:映射 PutAttribute 角色任意位置:設定 PutNotification 角色任意位置:設定 ResetNotification 角色任何地方：UpdateCrl 角色任何地方：UpdateProfile 角色任意位置:錨 UpdateTrust

服務前綴	動作
route53	路線 53: ActivateKey SigningKey
	路線 53: 協會WithHosted專區
	路線 53: 收藏 ChangeCidr
	路線 53: ChangeResource RecordSets
	路線 53: 收藏 CreateCidr
	路線 53: 檢查 CreateHealth
	路線 53: 區域 CreateHosted
	路線 53: CreateKey SigningKey
	路線 53: CreateQuery LoggingConfig
	路線 53: CreateReusable DelegationSet
	路線 53：政策 CreateTraffic
	路線 53: CreateTraffic PolicyInstance
	路線 53: CreateTraffic PolicyVersion
	路線 53: 創建 VPC AssociationAuthorization
	路線 53: DeactivateKey SigningKey
	路線 53: 收藏 DeleteCidr
	路線 53: 檢查 DeleteHealth
	路線 53: 區域 DeleteHosted
	路線 53: DeleteKey SigningKey
	路線 53: DeleteQuery LoggingConfig
	路線 53: DeleteReusable DelegationSet

服務前綴	動作
	路線 53：政策 DeleteTraffic
	路線 53: DeleteTraffic PolicyInstance
	路線 53：刪除 VPC AssociationAuthorization
	路線 53: 分區安全委員會 DisableHosted
	路線 53：解除FromHosted關聯 VPC 專區
	路線 53: 分區安全委員會 EnableHosted
	路線 53: 限制 GetAccount
	路線 53: GetChange
	路線 53: GetChecker IpRanges
	route53:GetDNSSEC
	路線 53: 位置 GetGeo
	路線 53: 檢查 GetHealth
	路線 53: GetHealth CheckCount
	路線 53: GetHealth CheckLast FailureReason
	路線 53: GetHealth CheckStatus
	路線 53: 區域 GetHosted
	路線 53: GetHosted ZoneCount
	路線 53: GetHosted ZoneLimit
	路線 53: GetQuery LoggingConfig
	路線 53: GetReusable DelegationSet
	路線 53: 限制 GetReusable DelegationSet

服務前綴	動作
	路線 53：政策 GetTraffic
	路線 53: GetTraffic PolicyInstance
	路線 53: 計數 GetTraffic PolicyInstance
	路線 53：塊 ListCidr
	路線 53: 收藏 ListCidr
	路線 53: 位置 ListCidr
	路線 53: 位置 ListGeo
	路線 53: 檢查 ListHealth
	路線 53: 區域 ListHosted
	路線 53: 名稱 ListHosted ZonesBy
	路線 53: VPC ListHosted ZonesBy
	路線 53: ListQuery LoggingConfigs
	路線 53: ListResource RecordSets
	路線 53: ListReusable DelegationSets
	路線 53: 政策 ListTraffic
	路線 53: ListTraffic PolicyInstances
	路線 53: 區域 ListTraffic PolicyInstances ByHosted
	路線 53: ListTraffic PolicyInstances ByPolicy
	路線 53: ListTraffic PolicyVersions
	路線 53：列表 VPC AssociationAuthorizations
	route53:TestDNSAnswer

服務前綴	動作
	路線 53: 檢查 UpdateHealth 路線 53: UpdateHosted ZoneComment 路線 53: UpdateTraffic PolicyComment 路線 53: UpdateTraffic PolicyInstance

服務前綴	動作
route53-recovery-control-config	路由 53 恢復控制配置：CreateCluster 路由 53 恢復控制配置：面板 CreateControl 路由 53 恢復控制配置：控制 CreateRouting 路由 53 恢復控制配置：規則 CreateSafety 路由 53 恢復控制配置：DeleteCluster 路由 53 恢復控制配置：面板 DeleteControl 路由 53 恢復控制配置：控制 DeleteRouting 路由 53 恢復控制配置：規則 DeleteSafety 路由 53 恢復控制配置：DescribeCluster 路由 53 恢復控制配置：面板 DescribeControl 路由 53 恢復控制配置：控制 DescribeRouting 路由 53 恢復控制配置：規則 DescribeSafety 路由 53 恢復控制配置：策略 GetResource 路由 53 恢復控制配置：路由 53 ListAssociated HealthChecks 路由 53 恢復控制配置：ListClusters 路由 53 恢復控制配置：面板 ListControl 路由 53 恢復控制配置：控制 ListRouting 路由 53 恢復控制配置：規則 ListSafety 路由 53 恢復控制配置：面板 UpdateControl 路由 53 恢復控制配置：控制 UpdateRouting 路由 53 恢復控制配置：規則 UpdateSafety

服務前綴	動作
route53-recovery-readiness	路由 53-恢復準備：CreateCell
	路由 53-恢復準備：CreateCrossAccountAuthorization
	路由 53-恢復準備：檢查 CreateReadiness
	路由 53-恢復準備：組 CreateRecovery
	路由 53-恢復準備：設置 CreateResource
	路由 53-恢復準備：DeleteCell
	路由 53-恢復準備：DeleteCrossAccountAuthorization
	路由 53-恢復準備：檢查 DeleteReadiness
	路由 53-恢復準備：組 DeleteRecovery
	路由 53-恢復準備：設置 DeleteResource
	路由 53-恢復準備：建議 GetArchitecture
	路由 53-恢復準備：GetCell
	路由 53-恢復準備：GetCellReadinessSummary
	路由 53-恢復準備：檢查 GetReadiness
	路由 53-恢復準備：狀態 GetReadiness CheckResource
	路由 53-恢復準備：GetReadinessCheckStatus
	路由 53-恢復準備：組 GetRecovery
	路由 53-恢復準備：摘要 GetRecovery GroupReadiness
	路由 53-恢復準備：設置 GetResource
	路由 53-恢復準備：ListCells
	路由 53-恢復準備：ListCrossAccountAuthorizations

服務前綴	動作
	路由 53-恢復準備：檢查 ListReadiness
	路由 53-恢復準備：組 ListRecovery
	路由 53-恢復準備：設置 ListResource
	路由 53-恢復準備：ListRules
	路由 53-恢復準備：UpdateCell
	路由 53-恢復準備：檢查 UpdateReadiness
	路由 53-恢復準備：組 UpdateRecovery
	路由 53-恢復準備：設置 UpdateResource

服務前綴	動作
route53resolver	路由 53 解析器：AssociateFirewallRuleGroup 路由 53 解析器：地址 AssociateResolver EndpointIp 路由 53 解析器：Config AssociateResolver QueryLog 路由 53 解析器：規則 AssociateResolver 路由 53 解析器：CreateFirewallDomainList 路由 53 解析器：規則 CreateFirewall 路由 53 解析器：CreateFirewallRuleGroup 路由 53 解析器：端點 CreateResolver 路由 53 解析器：Config CreateResolver QueryLog 路由 53 解析器：規則 CreateResolver 路由 53 解析器：DeleteFirewallDomainList 路由 53 解析器：規則 DeleteFirewall 路由 53 解析器：DeleteFirewallRuleGroup 路由 53 解析器:解析器 DeleteOutpost 路由 53 解析器：端點 DeleteResolver 路由 53 解析器：Config DeleteResolver QueryLog 路由 53 解析器：規則 DeleteResolver 路由 53 解析器：DisassociateFirewallRuleGroup 路由 53 解析器：地址 DisassociateResolver EndpointIp 路由 53 解析器：Config DisassociateResolver QueryLog 路由 53 解析器：規則 DisassociateResolver

服務前綴	動作
	路由 53 解析器：Config GetFirewall
	路由 53 解析器：GetFirewallDomainList
	路由 53 解析器：GetFirewallRuleGroup
	路由 53 解析器：協會 GetFirewall RuleGroup
	路由 53 解析器：政策 GetFirewall RuleGroup
	路由 53 解析器:解析器 GetOutpost
	路由 53 解析器：Config GetResolver
	路由 53 解析器：GetResolverDnssecConfig
	路由 53 解析器：端點 GetResolver
	路由 53 解析器：Config GetResolver QueryLog
	路由 53 解析器：GetResolverQueryLogConfigAssociation
	路由 53 解析器：GetResolverQueryLogConfigPolicy
	路由 53 解析器：規則 GetResolver
	路由 53 解析器：GetResolverRuleAssociation
	路由 53 解析器：GetResolverRulePolicy
	路由 53 解析器：域 ImportFirewall
	路由 53 解析器：配置 ListFirewall
	路由 53 解析器：ListFirewallDomainLists
	路由 53 解析器：域 ListFirewall
	路由 53 解析器：關聯 ListFirewall RuleGroup
	路由 53 解析器：ListFirewallRuleGroups

服務前綴	動作
	路由 53 解析器：規則 ListFirewall 路由 53 解析器:解析器 ListOutpost 路由 53 解析器：配置 ListResolver 路由 53 解析器：ListResolverDnssecConfigs 路由 53 解析器：地址 ListResolver EndpointIp 路由 53 解析器：端點 ListResolver 路由 53 解析器：ListResolverQueryLogConfigAssociations 路由 53 解析器：配置 ListResolver QueryLog 路由 53 解析器：ListResolverRuleAssociations 路由 53 解析器：規則 ListResolver 路由 53 解析器：政策 PutFirewall RuleGroup 路由 53 解析器：PutResolverQueryLogConfigPolicy 路由 53 解析器：Config UpdateFirewall 路由 53 解析器：域 UpdateFirewall 路由 53 解析器：規則 UpdateFirewall 路由 53 解析器：協會 UpdateFirewall RuleGroup 路由 53 解析器:解析器 UpdateOutpost 路由 53 解析器：Config UpdateResolver 路由 53 解析器：UpdateResolverDnssecConfig 路由 53 解析器：端點 UpdateResolver 路由 53 解析器：規則 UpdateResolver

服務前綴	動作
rum	朗姆酒：BatchCreateRumMetric定義 朗姆酒：BatchDeleteRumMetric定義 朗姆酒：BatchGetRumMetric定義 朗姆酒:CreateApp監視器 朗姆酒:DeleteApp監視器 朗姆酒：DeleteRumMetricsDestination 朗姆酒:GetApp監視器 朗姆酒：GetAppMonitorData 朗姆酒：ListApp監視器 朗姆酒：ListRumMetricsDestinations 朗姆酒：PutRumMetricsDestination 朗姆酒:UpdateApp監視器 朗姆酒：UpdateRumMetricDefinition

服務前綴	動作
s3	S3：AssociateAccessGrantsIdentity中心 S3：CreateAccess授予 S3：CreateAccessGrantsInstance S3：CreateAccessGrantsLocation S3：CreateAccess點 S3：CreateAccessPointForObjectLambda S3：CreateBucket S3：CreateJob S3：CreateMultiRegionAccess點 S3：DeleteAccess授予 S3：DeleteAccessGrantsInstance S3：DeleteAccessGrantsInstanceResourcePolicy S3：DeleteAccessGrantsLocation S3：DeleteAccess點 S3：DeleteAccessPointForObjectLambda S3：DeleteAccessPointPolicy S3：DeleteAccessPointPolicyForObjectLambda S3：PutAccountPublicAccess塊 S3：DeleteBucket S3：PutAnalytics配置 S3：PutBucket科爾斯

服務前綴	動作
	S3：PutEncryption配置 S3：PutIntelligentTieringConfiguration S3：PutInventory配置 S3：PutLifecycle配置 S3：PutMetrics配置 S3：PutBucketOwnershipControls S3：DeleteBucket政策 S3：PutBucketPublicAccess塊 S3：PutReplication配置 s3：DeleteBucket網站 S3：DeleteMultiRegionAccess點 S3：DeleteStorageLensConfiguration S3：DescribeJob S3：DescribeMultiRegionAccessPointOperation S3：DissociateAccessGrantsIdentity中心 S3：GetAccelerate配置 S3：GetAccess授予 S3：GetAccessGrantsInstance S3：GetAccessGrantsInstanceForPrefix S3：GetAccessGrantsInstanceResourcePolicy S3：GetAccessGrantsLocation

服務前綴	動作
	S3：GetAccess點 S3：GetAccessPointConfigurationForObjectLambda S3：GetAccessPointForObjectLambda S3：GetAccessPointPolicy S3：GetAccessPointPolicyForObjectLambda S3：GetAccessPointPolicy狀態 S3：GetAccessPointPolicyStatusForObjectLambda S3：GetAccountPublicAccess塊 S3：GetBucket十字韌帶 S3：GetAnalytics配置 S3：GetBucket科爾斯 S3：GetEncryption配置 S3：GetIntelligentTieringConfiguration S3：GetInventory配置 S3：GetLifecycle配置 S3：GetBucket位置 S3：GetBucket日誌記錄 S3：GetMetrics配置 s3：GetBucket通知 S3：GetBucketObjectLock配置 S3：GetBucketOwnershipControls

服務前綴	動作
	S3：GetBucket政策 S3：GetBucketPolicyStatus S3：GetBucketPublicAccess塊 S3：GetReplication配置 S3：GetBucketRequestPayment s3：GetBucket版本控制 s3：GetBucket網站 S3：GetData訪問 S3：GetMultiRegionAccess點 S3：GetMultiRegionAccessPointPolicy S3：GetMultiRegionAccessPointPolicy狀態 S3：GetMultiRegionAccessPointRoutes S3：GetObject屬性 S3：GetStorageLensConfiguration S3：GetStorageLensDashboard S3：ListAccess贈款 S3：ListAccessGrantsInstances S3：ListAccessGrantsLocations S3：ListAccess點 S3：ListAccessPointsForObjectLambda S3：ListAllMyBuckets

服務前綴	動作
	S3：ListJobs S3：ListBucketMultipartUploads S3：ListMultiRegionAccess點 S3：ListStorageLensConfigurations S3：PutAccelerate配置 S3：PutAccessGrantsInstanceResourcePolicy S3：PutAccessPointConfigurationForObjectLambda S3：PutAccessPointPolicy S3：PutAccessPointPolicyForObjectLambda S3：PutAccountPublicAccess塊 S3：PutBucket十字韌帶 S3：PutAnalytics配置 S3：PutBucket科爾斯 S3：PutEncryption配置 S3：PutIntelligentTieringConfiguration S3：PutInventory配置 S3：PutLifecycle配置 S3：PutBucket日誌記錄 S3：PutMetrics配置 s3：PutBucket通知 S3：PutBucketObjectLock配置

服務前綴	動作
	S3：PutBucketOwnershipControls S3：PutBucket政策 S3：PutBucketPublicAccess塊 S3：PutReplication配置 S3：PutBucketRequestPayment s3：PutBucket版本控制 s3：PutBucket網站 S3：PutMultiRegionAccessPointPolicy S3：PutStorageLensConfiguration S3：SubmitMultiRegionAccessPointRoutes S3：UpdateAccessGrantsLocation S3：UpdateJob優先順序 S3：UpdateJob狀態
s3-outposts	S3-前哨站：CreateEndpoint S3-前哨站：DeleteEndpoint S3-前哨站：ListEndpoints S3-前哨：與 S3 ListOutposts S3-前哨：端點 ListShared

服務前綴	動作
sagemaker-geospatial	圖形機-地理空間：DeleteEarthObservationJob 圖形機-地理空間：DeleteVectorEnrichmentJob 圖形機-地理空間：ExportEarthObservationJob 圖形機-地理空間：ExportVectorEnrichmentJob 圖形機-地理空間：GetEarthObservationJob 圖形機-地理空間：GetRasterDataCollection 圖形機-地理空間：GetTile 圖形機-地理空間：GetVectorEnrichmentJob 圖形機-地理空間：ListEarthObservationJobs 圖形機-地理空間：ListRasterDataCollections 圖形機-地理空間：ListVectorEnrichmentJobs 圖形機-地理空間：SearchRasterDataCollection 圖形機-地理空間：StartEarthObservationJob 圖形機-地理空間：StartVectorEnrichmentJob 圖形機-地理空間：StopEarthObservationJob 圖形機-地理空間：StopVectorEnrichmentJob

服務前綴	動作
savingsplans	儲蓄者：計劃 CreateSavings 儲蓄者：DeleteQueuedSavingsPlan 儲蓄者：DescribeSavingsPlanRates 儲蓄者：計劃 DescribeSavings 儲蓄者：價格 DescribeSavings PlansOffering 儲蓄者：DescribeSavingsPlansOfferings 儲蓄者：計劃 ReturnSavings

服務前綴	動作
schemas	綱要:CreateDiscoverer 綱要:CreateRegistry 綱要:CreateSchema 綱要:DeleteDiscoverer 綱要:DeleteRegistry 綱要:DeleteResource政策 綱要:DeleteSchema 綱要:DeleteSchema版本 綱要:DescribeCode繫結 綱要:DescribeDiscoverer 綱要:DescribeRegistry 綱要:DescribeSchema 綱要:ExportSchema 綱要:GetCodeBindingSource 綱要:GetDiscovered綱要 綱要:GetResource政策 綱要:ListDiscoverers 綱要:ListRegistries 綱要:ListSchemas 綱要:ListSchema版本 綱要:PutCode繫結

服務前綴	動作
	綱要:PutResource政策 綱要:SearchSchemas 綱要:StartDiscoverer 綱要:StopDiscoverer 綱要:UpdateDiscoverer 綱要:UpdateRegistry 綱要:UpdateSchema
sdb	SDB：CreateDomain SDB：DeleteDomain SDB：DomainMetadata SDB：ListDomains

服務前綴	動作
secretsmanager	秘密經理:秘密 CancelRotate 秘書經理:CreateSecret 秘密經理:政策 DeleteResource 秘書經理:DeleteSecret 秘書經理:DescribeSecret 秘密經理:密碼 GetRandom 秘密經理:政策 GetResource 秘密經理:價值 GetSecret 秘書經理:ListSecrets 秘書經理:ListSecretVersionIds 秘密經理:政策 PutResource 秘密經理:價值 PutSecret 秘書經理:RemoveRegionsFromReplication 秘書經理:ReplicateSecretToRegions 秘書經理:RestoreSecret 秘書經理:RotateSecret 秘書經理:StopReplicationToReplica 秘書經理:UpdateSecret 秘密經理:政策 ValidateResource

服務前綴	動作
securityhub	安全中心:邀請 AcceptAdministrator
	安全中心：AcceptInvitation
	安全中心：BatchDeleteAutomationRules
	安全中心：標準 BatchDisable
	安全中心：標準 BatchEnable
	安全中心：BatchGetAutomationRules
	安全中心:關聯 BatchGet ConfigurationPolicy
	安全中心：BatchGetSecurityControls
	安全中心:關聯 BatchGet StandardsControl
	安全中心：發現項目 BatchImport
	安全中心：BatchUpdateAutomationRules
	安全中心：發現項目 BatchUpdate
	安全中心:關聯 BatchUpdate StandardsControl
	安全中心：目標 CreateAction
	安全中心：規則 CreateAutomation
	安全中心：原則 CreateConfiguration
	安全中心：聚合器 CreateFinding
	安全中心：CreateInsight
	安全中心：CreateMembers
	安全中心：DeclineInvitations
	安全中心：目標 DeleteAction

服務前綴	動作
	安全中心：原則 DeleteConfiguration
	安全中心：聚合器 DeleteFinding
	安全中心：DeleteInsight
	安全中心：DeleteInvitations
	安全中心：DeleteMembers
	安全中心:目標 DescribeAction
	安全中心：DescribeHub
	安全中心：組態 DescribeOrganization
	安全中心：DescribeProducts
	安全中心：DescribeStandards
	安全中心:產品 DisableImport FindingsFor
	安全中心：DisableOrganizationAdminAccount
	安全中心：集線器 DisableSecurity
	安全中心：DisassociateFromAdministratorAccount
	安全中心：DisassociateFromMasterAccount
	安全中心：DisassociateMembers
	安全中心:產品 EnableImport FindingsFor
	安全中心：EnableOrganizationAdminAccount
	安全中心：集線器 EnableSecurity
	安全中心：帳戶 GetAdministrator
	安全中心：原則 GetConfiguration

服務前綴	動作
	安全中心：GetConfigurationPolicyAssociation
	安全中心：標準 GetEnabled
	安全中心：聚合器 GetFinding
	安全中心：歷史記錄 GetFinding
	安全中心：GetFindings
	安全中心：結果 GetInsight
	安全中心：GetInsights
	安全中心：計數 GetInvitations
	安全中心：帳戶 GetMaster
	安全中心：GetMembers
	安全中心：GetSecurityControlDefinition
	安全中心：InviteMembers
	安全中心：規則 ListAutomation
	安全中心：原則 ListConfiguration
	安全中心：ListConfigurationPolicyAssociations
	安全中心：匯入 ListEnabled ProductsFor
	安全中心：聚合器 ListFinding
	安全中心：ListInvitations
	安全中心：ListMembers
	安全中心：ListOrganizationAdminAccounts
	安全中心：ListSecurityControlDefinitions

服務前綴	動作
	安全中心：ListStandardsControlAssociations
	安全中心：StartConfigurationPolicyAssociation
	安全中心：StartConfigurationPolicyDisassociation
	安全中心：目標 UpdateAction
	安全中心：原則 UpdateConfiguration
	安全中心：聚合器 UpdateFinding
	安全中心：UpdateFindings
	安全中心：UpdateInsight
	安全中心：組態 UpdateOrganization
	安全中心:控制 UpdateSecurity
	安全中心：UpdateSecurityHubConfiguration

服務前綴	動作
securitylake	安全湖：CreateAwsLogSource 安全湖：CreateCustomLogSource 安全湖泊：訂閱 CreateData LakeException 安全湖泊:組態 CreateData LakeOrganization 安全湖：CreateSubscriber 安全湖:通知 CreateSubscriber 安全湖：DeleteAwsLogSource 安全湖：DeleteCustomLogSource 安全湖泊：訂閱 DeleteData LakeException 安全湖泊:組態 DeleteData LakeOrganization 安全湖：DeleteSubscriber 安全湖:通知 DeleteSubscriber 安全湖泊:管理員 DeregisterData LakeDelegated 安全湖泊：訂閱 GetData LakeException 安全湖泊:組態 GetData LakeOrganization 安全湖：GetDataLakeSources 安全湖：GetSubscriber 安全湖：湖泊 ListData 安全湖：來源 ListLog 安全湖：ListSubscribers 安全湖泊:管理員 RegisterData LakeDelegated

服務前綴	動作
	安全湖泊：訂閱 UpdateData LakeException 安全湖：UpdateSubscriber 安全湖:通知 UpdateSubscriber
serverlessrepo	無伺服器回購：CreateApplication 無伺服器回購：版本 CreateApplication 無伺服器回購：設定 CreateCloud FormationChange 無伺服器回購：CreateCloudFormationTemplate 無伺服器回購：DeleteApplication 無伺服器回購：GetApplication 無伺服器回購：政策 GetApplication 無伺服器回購：GetCloudFormationTemplate 無伺服器回購：相依性 ListApplication 無伺服器回購：ListApplications 無伺服器回購：版本 ListApplication 無伺服器回購：政策 PutApplication 無伺服器回購：UnshareApplication 無伺服器回購：UpdateApplication

服務前綴	動作
servicecatalog	服務目錄：分享 AcceptPortfolio 服務目錄：AssociateBudgetWithResource 服務目錄：AssociatePrincipalWithPortfolio 服務目錄：AssociateProductWithPortfolio 服務目錄：AssociateServiceActionWithProvisioningArtifact 服務目錄：Artifact BatchAssociate ServiceAction WithProvisioning 服務目錄：Artifact BatchDisassociate ServiceAction FromProvisioning 服務目錄：CopyProduct 服務目錄：CreateConstraint 服務目錄：CreatePortfolio 服務目錄：分享 CreatePortfolio 服務目錄：CreateProduct 服務目錄：CreateProvisionedProductPlan 服務目錄：Artifact CreateProvisioning 服務目錄：動作 CreateService 服務目錄：DeleteConstraint 服務目錄：DeletePortfolio 服務目錄：分享 DeletePortfolio 服務目錄：DeleteProduct 服務目錄：DeleteProvisionedProductPlan

服務前綴	動作
	服務目錄：Artifact DeleteProvisioning 服務目錄：動作 DeleteService 服務目錄：DescribeConstraint 服務目錄：DescribeCopyProductStatus 服務目錄：DescribePortfolio 服務目錄：股票 DescribePortfolio 服務目錄：DescribePortfolioShareStatus 服務目錄：DescribeProduct 服務目錄：DescribeProductAsAdmin 服務目錄：檢視 DescribeProduct 服務目錄：DescribeProvisionedProductPlan 服務目錄：Artifact DescribeProvisioning 服務目錄：參數 DescribeProvisioning 服務目錄：DescribeRecord 服務目錄：動作 DescribeService 服務目錄：參數 DescribeService ActionExecution 服務目錄:停用 AWSOrganizationsAccess 服務目錄：DisassociateBudgetFromResource 服務目錄：DisassociatePrincipalFromPortfolio 服務目錄：DisassociateProductFromPortfolio 服務目錄：DisassociateServiceActionFromProvisioningArtifact

服務前綴	動作
	服務目錄:啟用 AWSOrganizationsAccess 服務目錄：ExecuteProvisionedProductPlan 服務目錄：動作 ExecuteProvisioned ProductService 服務目錄:取得 AWSOrganizationsAccessStatus 服務目錄：GetProvisionedProductOutputs 服務目錄：ImportAsProvisionedProduct 服務目錄：ListAcceptedPortfolioShares 服務目錄：ListBudgetsForResource 服務目錄：ListConstraintsForPortfolio 服務目錄：路徑 ListLaunch 服務目錄：ListOrganizationPortfolioAccess 服務目錄:存取 ListPortfolio 服務目錄：ListPortfolios 服務目錄：ListPortfoliosForProduct 服務目錄：ListPrincipalsForPortfolio 服務目錄：ListProvisionedProductPlans 服務目錄:工件 ListProvisioning 服務目錄：ListProvisioningArtifactsForServiceAction 服務目錄：歷史 ListRecord 服務目錄：動作 ListService 服務目錄：ListServiceActionsForProvisioningArtifact

服務前綴	動作
	服務目錄：ListStackInstancesForProvisionedProduct 服務目錄：NotifyProvisionProductEngineWorkflowResult 服務目錄：結果 NotifyTerminate ProvisionedProduct EngineWorkflow 服務目錄：結果 NotifyUpdate ProvisionedProduct EngineWorkflow 服務目錄：ProvisionProduct 服務目錄：分享 RejectPortfolio 服務目錄：產品 ScanProvisioned 服務目錄：SearchProducts 服務目錄：SearchProductsAsAdmin 服務目錄：產品 SearchProvisioned 服務目錄：產品 TerminateProvisioned 服務目錄：UpdateConstraint 服務目錄：UpdatePortfolio 服務目錄：分享 UpdatePortfolio 服務目錄：UpdateProduct 服務目錄：產品 UpdateProvisioned 服務目錄：UpdateProvisionedProductProperties 服務目錄：Artifact UpdateProvisioning 服務目錄：動作 UpdateService

服務前綴	動作
servicediscovery	服務探索：CreateHttp命名空間
	服務探索：CreatePrivateDnsNamespace
	服務探索：CreatePublicDnsNamespace
	服務探索：CreateService
	服務探索：DeleteNamespace
	服務探索：DeleteService
	服務探索：DeregisterInstance
	服務探索：GetInstance
	服務探索：GetInstancesHealthStatus
	服務探索：GetNamespace
	服務探索：GetOperation
	服務探索：GetService
	服務探索：ListInstances
	服務探索：ListNamespaces
	服務探索：ListOperations
	服務探索：ListServices
	服務探索：RegisterInstance
	服務探索：UpdateHttp命名空間
	服務探索：狀態 UpdateInstance CustomHealth
	服務探索：UpdatePrivateDnsNamespace
	服務探索：UpdatePublicDnsNamespace

服務前綴	動作
	服務探索：UpdateService
servicequotas	服務：AssociateServiceQuotaTemplate 服務:模板 DeleteService QuotaIncrease RequestFrom 服務：DisassociateServiceQuotaTemplate 服務：GetAssociationForServiceQuotaTemplate 服務等級:取得 AWSDefaultServiceQuota 服務:變更 GetRequested ServiceQuota 服務:配額 GetService 服務:模板 GetService QuotaIncrease RequestFrom 服務等級:清單 AWSDefaultServiceQuotas 服務：ListRequestedServiceQuotaChangeHistory 服務：ListRequestedServiceQuotaChangeHistoryByQuota 服務:模板 ListService QuotaIncrease RequestsIn 服務:配額 ListService 服務：ListServices 服務:模板 PutService QuotaIncrease RequestInto 服務：RequestServiceQuotaIncrease

服務前綴	動作
ses	西斯:BatchGetMetricData
	西斯:CloneReceiptRuleSet
	SES: CreateConfiguration 設置
	SES: CreateConfiguration SetEvent 目的地
	SES: CreateConfiguration SetTracking 選項
	SES: CreateContact
	SES: CreateContact 清單
	SES: CreateCustom VerificationEmail 模板
	西斯:CreateDedicatedIpPool
	西斯:CreateDeliverabilityTestReport
	SES: CreateEmail 身份
	西斯:CreateEmailIdentityPolicy
	SES: CreateEmail 模板
	CreateImportJob
	ses: CreateReceipt 篩選
	SES: CreateReceipt 規則
	西斯:CreateReceiptRuleSet
	SES: CreateTemplate
	SES: DeleteConfiguration 設置
	SES: DeleteConfiguration SetEvent 目的地
	SES: DeleteConfiguration SetTracking 選項

服務前綴	動作
	SES: DeleteContact
	SES: DeleteContact 清單
	SES: DeleteCustom VerificationEmail 模板
	西斯:DeleteDedicatedIpPool
	SES: DeleteEmail 身份
	西斯:DeleteEmailIdentityPolicy
	SES: DeleteEmail 模板
	SES: DeleteIdentity
	DeleteIdentity策略
	ses: DeleteReceipt 篩選
	SES: DeleteReceipt 規則
	西斯:DeleteReceiptRuleSet
	SES: DeleteSuppressed 目的地
	SES: DeleteTemplate
	西斯:DeleteVerifiedEmailAddress
	SES: DescribeActive ReceiptRule 設置
	SES: DescribeConfiguration 設置
	SES: DescribeReceipt 規則
	西斯:DescribeReceiptRuleSet
	SES: GetAccount
	西斯:GetAccountSendingEnabled

服務前綴	動作
	SES: GetBlacklist 報告
	SES: GetConfiguration 設置
	SES: GetConfiguration SetEvent 目的地
	SES: GetContact
	SES: GetContact 清單
	SES: GetCustom VerificationEmail 模板
	西斯:GetDedicatedIP
	西斯:GetDedicatedIpPool
	西斯:GetDedicatedIps
	西斯:GetDeliverabilityDashboardOptions
	西斯:GetDeliverabilityTestReport
	西斯:GetDomainDeliverabilityCampaign
	西斯:GetDomainStatisticsReport
	SES: GetEmail 身份
	西斯:GetEmailIdentityPolicies
	SES: GetEmail 模板
	西斯:GetIdentityDkimAttributes
	SES: GetIdentity MailFrom DomainAttributes
	西斯:GetIdentityNotificationAttributes
	SES: GetIdentity 政策
	西斯:GetIdentityVerificationAttributes

服務前綴	動作
	GetImportJob
	SES: GetMessage 洞察
	GetSend配額
	SES: GetSend 統計
	SES: GetSuppressed 目的地
	SES: GetTemplate
	SES: ListConfiguration 套裝
	SES: ListContact 列表
	SES: ListContacts
	SES: ListCustom VerificationEmail 模板
	西斯:ListDedicatedIpPools
	西斯:ListDeliverabilityTestReports
	西斯:ListDomainDeliverabilityCampaigns
	SES: ListEmail 身份
	SES: ListEmail 模板
	ListExport工作
	SES: ListIdentities
	SES: ListIdentity 政策
	ListImport工作
	ses: ListReceipt 篩選條件
	西斯:ListReceiptRuleSets

服務前綴	動作
	西斯:ListRecommendations
	SES: ListSuppressed 目的地
	西斯:ListTemplates
	西斯:ListVerifiedEmailAddresses
	SES: PutAccount DedicatedIp WarmupAttributes
	ses: PutAccount 詳細資料
	西斯:PutAccountSendingAttributes
	西斯:PutAccountSuppressionAttributes
	西斯:PutAccountVdmAttributes
	SES: PutConfiguration SetDelivery 選項
	SES: PutConfiguration SetReputation 選項
	SES: PutConfiguration SetSending 選項
	SES: PutConfiguration SetSuppression 選項
	SES: PutConfiguration SetTracking 選項
	SES: PutConfiguration SetVdm 選項
	西斯:PutDedicatedIpIn游泳池
	SES: PutDedicated IpPool ScalingAttributes
	SES：PutDedicatedIpWarmup屬性
	西斯:PutDeliverabilityDashboardOption
	SES: PutEmail IdentityConfiguration SetAttributes
	SES：PutEmailIdentityDkim屬性

服務前綴	動作
	SES: PutEmail IdentityDkim SigningAttributes
	SES：PutEmailIdentityFeedback屬性
	SES: PutEmail IdentityMail FromAttributes
	PutIdentity策略
	SES: PutSuppressed 目的地
	西斯:ReorderReceiptRuleSet
	西斯:SendBounce
	西斯:SendCustomVerificationEmail
	SES: SetActive ReceiptRule 設置
	西斯:SetIdentityDkimEnabled
	SES: SetIdentity FeedbackForwarding 已啟用
	SES: SetIdentity HeadersIn NotificationsEnabled
	SES: SetIdentity MailFrom 網域名稱
	西斯:SetIdentityNotificationTopic
	西斯:SetReceiptRulePosition
	西斯:TestRenderEmailTemplate
	SES: TestRender 模板
	西斯:UpdateAccountSendingEnabled
	SES: UpdateConfiguration SetEvent 目的地
	SES: UpdateConfiguration SetReputation MetricsEnabled
	SES: UpdateConfiguration SetSending 已啟用

服務前綴	動作
	SES: UpdateConfiguration SetTracking 選項 西斯:UpdateContact SES: UpdateContact 清單 SES: UpdateCustom VerificationEmail 模板 西斯:UpdateEmailIdentityPolicy SES: UpdateEmail 模板 SES: UpdateReceipt 規則 西斯:UpdateTemplate 西斯VerifyDomain: SES: VerifyDomain 身份 SES: VerifyEmail 地址 SES: VerifyEmail 身份

服務前綴	動作
shield	屏蔽:關聯式 LogBucket 屏蔽：AssociateHealth檢查 屏蔽：AssociateProactiveEngagementDetails 屏蔽：CreateProtection 屏蔽：CreateProtection組 屏蔽：CreateSubscription 屏蔽：DeleteProtection 屏蔽：DeleteProtection組 屏蔽：DeleteSubscription 屏蔽：DescribeAttack 盾:DescribeAttack統計 shield:DescribeDRTAccess 屏蔽：DescribeEmergencyContactSettings 屏蔽：DescribeProtection 屏蔽：DescribeProtection組 屏蔽：DescribeSubscription 盾：DisableApplicationLayerAutomatic響應 盾牌：DisableProactive訂婚 屏蔽:取消關聯 DRT LogBucket shield:DisassociateDRTRole 屏蔽：DisassociateHealth檢查

服務前綴	動作
	盾：EnableApplicationLayerAutomatic響應 盾牌：EnableProactive訂婚 屏蔽：GetSubscription狀態 屏蔽：ListAttacks 屏蔽:ListProtection群組 屏蔽：ListProtections 屏蔽：ListResourcesInProtection組 盾：UpdateApplicationLayerAutomatic響應 屏蔽：UpdateEmergencyContactSettings 屏蔽：UpdateProtection組 屏蔽：UpdateSubscription

服務前綴	動作
signer	簽署者：AddProfile權限
	簽署者：CancelSigning設定檔
	簽署者：Job DescribeSigning
	簽署者：GetRevocation狀態
	簽署者：GetSigning平台
	簽署者：GetSigning設定檔
	簽署者：ListProfile權限
	簽署者：ListSigning工作
	簽署者：ListSigning平台
	簽署者：ListSigning設定檔
	簽署者：PutSigning設定檔
	簽署者：RemoveProfile權限
	簽署者：RevokeSignature
	簽署者：RevokeSigning設定檔
	簽署者：SignPayload
	簽署者：Job StartSigning

服務前綴	動作
simspaceweaver	模擬管理員：CreateSnapshot 模擬管理員：DeleteApp 模擬管理員：DeleteSimulation 模擬管理員：DescribeApp 模擬管理員：DescribeSimulation 模擬管理員：ListApps 模擬管理員：ListSimulations 模擬管理員：StartApp 模擬管理員：StartClock 模擬管理員：StartSimulation 模擬管理員：StopApp 模擬管理員：StopClock 模擬管理員：StopSimulation

服務前綴	動作
sms	短信:CreateApp 短信:CreateReplicationJob 短信:DeleteApp 短信:DeleteAppLaunchConfiguration 短信:DeleteAppReplicationConfiguration 短信:DeleteAppValidationConfiguration 短信:DeleteReplicationJob 短信：DeleteServer目錄 短信:DisassociateConnector 短信：GenerateChange設置 短信:GenerateTemplate 短信:GetApp 短信:GetAppLaunchConfiguration 短信:GetAppReplicationConfiguration 短信:GetAppValidationConfiguration 短信:GetAppValidationOutput 短信:GetConnectors 短信:GetReplication工作 短信：GetReplication運行 短信:GetServers 短信：ImportApp目錄

服務前綴	動作
	短信：ImportServer目錄
	短信:LaunchApp
	短信:ListApps
	短信:NotifyAppValidationOutput
	短信:PutAppLaunchConfiguration
	短信:PutAppReplicationConfiguration
	短信:PutAppValidationConfiguration
	短信:StartApp複製
	短信:StartOnDemandApp複製
	短信:StartOnDemandReplication運行
	短信:StopApp複製
	短信:TerminateApp
	短信:UpdateApp
	短信:UpdateReplicationJob

服務前綴	動作
sms-voice	短信語音：配置 AssociateProtect 短信語音：設置 CreateConfiguration 短信語音：目的地 CreateConfiguration SetEvent 短信語音：目的地 CreateEvent 短信語音：CreateOptOutList 短信語音：CreatePool 短信語音：配置 CreateProtect 短信語音：CreateRegistration 短信語音：協會 CreateRegistration 短信語音：附件 CreateRegistration 短信語音：版本 CreateRegistration 短信語音：CreateVerifiedDestinationNumber 短信語音：配置 DeleteAccount DefaultProtect 短信語音：設置 DeleteConfiguration 短信語音：目的地 DeleteConfiguration SetEvent 短信語音：DeleteDefaultMessageType 短信語音：DeleteDefaultSenderId 短信語音：目的地 DeleteEvent 短信語音：DeleteKeyword 短信語音：DeleteMediaMessageSpendLimitOverride 短信語音：DeleteOptedOutNumber

服務前綴	動作
	短信語音：DeleteOptOutList
	短信語音：DeletePool
	短信語音：配置 DeleteProtect
	短信語音：DeleteRegistration
	短信語音：附件 DeleteRegistration
	短信語音：DeleteTextMessageSpendLimitOverride
	短信語音：DeleteVerifiedDestinationNumber
	短信語音：DeleteVoiceMessageSpendLimitOverride
	短信語音：屬性 DescribeAccount
	短信語音：限制 DescribeAccount
	短信-語音：套 DescribeConfiguration
	短信語音：DescribeKeywords
	短信語音：DescribeOptedOutNumbers
	短信語音：DescribeOptOutLists
	短信語音：數字 DescribePhone
	短信語音：DescribePools
	短信語音：配置 DescribeProtect
	短信語音：附件 DescribeRegistration
	短信語音：DescribeRegistrationFieldDefinitions
	短信語音：DescribeRegistrationFieldValues
	短信語音：DescribeRegistrations

服務前綴	動作
	短信語音：DescribeRegistrationSectionDefinitions 短信語音：DescribeRegistrationTypeDefinitions 短信語音：版本 DescribeRegistration 短信語音：身份證 DescribeSender 短信語音：限制 DescribeSpend 短信語音：DescribeVerifiedDestinationNumbers 短信語音：身份 DisassociateOrigination 短信語音：配置 DisassociateProtect 短信語音：版本 DiscardRegistration 短信語音：目的地 GetConfiguration SetEvent 短信語音：GetProtectConfigurationCountryRuleSet 短信-語音：套 ListConfiguration 短信語音：ListPoolOriginationIdentities 短信語音：協會 ListRegistration 短信語音：PutKeyword 短信語音：PutOptedOutNumber 短信語音：數 ReleasePhone 短信語音：Id ReleaseSender 短信語音：數 RequestPhone 短信語音：Id RequestSender 短信語音：代碼 SendDestination NumberVerification

服務前綴	動作
	短信語音：配置 SetAccount DefaultProtect 短信語音：SetDefaultMessageType 短信語音：SetDefaultSenderId 短信語音：SetMediaMessageSpendLimitOverride 短信語音：SetTextMessageSpendLimitOverride 短信語音：SetVoiceMessageSpendLimitOverride 短信語音：版本 SubmitRegistration 短信語音：目的地 UpdateConfiguration SetEvent 短信語音：目的地 UpdateEvent 短信語音：數 UpdatePhone 短信語音：UpdatePool 短信語音：配置 UpdateProtect 短信語音：UpdateProtectConfigurationCountryRuleSet 短信語音：Id UpdateSender

服務前綴	動作
snowball	雪球：CancelCluster
	雪球：CancelJob
	雪球：CreateAddress
	雪球：CreateCluster
	雪球：CreateJob
	雪球：CreateLongTermPricing
	雪球：CreateReturnShippingLabel
	雪球：DescribeAddress
	雪球：DescribeAddresses
	雪球：DescribeCluster
	雪球：DescribeJob
	雪球：DescribeReturnShippingLabel
	雪球：GetJob清單
	雪球：GetJobUnlockCode
	雪球：GetSnowball用法
	雪球：GetSoftware更新
	雪球：ListCluster工作
	雪球：ListClusters
	雪球：圖ListCompatible片
	雪球：ListJobs
	雪球：ListLongTermPricing

服務前綴	動作
	雪球：位置 ListPickup 雪球：ListService版本 雪球：UpdateCluster 雪球：UpdateJob 雪球：UpdateJobShipmentState 雪球：UpdateLongTermPricing
sqs	平方：AddPermission 平方：CancelMessageMoveTask 平方：CreateQueue 平方：DeleteQueue 平方：PurgeQueue 平方：RemovePermission 平方米：SetQueue屬性

服務前綴	動作
ssm	特殊物品：項目 AssociateOps ItemRelated
	SSM：CancelCommand
	SSM：CancelMaintenanceWindowExecution
	SSM：CreateActivation
	SSM：CreateAssociation
	SSM: CreateAssociation Batch 處理
	SSM：CreateDocument
	SSM: CreateMaintenance 視窗
	特殊物品：項目 CreateOps
	ssm: CreateOps 中繼資料
	SSM：CreatePatch基準
	SSM：CreateResourceDataSync
	SSM：DeleteActivation
	SSM：DeleteAssociation
	SSM：DeleteDocument
	SSM：DeleteInventory
	SSM: DeleteMaintenance 視窗
	特殊物品：項目 DeleteOps
	ssm: DeleteOps 中繼資料
	SSM：DeleteParameter
	SSM：DeleteParameters

服務前綴	動作
	SSM：DeletePatch基準
	SSM：DeleteResourceDataSync
	政策 DeleteResource
	SSM: DeregisterManaged 執行個體
	SSM：DeregisterPatchBaselineForPatchGroup
	SSM: DeregisterTarget FromMaintenance 視窗
	SSM: DeregisterTask FromMaintenance 視窗
	SSM：DescribeActivations
	SSM：DescribeAssociation
	ssm: DescribeAssociation 執行程序
	SSM：DescribeAssociationExecutionTargets
	ssm: DescribeAutomation 執行程序
	SSM：DescribeAutomationStepExecutions
	ssm: 修補程式 DescribeAvailable
	SSM：DescribeDocument
	SSM：DescribeDocument參數
	SSM: DescribeDocument 權限
	SSM：DescribeEffectiveInstanceAssociations
	SSM：DescribeEffectivePatchesForPatchBaseline
	SSM：DescribeInstanceAssociationsStatus
	特殊信息 DescribeInstance

服務前綴	動作
	ssm: 修補程式 DescribeInstance
	SSM：DescribeInstancePatchStates
	SSM: DescribeInstance PatchStates ForPatch 集團
	SSM：DescribeInstance屬性
	ssm: 刪除 DescribeInventory
	SSM：DescribeMaintenanceWindowExecutions
	SSM：DescribeMaintenanceWindowExecutionTaskInvocations
	SSM：DescribeMaintenanceWindowExecution任務
	ssm: DescribeMaintenance 視窗
	SSM：DescribeMaintenanceWindowSchedule
	SSM：目標 DescribeMaintenance WindowsFor
	SSM：DescribeMaintenanceWindowTargets
	SSM：DescribeMaintenanceWindowTasks
	特殊物品：項目 DescribeOps
	SSM：DescribeParameters
	SSM: DescribePatch 基準線
	群組 DescribePatch
	SSM：DescribePatchGroupState
	SSM：DescribePatch屬性
	SSM：DescribeSessions
	特殊物品：項目 DisassociateOps ItemRelated

服務前綴	動作
	SSM：GetAutomation執行
	SSM: 州 GetCalendar
	ssm：GetCommand調用
	SSM：狀態 GetConnection
	SSM：GetDefaultPatchBaseline
	SSM：GetDeployablePatchSnapshotForInstance
	SSM：GetDocument
	SSM：GetInventory
	ssm: GetInventory 綱要
	SSM: GetMaintenance 視窗
	SSM：GetMaintenanceWindowExecution
	SSM：GetMaintenanceWindowExecution任務
	SSM：GetMaintenanceWindowExecutionTaskInvocation
	SSM：GetMaintenanceWindowTask
	特殊物品：項目 GetOps
	ssm: GetOps 中繼資料
	SSM：GetOps摘要
	SSM：GetParameter
	史：歷史 GetParameter
	SSM：GetParameters
	SSM：GetParametersByPath

服務前綴	動作
	SSM：GetPatch基準
	SSM：GetPatchBaselineForPatchGroup
	政策 GetResource
	特殊設置：GetService設置
	SSM: LabelParameter 版本
	SSM：ListAssociations
	SSM：ListAssociation版本
	ssm：ListCommand調用
	SSM：ListCommands
	特殊物品：項目 ListCompliance
	SSM：ListCompliance摘要
	SSM：ListDocumentMetadataHistory
	SSM：ListDocuments
	SSM：ListDocument版本
	SSM: ListInstance 協會
	ssm：ListInventory條目
	SSM：ListOpsItemEvents
	特殊物品：項目 ListOps ItemRelated
	ssm: ListOps 中繼資料
	SSM：ListResourceComplianceSummaries
	SSM：ListResourceDataSync

服務前綴	動作
	SSM: ModifyDocument 權限
	特殊物品：項目 PutCompliance
	SSM：PutInventory
	SSM：PutParameter
	政策 PutResource
	SSM：RegisterDefaultPatchBaseline
	SSM: RegisterManaged 執行個體
	SSM：RegisterPatchBaselineForPatchGroup
	SSM: RegisterTarget WithMaintenance 視窗
	SSM: RegisterTask WithMaintenance 視窗
	特殊設置：ResetService設置
	SSM：ResumeSession
	超音波：SendAutomation信號
	SSM：SendCommand
	ssm：StartAssociations一次
	SSM：StartAutomation執行
	SSM：StartChangeRequestExecution
	SSM：StartSession
	SSM：StopAutomation執行
	SSM：TerminateSession
	SSM: UnlabelParameter 版本

服務前綴	動作
	SSM：UpdateAssociation
	SSM：狀態 UpdateAssociation
	SSM：UpdateDocument
	SSM：UpdateDocumentDefaultVersion
	ssm: UpdateDocument 中繼資料
	特殊信息 UpdateInstance
	SSM: UpdateMaintenance 視窗
	SSM：UpdateMaintenanceWindowTarget
	SSM：UpdateMaintenanceWindowTask
	SSM：UpdateManagedInstanceRole
	特殊物品：項目 UpdateOps
	ssm: UpdateOps 中繼資料
	SSM：UpdatePatch基準
	SSM：UpdateResourceDataSync
	特殊設置：UpdateService設置

服務前綴	動作
ssm-incidents	短信事件：BatchGetIncidentFindings SSM 事件：設定 CreateReplication SSM 事件：計劃 CreateResponse SSM 事件：事件 CreateTimeline SSM 事件：記錄 DeleteIncident SSM 事件：設定 DeleteReplication SSM 事件：政策 DeleteResource SSM 事件：計劃 DeleteResponse SSM 事件：事件 DeleteTimeline SSM 事件：記錄 GetIncident SSM 事件：設定 GetReplication SSM 事件：政策 GetResource SSM 事件：計劃 GetResponse SSM 事件：事件 GetTimeline SSM 事件：發現項目 ListIncident SSM 事件：記錄 ListIncident SSM 事件：料號 ListRelated SSM 事件：集合 ListReplication SSM 事件：計劃 ListResponse SSM 事件：事件 ListTimeline SSM 事件：政策 PutResource

服務前綴	動作
	短信事件：StartIncident SSM 事件：保護 UpdateDeletion SSM 事件：記錄 UpdateIncident SSM 事件：料號 UpdateRelated SSM 事件：設定 UpdateReplication SSM 事件：計劃 UpdateResponse SSM 事件：事件 UpdateTimeline

服務前綴	動作
ssm-sap	SSM-汁液：BackupDatabase
	社會信息：權限 DeleteResource
	SSM-汁液：DeregisterApplication
	SSM-汁液：GetApplication
	糖果汁液：GetComponent
	糖果汁液：GetDatabase
	糖果汁液：GetOperation
	社會信息：權限 GetResource
	糖果汁液：ListApplications
	糖果汁液：ListComponents
	糖果汁液：ListDatabases
	社交活動：事件 ListOperation
	糖果汁液：ListOperations
	社會信息：權限 PutResource
	糖果汁液：RegisterApplication
	糖果汁液：RestoreDatabase
	糖果汁液：StartApplication
	SSM-汁液：重新整理 StartApplication
	糖果汁液：StopApplication
	社交媒體：設置 UpdateApplication
	SSM-SAP: 更新 BackupSettings

服務前綴	動作
states	州:CreateActivity 狀態:CreateState機器 州:CreateStateMachineAlias 州:DeleteActivity 狀態:DeleteState機器 州:DeleteStateMachineAlias 州:DeleteStateMachineVersion 州:DescribeActivity 州:DescribeExecution 狀態:DescribeMap執行 狀態:DescribeState機器 州:DescribeStateMachineAlias 狀態:DescribeStateMachineFor執行 州:GetExecution歷史 州:ListActivities 州:ListExecutions 狀態:ListMap執行 州:ListStateMachineAliases 州:ListState機器 州:ListStateMachineVersions 狀態:SendTask失敗

服務前綴	動作
	狀態:SendTask心跳 州:SendTask成功 州:StartExecution 州:StopExecution 狀態:UpdateMap執行 狀態:UpdateState機器 州:UpdateStateMachineAlias 州:ValidateStateMachineDefinition
sts	STS: AssumeRole STS：AssumeRole與 SAML STS: AssumeRole WithWeb 身份 STS: DecodeAuthorization 訊息 STS: GetAccess KeyInfo STS: GetCaller 身份 STS: GetFederation 令牌 STS: GetSession 令牌

服務前綴	動作
swf	SWF: DeprecateActivity 類型 SWF：DeprecateDomain SWF: DeprecateWorkflow 類型 SWF: DescribeActivity 類型 SWF：DescribeDomain SWF: DescribeWorkflow 類型 SWF: ListActivity 類型 SWF：ListDomains SWF: ListWorkflow 類型 SWF: RegisterActivity 類型 SWF：RegisterDomain SWF: RegisterWorkflow 類型 SWF: UndeprecateActivity 類型 SWF：UndeprecateDomain SWF: UndeprecateWorkflow 類型

服務前綴	動作
synthetics	合成材料：AssociateResource
	合成材料：CreateCanary
	合成材料：CreateGroup
	合成材料：DeleteCanary
	合成材料：DeleteGroup
	合成材料：DescribeCanaries
	合成材料：DescribeCanariesLastRun
	合成材料：DescribeRuntime版本
	合成材料：DisassociateResource
	合成材料：GetCanary
	合成材料：跑GetCanary步
	合成材料：GetGroup
	合成材料:ListAssociated團體
	合成材料：ListGroup資源
	合成材料：ListGroups
	合成材料：StartCanary
	合成材料：StopCanary
	合成材料：UpdateCanary

服務前綴	動作
標籤	標籤:DescribeReport創建 標籤:GetCompliance總結 標籤:GetResources 標籤:StartReport創建

服務前綴	動作
textract	文本：AnalyzeDocument 文本：AnalyzeExpense textract:AnalyzeID 文本：CreateAdapter 文字:CreateAdapter版本 文本：DeleteAdapter 文字:DeleteAdapter版本 文字：DetectDocument文字 文本：GetAdapter 文字:GetAdapter版本 文字:GetDocument分析 文本：GetDocumentTextDetection 文字:GetExpense分析 文字:GetLending分析 文本：GetLendingAnalysisSummary 文本：ListAdapters 文字：ListAdapter版本 文字:StartDocument分析 文本：StartDocumentTextDetection 文字:StartExpense分析 文字:StartLending分析

服務前綴	動作
	文本：UpdateAdapter

服務前綴	動作
timestream	時間流：CancelQuery 時間流：CreateDatabase 時間流:CreateScheduled查詢 時間流：CreateTable 時間流：DeleteDatabase 時間流:DeleteScheduled查詢 時間流：DeleteTable 時間流:DescribeAccount設置 時間流：DescribeDatabase 時間流:DescribeScheduled查詢 時間流：DescribeTable 時間流:ExecuteScheduled查詢 時間流：ListBatchLoadTasks 時間流：ListDatabases 時間流:ListScheduled查詢 時間流：ListTables 時間流：PrepareQuery 時間流:UpdateAccount設置 時間流：UpdateDatabase 時間流:UpdateScheduled查詢 時間流：UpdateTable

服務前綴	動作
tnb	注意事項：CancelSolNetworkOperation
	注意事項：CreateSolFunctionPackage
	注意事項：CreateSolNetworkInstance
	注意事項：CreateSolNetworkPackage
	注意事項：DeleteSolFunctionPackage
	注意事項：DeleteSolNetworkInstance
	注意事項：DeleteSolNetworkPackage
	注意事項：GetSolFunctionInstance
	注意事項：GetSolFunctionPackage
	注意:GetSolFunctionPackage內容
	TNB：描述元 GetSol FunctionPackage
	注意事項：GetSolNetworkInstance
	注意事項：GetSolNetworkOperation
	注意事項：GetSolNetworkPackage
	注意:GetSolNetworkPackage內容
	TNB：描述元 GetSol NetworkPackage
	注意事項：InstantiateSolNetworkInstance
	注意事項：ListSolFunctionInstances
	注意事項：ListSolFunctionPackages
	注意事項：ListSolNetworkInstances
	注意事項：ListSolNetworkOperations

服務前綴	動作
	注意事項：ListSolNetworkPackages 注意:PutSolFunctionPackage內容 注意:PutSolNetworkPackage內容 注意事項：TerminateSolNetworkInstance 注意事項：UpdateSolFunctionPackage 注意事項：UpdateSolNetworkInstance 注意事項：UpdateSolNetworkPackage 注意:ValidateSolFunctionPackage內容 注意:ValidateSolNetworkPackage內容

服務前綴	動作
transcribe	轉錄：CreateCallAnalyticsCategory 轉錄:模型 CreateLanguage 轉錄：詞彙 CreateMedical 轉錄：CreateVocabulary 轉錄:過濾器 CreateVocabulary 轉錄：DeleteCallAnalyticsCategory 轉錄：DeleteCallAnalyticsJob 轉錄:模型 DeleteLanguage 轉錄：DeleteMedicalScribeJob 轉錄：DeleteMedicalTranscriptionJob 轉錄：詞彙 DeleteMedical 轉錄:Job DeleteTranscription 轉錄：DeleteVocabulary 轉錄:過濾器 DeleteVocabulary 轉錄:模型 DescribeLanguage 轉錄：GetCallAnalyticsCategory 轉錄：GetCallAnalyticsJob 轉錄：GetMedicalScribeJob 轉錄：GetMedicalTranscriptionJob 轉錄：詞彙 GetMedical 轉錄:Job GetTranscription

服務前綴	動作
	轉錄：GetVocabulary 轉錄:過濾器 GetVocabulary 轉錄：ListCallAnalyticsCategories 轉錄：ListCallAnalyticsJobs 轉錄:模型 ListLanguage 轉錄：ListMedicalScribeJobs 轉錄：ListMedicalTranscriptionJobs 轉錄:詞彙 ListMedical 轉錄:工作 ListTranscription 轉錄：ListVocabularies 轉錄:篩選 ListVocabulary 轉錄：StartCallAnalyticsJob 轉錄:轉錄 StartCall AnalyticsStream 轉錄：插座 StartCall AnalyticsStream TranscriptionWeb 轉錄：StartMedicalScribeJob 轉錄：StartMedicalStreamTranscription 轉錄：StartMedicalStreamTranscriptionWebSocket 轉錄：StartMedicalTranscriptionJob 轉錄:轉錄 StartStream 轉錄：插座 StartStream TranscriptionWeb 轉錄:Job StartTranscription

服務前綴	動作
	轉錄：UpdateCallAnalyticsCategory 轉錄：詞彙 UpdateMedical 轉錄：UpdateVocabulary 轉錄:過濾器 UpdateVocabulary

服務前綴	動作
傳輸	轉移：CreateAccess
	轉移：CreateAgreement
	轉移：CreateConnector
	轉移：CreateProfile
	轉移：CreateServer
	轉移：CreateUser
	轉移：CreateWorkflow
	轉移：DeleteAccess
	轉移：DeleteAgreement
	轉移：DeleteCertificate
	轉移：DeleteConnector
	轉移:DeleteHost關鍵
	轉移：DeleteProfile
	轉移：DeleteServer
	轉移：DeleteSshPublicKey
	轉移：DeleteUser
	轉移：DeleteWorkflow
	轉移：DescribeAccess
	轉移：DescribeAgreement
	轉移：DescribeCertificate
	轉移：DescribeConnector

服務前綴	動作
	轉移：DescribeExecution
	轉移:DescribeHost關鍵
	轉移：DescribeProfile
	轉移:DescribeSecurity政策
	轉移：DescribeServer
	轉移：DescribeUser
	轉移：DescribeWorkflow
	轉移：ImportCertificate
	轉移:ImportHost關鍵
	轉移：ImportSshPublicKey
	轉移：ListAccesses
	轉移：ListCertificates
	轉移：ListConnectors
	轉移：ListExecutions
	轉移:ListHost鑰匙
	轉移：ListProfiles
	轉移:ListSecurity政策
	轉移：ListServers
	轉移：ListUsers
	轉移：ListWorkflows
	轉移：SendWorkflowStepState

服務前綴	動作
	轉移:StartDirectory清單
	轉移:StartFile轉移
	轉移：StartServer
	轉移：StopServer
	轉移：TestConnection
	轉移:TestIdentity供應商
	轉移：UpdateAccess
	轉移：UpdateAgreement
	轉移：UpdateCertificate
	轉移：UpdateConnector
	轉移:UpdateHost關鍵
	轉移：UpdateProfile
	轉移：UpdateServer
	轉移：UpdateUser

服務前綴	動作
translate	翻譯:CreateParallel資料 翻譯:DeleteParallel資料 翻譯：DeleteTerminology 翻譯：DescribeTextTranslationJob 翻譯:GetParallel資料 翻譯：GetTerminology 翻譯：ImportTerminology 翻譯：ListLanguages 翻譯:ListParallel資料 翻譯：ListTerminologies 翻譯：ListTextTranslationJobs 翻譯：StartTextTranslationJob 翻譯：StopTextTranslationJob 翻譯：TranslateDocument 翻譯：TranslateText 翻譯:UpdateParallel資料

服務前綴	動作
voiceid	語音符號：AssociateFraudster
	語音符號：CreateDomain
	語音符號：CreateWatchlist
	語音符號：DeleteDomain
	語音符號：DeleteFraudster
	語音符號：DeleteSpeaker
	語音符號：DeleteWatchlist
	語音符號：DescribeDomain
	語音符號：DescribeFraudster
	語音符號：DescribeFraudsterRegistrationJob
	語音符號：DescribeSpeaker
	語音符號：DescribeSpeakerEnrollmentJob
	語音符號：DescribeWatchlist
	語音符號：DisassociateFraudster
	語音符號：EvaluateSession
	語音符號：ListDomains
	語音符號：ListFraudsterRegistrationJobs
	語音符號：ListFraudsters
	語音符號：ListSpeakerEnrollmentJobs
	語音符號：ListSpeakers
	語音符號：ListWatchlists

服務前綴	動作
	語音識別:OptOut揚聲器 語音符號：StartFraudsterRegistrationJob 語音符號：StartSpeakerEnrollmentJob 語音符號：UpdateDomain 語音符號：UpdateWatchlist

服務前綴	動作
vpc-lattice	虛擬電路晶格：CreateAccessLogSubscription 虛擬電路晶格：CreateListener 虛擬電路晶格：CreateRule 虛擬電路晶格：CreateService 虛擬電腦晶格:網路 CreateService 虛擬電腦晶格:協會 CreateService NetworkService 虛擬電腦晶格:協會 CreateService NetworkVpc 虛擬電腦格子:組 CreateTarget 虛擬電路晶格：DeleteAccessLogSubscription 虛擬電腦格子：政策 DeleteAuth 虛擬電路晶格：DeleteListener 虛擬電腦格子：政策 DeleteResource 虛擬電路晶格：DeleteRule 虛擬電路晶格：DeleteService 虛擬電腦晶格:網路 DeleteService 虛擬電腦晶格:協會 DeleteService NetworkService 虛擬電腦晶格:協會 DeleteService NetworkVpc 虛擬電腦格子:組 DeleteTarget 虛擬電路晶格：DeregisterTargets 虛擬電路晶格：GetAccessLogSubscription 虛擬電腦格子：政策 GetAuth

服務前綴	動作
	虛擬電路晶格：GetListener 虛擬電腦格子：政策 GetResource 虛擬電路晶格：GetRule 虛擬電路晶格：GetService 虛擬電腦晶格:網路 GetService 虛擬電腦晶格:協會 GetService NetworkService 虛擬電腦晶格:協會 GetService NetworkVpc 虛擬電腦格子:組 GetTarget 虛擬電路晶格：ListAccessLogSubscriptions 虛擬電路晶格：ListListeners 虛擬電路晶格：ListRules 虛擬電腦晶格:網路 ListService 虛擬電腦晶格:協會 ListService NetworkService 虛擬電腦晶格:協會 ListService NetworkVpc 虛擬電路晶格：ListServices 虛擬電腦格子:群 ListTarget 虛擬電路晶格：ListTargets 虛擬電腦格子：政策 PutAuth 虛擬電腦格子：政策 PutResource 虛擬電路晶格：RegisterTargets 虛擬電路晶格：UpdateAccessLogSubscription

服務前綴	動作
	虛擬電路晶格：UpdateListener 虛擬電路晶格：UpdateRule 虛擬電路晶格：UpdateService 虛擬電腦晶格:網路 UpdateService 虛擬電腦晶格:協會 UpdateService NetworkVpc 虛擬電腦格子:組 UpdateTarget

服務前綴	動作
wafv2	wafv2: 前十字韌帶 AssociateWeb 波夫 2：CheckCapacity wafv2:CreateAPIKey wafv2:CreateIPSet 波夫 2：CreateRegexPatternSet 波夫 2: 集團 CreateRule wafv2: 前十字韌帶 CreateWeb WAFV2: 刪除安全鑰 波夫 2：群組 DeleteFirewall ManagerRule wafv2:DeleteIPSet wafv2：組態 DeleteLogging WAFV2：政策 DeletePermission 波夫 2：DeleteRegexPatternSet 波夫 2: 集團 DeleteRule wafv2: 前十字韌帶 DeleteWeb 波夫 2：DescribeAllManagedProducts 瓦夫 2: 供應商 DescribeManaged ProductsBy 波夫 2：DescribeManagedRuleGroup wafv2: 前十字韌帶 DisassociateWeb 瓦夫 2：網址 GenerateMobile SdkRelease wafv2：API 密鑰 GetDecrypted

服務前綴	動作
	wafv2:GetIPSet
	wafv2：組態 GetLogging
	波夫 2：GetManagedRuleSet
	波夫 2：GetMobileSdkRelease
	WAFV2：政策 GetPermission
	波夫 2：GetRateBasedStatementManagedKeys
	波夫 2：GetRegexPatternSet
	波夫 2: 集團 GetRule
	請求 GetSampled
	wafv2: 前十字韌帶 GetWeb ForResource
	wafv2:ListAPIKeys
	波夫 2：群組 ListAvailable ManagedRule
	波夫 2：ListAvailableManagedRuleGroupVersions
	wafv2:ListIPSets
	wafv2：配置 ListLogging
	波夫 2：ListManagedRuleSets
	波夫 2：ListMobileSdkReleases
	波夫 2：ListRegexPatternSets
	wafv2: 前十字韌帶 ListResources ForWeb
	波夫 2：群組 ListRule
	wafv2：ACL ListWeb

服務前綴	動作
	wafv2：組態 PutLogging WAFV2：版本 PutManaged RuleSet WAFV2：政策 PutPermission wafv2:UpdateIPSet 瓦夫 2：日期 UpdateManaged RuleSet VersionExpiry 波夫 2：UpdateRegexPatternSet 波夫 2: 集團 UpdateRule wafv2: 前十字韌帶 UpdateWeb

服務前綴	動作
wellarchitected	通過描述：AssociateLenses 通過描述：AssociateProfiles 致辭:分享 CreateLens 主要版本:版本 CreateLens 通過描述：CreateMilestone 通過描述：CreateProfile 致辭:分享 CreateProfile 常用:模板 CreateReview 通過描述：CreateWorkload 致辭:分享 CreateWorkload 通過描述：DeleteLens 致辭:分享 DeleteLens 通過描述：DeleteProfile 致辭:分享 DeleteProfile 常用:模板 DeleteReview 致辭:分享 DeleteTemplate 通過描述：DeleteWorkload 致辭:分享 DeleteWorkload 通過描述：DisassociateLenses 通過描述：DisassociateProfiles 通過描述：ExportLens

服務前綴	動作
	通過描述：GetAnswer 公司名稱:報告 GetConsolidated 常用:設置 GetGlobal 通過描述：GetLens 非常重要的:評論 GetLens 通過描述：GetLensReviewReport 通過描述：GetLensVersionDifference 通過描述：GetMilestone 通過描述：GetProfile 常用:模板 GetProfile 常用:模板 GetReview 通過描述：GetReviewTemplateAnswer 非常重要的:評論 GetReview TemplateLens 通過描述：GetWorkload 通過描述：ImportLens 通過描述：ListAnswers 通用：詳細信息 ListCheck 普遍性:摘要 ListCheck 通過描述：ListLenses 通過描述：ListLensReviewImprovements 非常重要的:點評 ListLens

服務前綴	動作
	公司名稱:股票 ListLens 通過描述：ListMilestones 通過描述：ListNotifications 通知:通知 ListProfile 通過描述：ListProfiles 公司名稱:股票 ListProfile 通過描述：ListReviewTemplateAnswers 引用:模板 ListReview 致辭:邀請函 ListShare 公司名稱:股票 ListTemplate 通過描述：ListWorkloads 公司名稱:股票 ListWorkload 通過描述：UpdateAnswer 常用:設置 UpdateGlobal 通過描述：UpdateIntegration 非常重要的:評論 UpdateLens 通過描述：UpdateProfile 常用:模板 UpdateReview 非常重要的:評論 UpdateReview TemplateLens 受邀者:邀請 UpdateShare 通過描述：UpdateWorkload

服務前綴	動作
	致辭:分享 UpdateWorkload 非常重要的:評論 UpgradeLens 主要版本:版本 UpgradeProfile 非常重要的:評論 UpgradeReview TemplateLens

服務前綴	動作
wisdom	智慧：CreateAssistant
	智慧：CreateAssistant協會
	智慧：CreateContent
	智慧：CreateKnowledge基礎
	智慧：CreateQuick回應
	智慧：CreateSession
	智慧：DeleteAssistant
	智慧：DeleteAssistant協會
	智慧：DeleteContent
	智慧：DeleteImportJob
	智慧：DeleteKnowledge基礎
	智慧：DeleteQuick回應
	智慧：GetAssistant
	智慧：GetAssistant協會
	智慧：GetContent
	智慧：GetContent總結
	智慧：GetImportJob
	智慧：GetKnowledge基礎
	智慧：GetRecommendations
	智慧：GetSession
	智慧：ListAssistant協會

服務前綴	動作
	智慧：ListAssistants
	智慧：ListContents
	智慧：ListImport工作
	智慧：ListKnowledge基地
	智慧：ListQuick回應
	智慧：NotifyRecommendations收到
	智慧：QueryAssistant
	智慧：RemoveKnowledgeBaseTemplate烏里
	智慧：SearchContent
	智慧：SearchQuick回應
	智慧：SearchSessions
	智慧：StartContent上傳
	智慧：StartImportJob
	智慧：UpdateContent
	智慧：UpdateKnowledgeBaseTemplate烏里
	智慧：UpdateQuick回應

服務前綴	動作
worklink	工作連結：AssociateDomain 工作連結：AssociateWebsiteAuthorizationProvider 工作連結：AssociateWebsiteCertificateAuthority 工作連結：CreateFleet 工作連結：DeleteFleet 工作連結：DescribeAuditStreamConfiguration 工作連結：DescribeCompanyNetworkConfiguration 工作連結：DescribeDevice 工作連結：DescribeDevicePolicyConfiguration 工作連結：DescribeDomain 工作連結:DescribeFleet詮釋資料 工作連結：DescribeIdentityProviderConfiguration 工作連結：DescribeWebsiteCertificateAuthority 工作連結：DisassociateDomain 工作連結：DisassociateWebsiteAuthorizationProvider 工作連結：DisassociateWebsiteCertificateAuthority 工作連結：ListDevices 工作連結：ListDomains 工作連結：ListFleets 工作連結：ListWebsiteAuthorizationProviders 工作連結：ListWebsiteCertificateAuthorities

服務前綴	動作
	工作連結:RestoreDomain存取 工作連結:RevokeDomain存取 工作連結:SignOut使用者 工作連結：UpdateAuditStreamConfiguration 工作連結：UpdateCompanyNetworkConfiguration 工作連結：UpdateDevicePolicyConfiguration 工作連結:UpdateDomain詮釋資料 工作連結:UpdateFleet詮釋資料 工作連結：UpdateIdentityProviderConfiguration

服務前綴	動作
工作區	工作區:AcceptAccountLinkInvitation
	工作區:AssociateConnection別名
	工作區:AssociateIp群組
	工作區:AssociateWorkspace應用
	工作區:CopyWorkspace影像
	工作區:CreateConnectClientAdd在
	工作區:CreateConnection別名
	工作區:CreateIp群組
	工作區:CreateStandby工作區
	工作區:CreateUpdatedWorkspaceImage
	工作區:CreateWorkspace套件
	工作區:CreateWorkspace影像
	工作區:CreateWorkspaces
	工作區:DeleteClient品牌
	工作區:DeleteConnectClientAdd在
	工作區:DeleteConnection別名
	工作區:DeleteIp群組
	工作區:DeleteWorkspace套件
	工作區:DeleteWorkspace影像
	工作區:DeployWorkspace應用
	工作區:DeregisterWorkspace目錄

服務前綴	動作
	工作區:DescribeAccount
	工作區:DescribeAccount修改
	工作區:DescribeApplication關聯
	工作區:DescribeApplications
	工作區:DescribeBundle關聯
	工作區:DescribeClient品牌
	工作區:DescribeClient性質
	工作區:DescribeConnectClientAdd插件
	工作區:DescribeConnection別名
	工作區:DescribeConnectionAliasPermissions
	工作區:DescribeImage關聯
	工作區:DescribeIp群組
	工作區:DescribeWorkspace關聯
	工作區:DescribeWorkspace套裝
	工作區:DescribeWorkspace目錄
	工作區:DescribeWorkspaceImagePermissions
	工作區:DescribeWorkspaces
	工作區:DescribeWorkspacesConnectionStatus
	工作區:DescribeWorkspace快照
	工作區:DisassociateConnection別名
	工作區:DisassociateIp群組

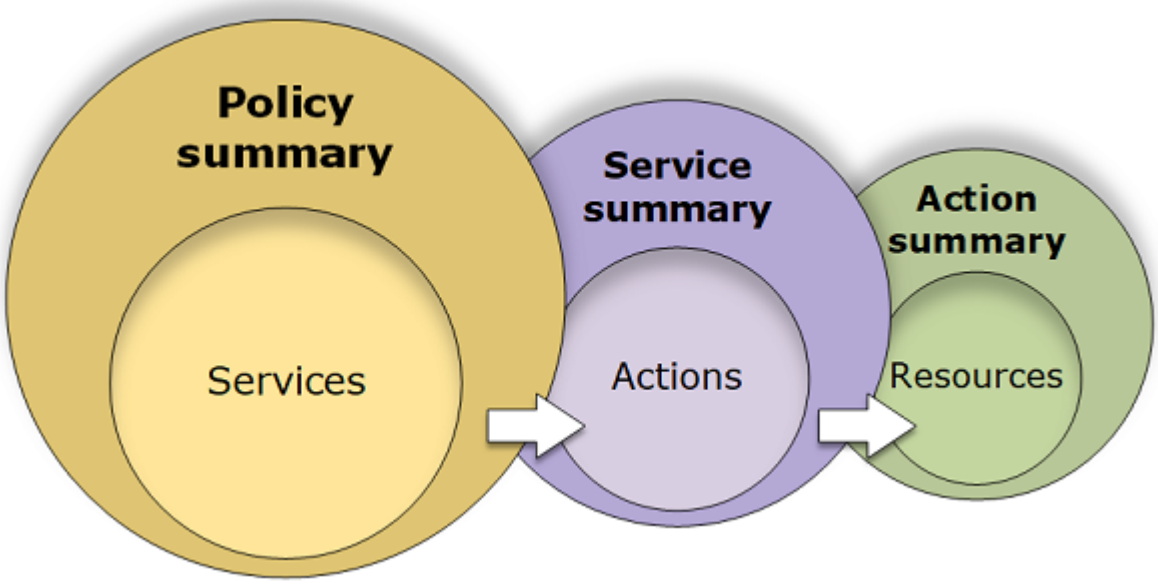
服務前綴	動作
	工作區:DisassociateWorkspace應用
	工作區:GetAccount連結
	工作區:ImportClient品牌
	工作區:ImportWorkspace影像
	工作區:ListAccount連結
	工作區:ListAvailableManagementCidr範圍
	工作區:MigrateWorkspace
	工作區:ModifyAccount
	工作區:ModifyCertificateBasedAuth性質
	工作區:ModifyClient性質
	工作區:ModifySaml性質
	工作區:ModifySelfservice權限
	工作區:ModifyWorkspaceAccessProperties
	工作區:ModifyWorkspaceCreationProperties
	工作區:ModifyWorkspace性質
	工作區:ModifyWorkspace狀態
	工作區:RebootWorkspaces
	工作區:RebuildWorkspaces
	工作區:RegisterWorkspace目錄
	工作區:RejectAccountLinkInvitation
	工作區:RestoreWorkspace

服務前綴	動作
	工作區:StartWorkspaces
	工作區:StopWorkspaces
	工作區:TerminateWorkspaces
	工作區:UpdateConnectClientAdd在
	工作區:UpdateConnectionAliasPermission
	工作區:UpdateWorkspace套件
	工作區:UpdateWorkspaceImagePermission

服務前綴	動作
xray	X 射線：CreateGroup X 射線：CreateSampling規則 X 射線：DeleteGroup X 射線：政策 DeleteResource X 射線：DeleteSampling規則 X 射線：Config GetEncryption X 射線：GetGroup X 射線：GetGroups X 射線：GetInsight X 射線：GetInsight活動 X 射線：GetInsightImpactGraph X 射線：GetInsight摘要 X 射線：GetSampling規則 X 射線：政策 ListResource X 射線：Config PutEncryption X 射線：政策 PutResource X 射線：UpdateGroup X 射線：UpdateSampling規則

了解政策授予的許可

IAM 主控台中提供了政策摘要表，這些表總結政策中對每個服務允許或拒絕的存取級別、資源和條件。政策摘要列於三個表中：政策摘要、服務摘要以及動作摘要。政策摘要表中包含服務清單。選擇其中一個服務來查看服務摘要。此摘要表包含所選服務的動作與相關許可的清單。您可以選擇該表中的動作以查看動作摘要。此表格包含所選動作的資源和條件清單。



您可以在 Users (使用者) 頁面或 Roles (角色) 頁面上查看連接到該使用者的所有政策 (受管和內嵌) 的政策摘要。在 Policies (政策) 頁面上檢視所有受管政策的摘要。受管政策包括 AWS 受管理的政策、AWS 受管理的工作職能政策和客戶管理的政策。您可以在 Policies (政策) 頁面上查看這些政策的摘要，無論它們是連接到使用者或其他 IAM 身分。

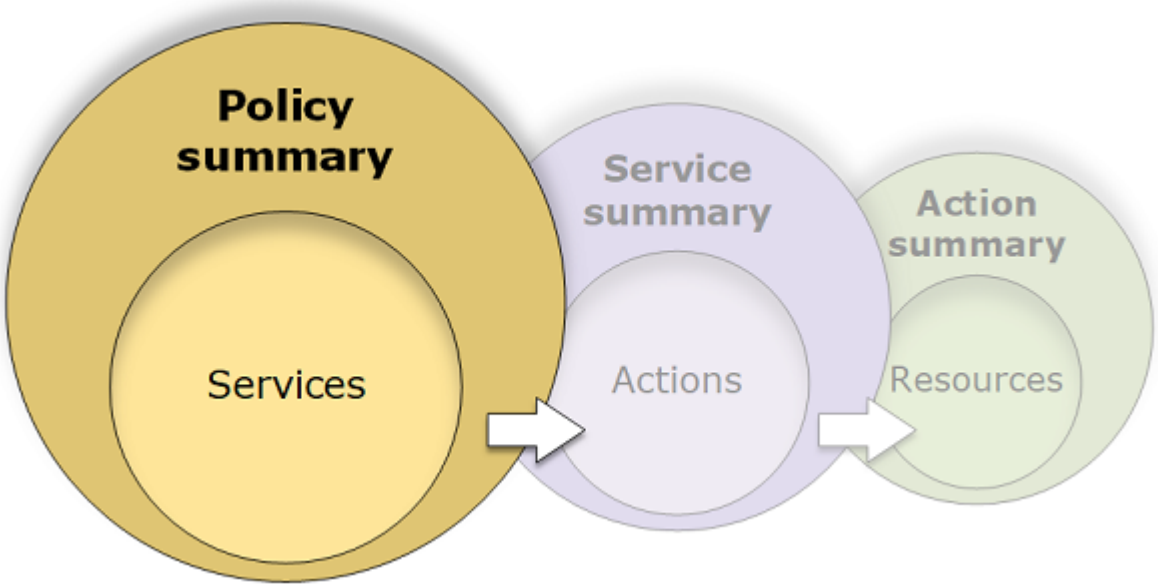
您可以使用政策摘要中的資訊來了解政策允許或拒絕的許可。政策摘要可協助您排除故障並修復未提供您預期的許可的政策。

主題

- 政策摘要 (服務清單)
- 服務摘要 (動作清單)
- 動作摘要 (資源清單)
- 政策摘要的範例

政策摘要 (服務清單)

政策摘要列於三個表中：政策摘要、服務摘要以及動作摘要。政策摘要表包括由所選政策定義的服務清單和許可摘要。



政策摘要表格是分成一或多個 Uncategorized services (未分類服務)、Explicit deny (明確拒絕)、以及 Allow (允許) 等部分。如果政策包含 IAM 無法辨識的服務，則該服務將包含在表格的 Uncategorized services (未分類服務) 部分中。如果 IAM 能夠辨識該服務，則它將包含在表格的 Explicit deny (明確拒絕) 或 Allow (允許) 部分中，取決於政策的效果 (`Deny` 或 `Allow`)。

檢視政策摘要

您可以在使用者詳細資料頁面的許可索引標籤上選擇政策名稱，以檢視連接至使用者的任何政策之摘要。您可以在角色詳細資料頁面的許可索引標籤上選擇政策名稱，以檢視連接至角色之任何政策的摘要。您可以在 Policies (政策) 頁面上檢視受管政策的政策摘要。如果您的政策不包含政策摘要，請參閱缺少政策摘要 以了解原因。

從 Policies (政策) 頁面查看政策摘要

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇您要檢視的政策名稱。
4. 在政策的政策詳細資訊頁面上，檢視許可索引標籤，可查看政策摘要。

查看連接到使用者的政策摘要

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 從導覽窗格選擇 Users (使用者)。
3. 在使用者清單中，選擇要檢視其政策的使用者的名稱。
4. 在使用者的 Summary (摘要) 頁面上，檢視 Permissions (許可) 標籤來查看直接連接到使用者或從群組連接的政策清單。
5. 在使用者的政策表中，展開您要檢視的政策列。

查看連接到角色的政策摘要

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Roles (角色)。
3. 在角色清單中，選擇要檢視其政策的角色名稱。
4. 在角色的 Summary (摘要) 頁面上，檢視 Permissions (許可) 標籤來查看直接連接到角色的政策清單。
5. 在角色的政策表中，展開您要檢視的政策列。

編輯政策以修正警告

在查看政策摘要時，您可能會發現拼寫錯誤，或者注意到政策未提供所需的許可。您無法直接編輯政策摘要。不過，您可以使用視覺化政策編輯器編輯客戶託管政策，該編輯器會捕捉政策摘要報告中許多相同的錯誤和警告。然後，您可以在政策摘要中查看更改以確認已修正所有問題。若要了解如何編輯內嵌政策，請參閱 the section called “編輯 IAM 政策”。您無法編輯 AWS 受管理的策略。

使用視覺化選項可編輯政策摘要的政策

1. 按照上述步驟中的說明開啟政策摘要。
2. 選擇編輯。

   如果您已打開 Users (使用者) 頁面，並選擇編輯連接到該使用者的客戶受管政策，系統會將您重新引導 Policies (政策) 頁面。您只能在 Policies (政策) 頁面上編輯客戶受管政策。

3. 選擇視覺化選項可檢視您的政策之可編輯視覺化形式。IAM 可能會調整您的政策結構以針對視覺化編輯器進行最佳化，並使您更輕鬆地查詢和解決任何問題。頁面上的警告和錯誤訊息可以引導您解決政策中的任何問題。如需有關 IAM 重新建構這些政策的詳細資訊，請參閱 政策結構調整。
4. 編輯您的政策，然後選擇下一步，可查看已在政策摘要中反映的變更。如果仍出現問題，請選擇 Previous (上一步) 以返回到編輯螢幕。
5. 選擇儲存變更，以儲存您所做的變更。

使用 JSON 選項可編輯政策摘要的政策

1. 按照上述步驟中的說明開啟政策摘要。
2. 您可以使用摘要和 JSON 按鈕，將政策摘要與 JSON 政策文件進行比較。您可以使用該資訊來決定要更改政策文件中的哪些行。
3. 選擇編輯，然後選擇 JSON 選項，以編輯 JSON 政策文件。

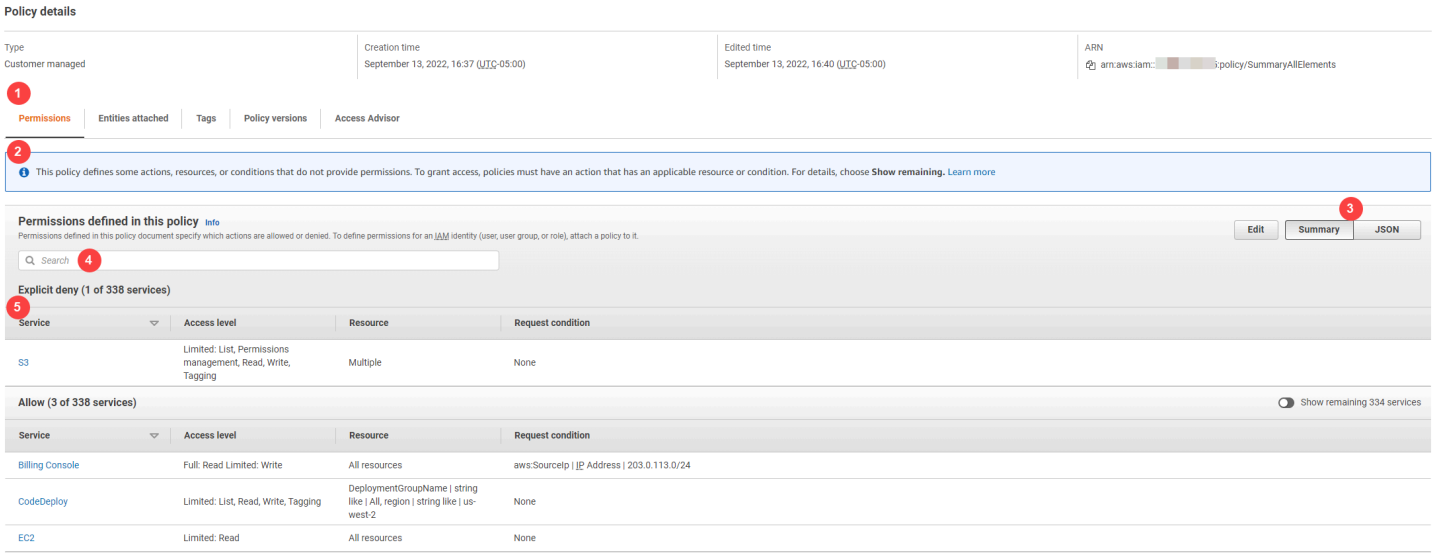
> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器選項中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

如果您已打開 Users (使用者) 頁面，並選擇編輯連接到該使用者的客戶受管政策，系統會將您重新引導 Policies (政策) 頁面。您只能在 Policies (政策) 頁面上編輯客戶受管政策。

4. 編輯政策。解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。如果仍出現問題，請選擇 Previous (上一步) 以返回到編輯螢幕。
5. 選擇儲存變更，以儲存您所做的變更。

了解政策摘要的元素

在下列原則詳細資料頁面範例中，SummaryAllElements原則是直接附加至使用者的受管理原則 (客戶管理策略)。此政策已展開並顯示了政策摘要。



在上圖中，政策摘要在政策頁面顯示：

1. 許可索引標籤包括政策中定義的許可。
2. 如果政策未授予許可給政策中定義的所有操作、資源和條件，則將在頁面頂部顯示警告或錯誤橫幅。政策摘要中包含關於問題的詳細資訊。若要了解政策摘可如何協助您了解政策授予的許權並進行相關問題故障排除，請參閱the section called “我的政策未授與預期的許可”.。
3. 使用摘要和 JSON 按鈕，可在政策摘要和 JSON 政策文件之間切換。
4. 使用搜尋方塊，可減少服務清單並查詢特定服務。
5. 展開的檢視會顯示SummaryAllElements原則的其他詳細資訊。

下列原則摘要表格影像顯示SummaryAllElements原則詳細資訊頁面上的擴充原則。

Explicit deny (1 of 338 services) A

Service B	Access level C	Resource D	Request condition E
S3	Limited: List, Permissions management, Read, Write, Tagging	Multiple	None

Allow (3 of 338 services) F Show remaining 334 services

Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp | IP Address | 203.0.113.0/24
CodeDeploy	Limited: List, Read, Write, Tagging	DeploymentGroupName | string like | All, region | string like | us-west-2	None
EC2	Limited: Read	All resources	None

在上圖中，政策摘要在政策頁面顯示：

A. 對於 IAM 辨識的那些服務，它根據政策是允許還是明確拒絕使用該服務來安排服務。在此範例中，該政策包括 Amazon S3 服務的Deny聲明和Allow帳單和 Amazon EC2 服務的聲明。 CodeDeploy

B. Service (服務) – 此欄列出在政策內定義的服務並提供每項服務的詳細資訊。政策摘要表中的每個服務名稱都是指向服務摘要表的一個連結，服務摘要 (動作清單) 中對其進行了說明。在此範例中，已定義 Amazon S3 CodeDeploy、帳單和 Amazon EC2 服務的許可。

C. 存取層級 – 此欄指出了在每個存取級別中的動作 (List、Read、Write、Permission Management 以及 Tagging) 是擁有 Full 許可還是政策中定義的 Limited 許可。有關存取許可級別摘要的更多詳細資訊和範例，請參閱 瞭解原則摘要中的存取層級。

- Full access (完整存取) – 此項目顯示服務對於該服務可用的全部四個存取層級中的所有動作都擁有存取許可。
- 如果該項不包含 Full access (完整存取)，則服務可以存取部分但不是全部用於該服務的動作。然後，根據每個存取級別分類 (List、Read、Write、Permission Management 以及 Tagging) 的說明，來定義存取許可：

  Full (完整)：政策提供對列出的每個存取級別分類中的所有動作的存取許可。在此範例中，政策提供對所有帳單 Read 操作的存取許可。

  Limited (限制)：政策提供對所列每個存取級別分類內的一或多個但非全部動作的存取許可。在此範例中，政策提供對部分帳單 Write 操作的存取許可。

D. Resource (資源) – 此欄顯示政策為每項服務指定的資源。

- Multiple (多個) – 政策包含服務內的多項資源，但並非全部。在此範例中，明確拒絕對多個 Amazon S3 資源的存取權限。
- 所有資源 – 為服務內的所有資源而定義政策。在此範例中，政策允許對所有帳單資源執行列出的操作。
- 資源文字 – 該政策包含服務內的一個資源。在此範例中，僅允許DeploymentGroupName CodeDeploy 資源上列出的動作。根據服務提供給 IAM 的資訊，您可能會看到一個 ARN，或者可能會看到定義的資源類型。

> **Note**
>
> 此欄位可能包含不同服務的資源。如果包含資源的政策陳述式，不包含來自相同服務的動作和資源，則您的政策包含不相符的資源。在建立政策或在政策摘要中查看政策時，IAM

不會警告您關於無法配對的資源。如果此欄位包含不相符的資源，則您應該檢閱您的政策錯誤。若要更全面了解您的政策，請一律以政策模擬器來測試。

E. Request condition (請求條件) – 此欄顯示與資源關聯的服務或動作是否受條件約束。

- None (無) – 政策對服務不包含任何條件。在此範例中，沒有條件適用於 Amazon S3 服務中拒絕的操作。
- 條件文字 – 政策包含服務的一項條件。在此範例中，僅當來源的 IP 地址與 `203.0.113.0/24` 匹配時，列出的帳單動作才會獲允許。
- Multiple (多項) – 政策包含服務的多項條件。若要檢視政策多項條件中的每一項條件，請選擇 JSON 來檢視政策文件。

F. 顯示剩餘的服務 – 切換此按鈕可展開資料表，以包括政策未定義的服務。這些服務在該政策中被隱含拒絕 (或根據預設拒絕)。但是，另一政策中的陳述中可能仍然允許或明確拒絕使用該服務。政策摘要匯總了單一政策的許可。若要瞭解 AWS 服務如何決定是否允許或拒絕指定要求，請參閱政策評估邏輯。

當政策或政策中的元素未授予許可時，IAM 在政策摘要中提供額外的警告和資訊。下列原則摘要表格顯示「在原SummaryAllElements則詳細資料」頁面上展開的「顯示剩餘服務」，其中包含可能的警告。

Explicit deny (1 of 338 services)

Service	Access level	Resource a	Request condition b
S3	Limited: List, Permissions management, Read, Write, Tagging	c Multiple| ⚠ One or more actions do not have an applicable resource.	None

Allow (3 of 338 services) Show remaining 334 services

Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp | IP Address | 203.0.113.0/24
CodeCommit	None	d ⚠ No resources are defined.	None
CodeDeploy	Limited: List, Read, Write, Tagging	e DeploymentGroupName | string like | All, region | string like | us-west-2| ⚠ One or more actions do not have an applicable resource.	None
EC2	Limited: Read	All resources	None
S3	None	None| ⚠ One or more actions do not have an applicable resource.	f None | ⚠ One or more conditions do not have an applicable action.

在上圖中，您可以看到包含沒有許可的已定義操作、資源或條件的所有服務：

a. Resource warnings (資源警告) – 對於沒有為所有包含的動作或資源提供許可的服務，您將在資料表的 Resource (資源) 欄中看到以下警告之一：

- 
  未定義資源。– 這表示該服務具有已定義的動作，但政策中不包含支援的資源。
- 
  一或多個動作沒有適用資源。– 表示該服務具有已定義的動作，但其中一些動作沒有支援的資源。
- 
  一或多個資源沒有適用動作。– 表示該服務具有已定義的資源，但其中一些資源沒有支援的動作。

如果服務同時包含沒有適用資源的動作和沒有適用資源的資源，系統只會顯示一個或多個資源沒有適用的動作警告。這是因為當您查看服務的服務摘要時，不會顯示不適用於任何動作的資源。對於 `ListAllMyBuckets` 動作，此政策包含最後一條警告，因為該動作不支援資源級許可，也不支援 `s3:x-amz-acl` 條件索引鍵。如果您修復資源問題或條件問題，剩餘的問題會出現在詳細的警告中。

b. Request condition warnings (請求條件警告) – 對於沒有為所有包含的條件提供許可的服務，您將在資料表的 Request condition (請求條件) 欄中看到以下警告之一：

- 
  一或多個動作沒有適用條件。– 表示該服務具有已定義的動作，但其中一些動作沒有支援的條件。
- 
  一或多個條件沒有適用動作。– 表示該服務具有已定義的條件，但其中一些條件沒有支援的動作。

c. Multiple (多個) |

一或多個動作沒有適用資源。– Amazon S3 的 `Deny` 陳述式包含一個以上的資源。它還包含多個動作，但其中一些動作支援資源，一些不支援。若要查看此政策，請參閱 the section called “SummaryAllElements JSON 政策文件”。在這種情況下，政策包含所有 Amazon S3 動作，只會拒絕可在儲存貯體或儲存貯體物件上執行的動作。

d. 

No resources are defined (未定義資源) – 服務具有已定義的動作，但政策中不包含支援的資源，因此服務不提供任何許可。在這種情況下，該策略包括 CodeCommit 動作，但不包括 CodeCommit 資源。

e. DeploymentGroupName | 類似的字符串 | 全部，區域 | 像 | us-west-2 |

⚠

一個或多個操作沒有適用的資源。– 服務具有定義的動作，且至少有額外一個沒有支援資源的動作。

f. 無 |

⚠

一個或多個條件沒有適用的動作。– 服務具有至少一個沒有支援動作的條件索引鍵。

SummaryAllElements JSON 政策文件

該SummaryAllElements策略不適用於您在帳戶中定義權限。包含它的目的在於說明您在查看政策摘要時可能會遇到的錯誤和警告。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "billing:Get*",
                "payments:List*",
                "payments:Update*",
                "account:Get*",
                "account:List*",
                "cur:GetUsage*"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "s3:*"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::customer",
                "arn:aws:s3:::customer/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshots"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedploy:*",
                "codecommit:*"
            ],
            "Resource": [
                "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*",
                "arn:aws:codebuild:us-east-1:123456789012:project/my-demo-project"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:GetObject",
                "s3:DeletObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::developer_bucket",
                "arn:aws:s3:::developer_bucket/*",
                "arn:aws:autoscling:us-east-2:123456789012:autoscalgrp"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ],
                    "s3:prefix": [
```

```
                    "custom",
                    "other"
                ]
            }
        }
    }
]
}
```

瞭解原則摘要中的存取層級

AWS 存取層級摘要

政策摘要包括說明針對政策中所提及的每項服務定義的動作許可的存取層級摘要。若要進一步了解政策摘要，請參閱了解政策授予的許可。從存取層級摘要可知政策中是否已為每個存取層級中的動作 (`List`、`Read`、`Tagging`、`Write` 和 `Permissions management`) 定義 `Full` 或 `Limited` 許可。若要檢視指派給服務中每個動作的存取層級分類，請參閱服務的動作、資源和條件索 AWS 引鍵。

以下範例說明針對特定服務政策所提供的存取權。如需完整的 JSON 政策文件及其相關摘要的範例，請參閱政策摘要的範例。

服務	存取層級	此政策提供下列
IAM	完整存取	存取 IAM 服務內的所有動作
CloudWatch	完整：清單	存取存`List`取層級中的所有 CloudWatch 動作，但無法存取`ReadWrite`、或`Permissions management` 存取層級分類的動作。
Data Pipeline	有限：清單、讀取	存取`List`和存取層級中至少一個但不是所有 AWS Data Pipeline 動作，但無法`Read`存取`Write`或`Permissions management` 動作。
EC2	完整：清單、讀取有限：寫入	存取所有 Amazon EC2 `List` 和 `Read` 動作，以及存取至少一個但並非所有 Amazon EC2 `Write` 動作，但是無法存取具 `Permissions management` 存取層級分類的動作。

服務	存取層級	此政策提供下列
S3	有限：讀取、寫入、許可管理	存取至少一個但並非所有 Amazon S3 `Read`、`Write` 和 `Permissions management` 動作。
CodeDeploy	(空白)	未知存取，因為 IAM 無法辨識此服務。
API Gateway	無	政策中未定義任何存取權。
CodeBuild	⚠ 未定義任何動作。	無法存取，因為未針對服務定義任何動作。若要了解如何了解和故障排除此問題，請參閱the section called “我的政策未授與預期的許可”。

如前述，完整存取表示政策提供服務內所有動作的存取權。政策提供存取服務內部分但非所有動作，並會進一步根據存取層級分類分組。這是根據下列存取層級的群組指出：

- Full (完整)：政策提供對指定的存取層級分類中的所有動作的存取權。
- Limited (有限)：政策提供指定存取層級分類內的一或多個但非全部動作的存取權。
- None (無)：政策不提供存取權。
- (空)：IAM 無法辨識此服務。如果服務名稱包含錯別字，則政策不提供服務的存取權。如果服務名稱正確，則該服務可能不支援政策摘要，或者可能處於預覽狀態。在這種情況下，政策可能會提供存取權，但無法顯示在政策摘要內。若請求全面供應 (GA) 服務的政策摘要支援，請參閱 服務不支援 IAM 政策摘要。

包含有限 (部分) 動作存取權限的存取層級摘要，會使用 AWS 存取層級分類`List`、`ReadTaggingWrite`、或`Permissions management`進行分組。

AWS 存取層級

AWS 為服務中的動作定義下列存取層級分類：

- List (清單)：列出服務內資源的許可，以判斷物件是否存在。具有此層級存取權的動作，可以列出物件，但無法查看資源的內容。例如，Amazon S3 動作 `ListBucket` 具有 List (清單) 存取層級。

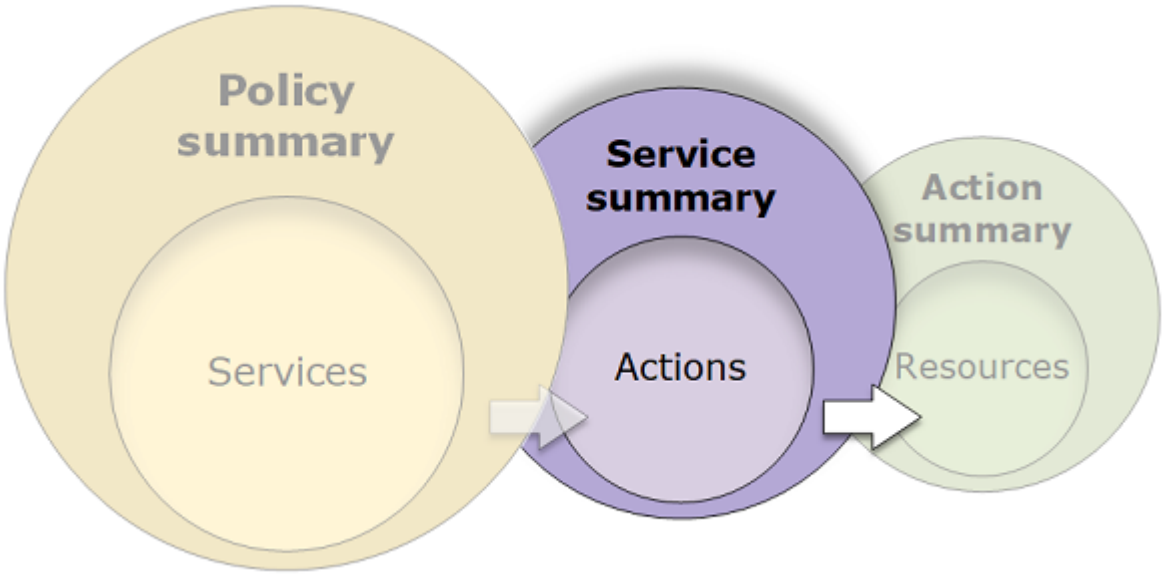
- Read (讀取)：可讀取但無法編輯服務內資源的內容和屬性的許可。例如，Amazon S3 動作 `GetObject` 和 `GetBucketLocation` 具有 Read (讀取) 存取層級。
- Tagging (標記)：執行僅變更資源標籤狀態之動作的許可。例如，IAM 動作 `TagRole` 及 `UntagRole` 具有 Tagging (標記) 存取層級，因為它們只允許標記或取消標記角色。不過，當您建立該角色時，`CreateRole` 動作允許標記角色資源。由於動作不會僅新增標籤，所以它具有 `Write` 存取層級。
- Write (寫入)：可建立、刪除或修改服務內資源的許可。例如，Amazon S3 動作 `CreateBucket`、`DeleteBucket` 及 `PutObject` 具有 Write (寫入) 存取層級。`Write` 動作可能也允許修改資源標籤。不過，動作僅允許變更具有 `Tagging` 存取層級的標籤。
- Permissions management (許可管理)：可授予或修改服務內資源許可的許可。例如，大多數 IAM 和 AWS Organizations 動作，以及 Amazon S3 動作之類的動作，`PutBucketPolicy`並`DeleteBucketPolicy`具有許可管理存取層級。

> **秘訣**
>
> 若要改善您的安全性 AWS 帳戶，請限制或定期監控包含權限管理存取層級分類的原則。

若要檢視服務中所有動作的存取層級分類，請參閱服務的動作、資源和條件索 AWS 引鍵。

服務摘要 (動作清單)

政策摘要列於三個表中：政策摘要、服務摘要以及動作摘要。服務摘要表包括動作清單和由政策針對所選服務定義的許可摘要。



您可以檢視授予許可的政策摘要中，所列的每個服務的服務摘要。該表已分組為 Uncategorized actions (未分類的動作)、Uncategorized resource types (未分類的資源類型) 和存取層級區段。如果政策包含 IAM 無法辨識的動作，則該動作將包含在資料表的 Uncategorized actions (未分類的動作) 區段中。若 IAM 可以辨識動作，則會將其包含在表格的其中一個存取層級區段之下 (List (列出)、Read (讀取)、Write (寫入) 和 Permissions management (許可管理))。若要檢視指派給服務中每個動作的存取層級分類，請參閱服務的動作、資源和條件索 AWS 引鍵。

檢視服務摘要

您可以在政策頁面上檢視受管政策的動作摘要。

若要檢視受管政策的服務摘要

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇您要檢視的政策名稱。
4. 在政策的政策詳細資訊頁面上，檢視許可索引標籤，可查看政策摘要。
5. 在服務的政策摘要清單中，選擇您要檢視的服務名稱。

若要檢視連接到使用者的政策服務摘要

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)。
3. 在使用者清單中，選擇要檢視其政策的使用者的名稱。
4. 在使用者的 Summary (摘要) 頁面上，檢視 Permissions (許可) 標籤來查看直接連接到使用者或從群組連接的政策清單。
5. 在使用者的政策表中，選擇您要檢視的政策名稱。

   如果您已打開使用者頁面，並選擇檢視連接到該使用者的政策服務摘要，系統會將您重新引導至政策頁面。您只能在政策頁面上檢視服務摘要。
6. 選擇摘要。在服務的政策摘要清單中，選擇您要檢視的服務名稱。

> **Note**
>
> 如果您選擇的政策是直接連接到使用者的內嵌政策，就會顯示服務摘要表。如果政策是從群組連接過來的內嵌政策，您會被帶引到 JSON 政策文件中的該群組。如果政策是受管政策，那麼您會被帶引到 Policies (政策) 頁面上該政策的服務摘要。

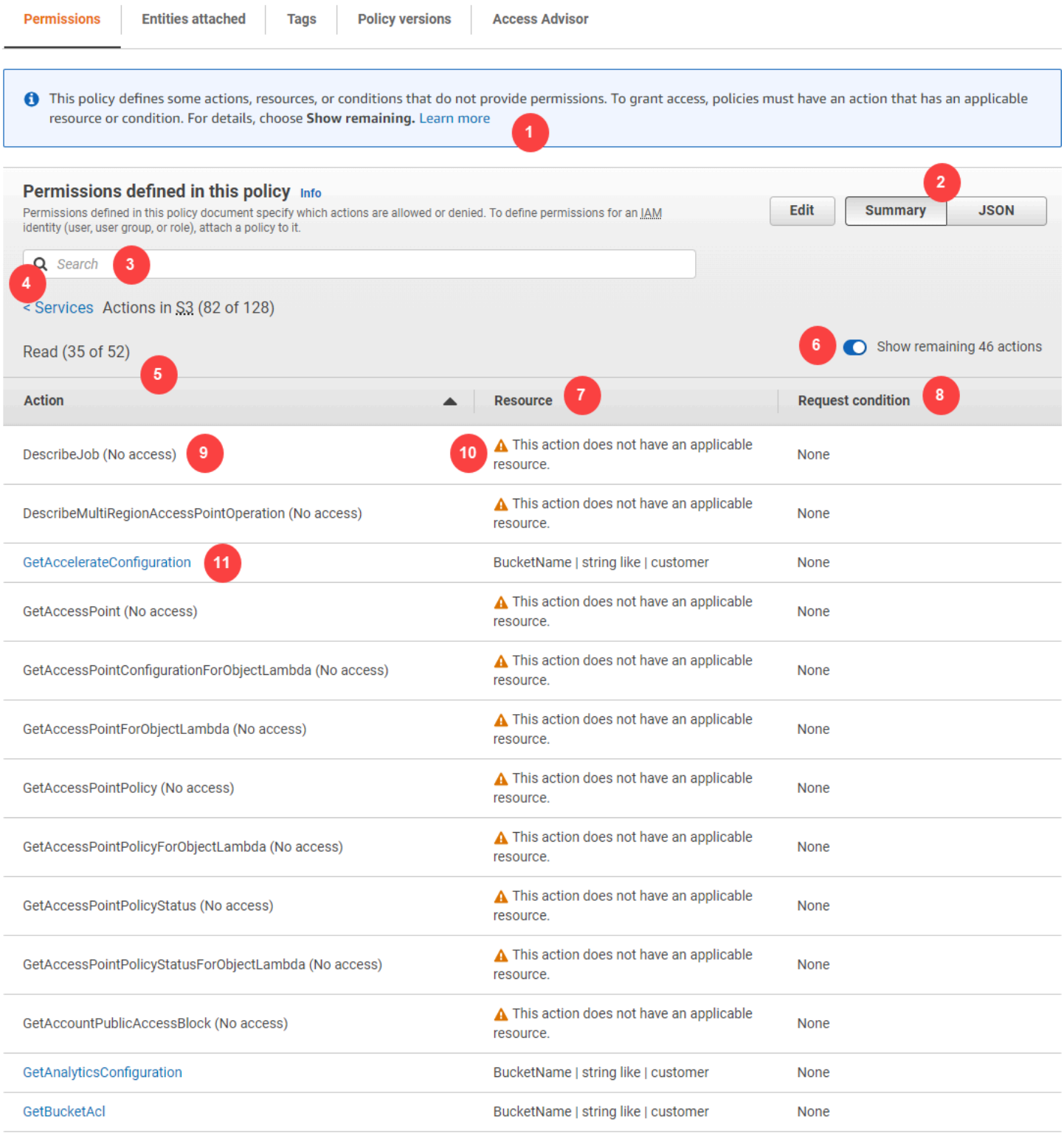
若要檢視連接到角色的政策服務摘要

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 從導覽窗格選擇 Roles (角色)。
3. 在角色清單中，選擇要檢視其政策的角色名稱。
4. 在角色的 Summary (摘要) 頁面上，檢視 Permissions (許可) 標籤來查看直接連接到角色的政策清單。
5. 在角色的政策表中，選擇您要檢視的政策名稱。

   如果您已打開角色頁面，並選擇檢視連接到該使用者的政策服務摘要，系統會將您重新引導至政策頁面。您只能在政策頁面上檢視服務摘要。
6. 在服務的政策摘要清單中，選擇您要檢視的服務名稱。

了解服務摘要的元素

以下範例是從政策摘要中允許的 Amazon S3 動作之服務摘要。此服務的動作會按照存取層級分組。例如，在可供服務使用的總計 52 個讀取動作中，有 35 個讀取動作已定義。



受管政策的服務摘要頁面包含以下資訊：

1. 如果政策未授予許可給政策中為服務定義的所有操作、資源和條件，則將在頁面頂部顯示警告橫幅。服務摘要中包含關於問題的詳細資訊。若要了解政策摘可如何協助您了解政策授予的許權並進行相關問題故障排除，請參閱the section called “我的政策未授與預期的許可”.。
2. 若要查看政策的其他詳細資訊，請選擇 JSON。您可以執行此操作，來查看套用到動作的所有條件。(如果您正在檢視直接連接到使用者的內嵌政策的服務摘要，您必須關閉服務摘要對話方塊並返回政策摘要，以存取 JSON 政策文件)。
3. 若要檢視特定動作的摘要，請在搜尋方塊中輸入關鍵字，以縮短可用動作清單。
4. 在服務返回箭頭旁邊，將顯示服務的名稱 (在此案例中，為 S3)。這項服務的服務摘要包含政策中定義的允許或拒絕之動作清單。如果服務出現在許可索引標籤上的 (明確拒絕) 下，系統會明確拒絕服務摘要表中列出的動作。如果服務出現在許可索引標籤上的允許下，系統會允許服務摘要表中列出的動作。
5. 動作 – 此欄位會列出在政策中定義的動作，並提供每個動作的資源和條件。如果政策授予或拒絕授予許可給動作，動作名稱會連結到動作摘要資料表。表格會根據政策允許或拒絕的存取層級，將這些動作分組到至少一個或多達五個區段。區段為列出、讀取、寫入、許可管理以及標記。計數表示在每個存取層級內提供許可的已認可動作。總計是服務的已知動作數。在這個範例中，在總計 52 個已知 Amazon S3 讀取動作中，有 35 個動作提供許可。若要檢視指派給服務中每個動作的存取層級分類，請參閱服務的動作、資源和條件索 AWS 引鍵。
6. 顯示剩餘的動作 – 切換此按鈕可展開或隱藏資料表，以包含已知但不提供此服務的許可之動作。切換該按鈕也會針對不提供許可的任何元素顯示警告。
7. Resource (資源) – 此欄位顯示政策為服務定義的資源。IAM 不會檢查資源是否適用於每個動作。在此範例中，Amazon S3 服務中的動作只允許對 `developer_bucket` Amazon S3 儲存貯體資源執行。根據服務提供給 IAM 的資訊，您可能會看到一個 ARN (例如 `arn:aws:s3:::developer_bucket/*`)，或者您可能會看到定義的資源類型 (例如 `BucketName = developer_bucket`)。

   > **Note**
   >
   > 此欄位可能包含不同服務的資源。如果包含資源的政策陳述式，不包含來自相同服務的動作和資源，則您的政策包含不相符的資源。在建立政策或在服務摘要中查看政策時，IAM 不會警告您關於不相符的資源。IAM 也不會指出動作是否適用於資源或者服務是否相符。如果此欄位包含不相符的資源，則您應該檢閱您的政策錯誤。若要更全面了解您的政策，請一律以政策模擬器來測試。

8. Request condition (請求條件) – 此欄指出與資源關聯的動作是否受條件約束。若要進一步了解相關條件，請選擇 JSON 以檢視 JSON 政策文件。
9. (No access) (沒有存取) – 此政策包含不提供許可的動作。

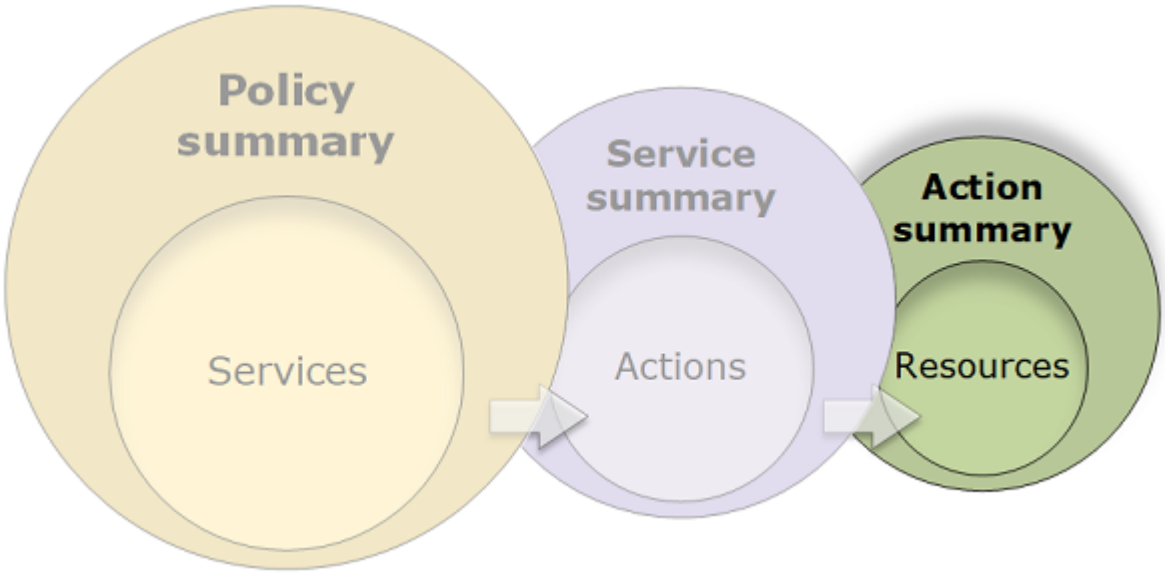
10Resource warning (資源警告) – 對於不提供完整許可的資源動作，請參閱以下其中一個警告：

- This action does not support resource-level permissions. This requires a wildcard (*) for the resource. (此動作不支援資源層級的許可，這需要對資源使用萬用字元 (*))。– 這表示政策包含資源層級許可，但必須包含 `"Resource": ["*"]` 才能提供此動作的許可。
- This action does not have an applicable resource. (此動作沒有適用的資源)。– 這表示動作包含在政策內，而無支援的資源。
- This action does not have an applicable resource and condition. (此動作沒有適用的資源和條件)。– 這表示動作包含在政策內，而無支援的資源，也無支援的條件。在這種情況下，此服務的政策內也包含條件，但沒有適用此動作的條件。

11提供許可的動作包含連到動作摘要的連結。

動作摘要 (資源清單)

政策摘要列於三個表中：政策摘要、服務摘要以及動作摘要。動作摘要表中包含一份資源清單，以及適用於所選動作的關聯條件。



若要檢視每個授予許可的動作摘要，請選擇服務摘要中的連結。動作摘要表包含關於資源的詳細資訊，包括其 Region (區域) 和 Account (帳戶)。您也可以檢視套用到各個資源的條件。這會顯示適用某些資源，但不適用其他資源的條件。

檢視動作摘要

您可以在政策頁面上檢視受管政策、任何連接至使用者的政策，以及任何連接至角色的政策之動作摘要。

檢視受管政策的動作摘要

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇您要檢視的政策名稱。
4. 在政策的政策詳細資訊頁面上，檢視許可索引標籤，可查看政策摘要。
5. 在服務的政策摘要清單中，選擇您要檢視的服務名稱。
6. 在動作的服務摘要清單中，選擇您要檢視的動作名稱。

檢視連接到使用者的政策動作摘要

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 從導覽窗格選擇 Users (使用者)。
3. 在使用者清單中，選擇要檢視其政策的使用者的名稱。
4. 在使用者的 Summary (摘要) 頁面上，檢視 Permissions (許可) 標籤來查看直接連接到使用者或從群組連接的政策清單。
5. 在使用者的政策表中，選擇您要檢視的政策名稱。

   如果您已打開使用者頁面，並選擇檢視連接到該使用者的政策服務摘要，系統會將您重新引導至政策頁面。您只能在政策頁面上檢視服務摘要。
6. 在服務的政策摘要清單中，選擇您要檢視的服務名稱。

   > **Note**
   >
   > 如果您選擇的政策是直接連接到使用者的內嵌政策，就會顯示服務摘要表。如果政策是從群組連接過來的內嵌政策，您會被帶引到 JSON 政策文件中的該群組。如果政策是受管政策，那麼您會被帶引到 Policies (政策) 頁面上該政策的服務摘要。
7. 在動作的服務摘要清單中，選擇您要檢視的動作名稱。

檢視連接到角色的政策動作摘要

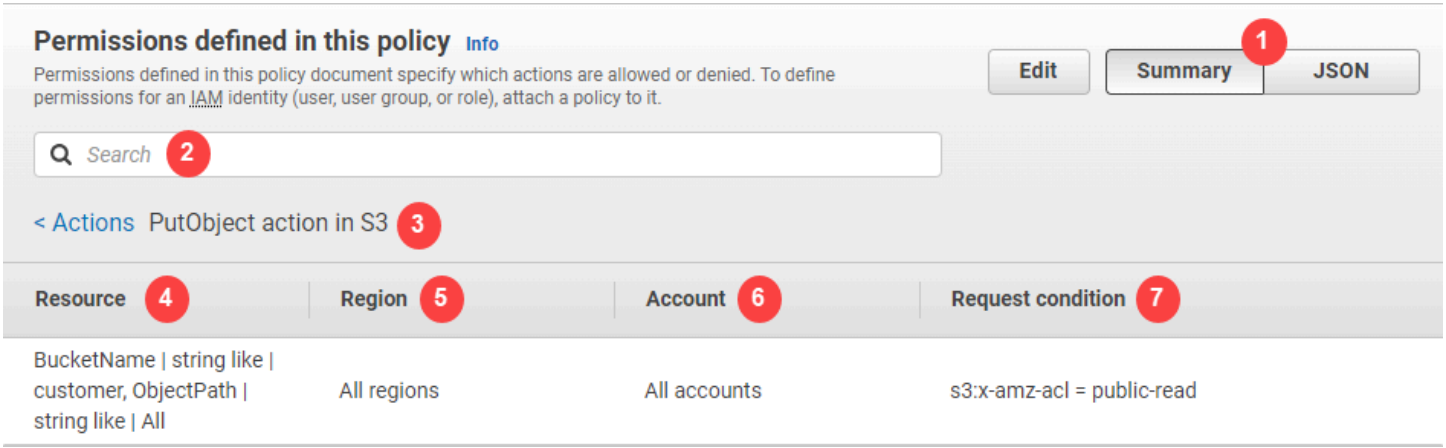
1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在導覽窗格中，選擇 Roles (角色)。
3. 在角色清單中，選擇要檢視其政策的角色名稱。
4. 在角色的 Summary (摘要) 頁面上，檢視 Permissions (許可) 標籤來查看直接連接到角色的政策清單。
5. 在角色的政策表中，選擇您要檢視的政策名稱。

   如果您已打開角色頁面，並選擇檢視連接到該使用者的政策服務摘要，系統會將您重新引導至政策頁面。您只能在政策頁面上檢視服務摘要。
6. 在服務的政策摘要清單中，選擇您要檢視的服務名稱。
7. 在動作的服務摘要清單中，選擇您要檢視的動作名稱。

了解動作摘要的元素

以下範例是來自 Amazon S3 服務摘要的 `PutObject` (寫入) 動作的動作摘要 (請參閱 服務摘要 (動作清單))。對於此動作，政策在單一資源定義多個條件。



動作摘要頁面包含以下資訊：

1. 選擇 JSON，可查看其他有關政策的詳細資訊，例如檢視套用到動作的多個條件。(如果您正在檢視直接連接到使用者的內嵌政策的動作摘要，則步驟有所不同。在這種情況下，您必須關閉動作摘要對話方塊並返回政策摘要，才能存取 JSON 政策文件)。
2. 若要檢視特定資源的摘要，請在搜尋方塊中輸入關鍵字，以減少可用資源清單。
3. 旁邊的動作返回箭頭出現的格式服務和動作的名稱`action name action in service`（在這種情況下，在 S3 中的操PutObject作）。這項服務的動作摘要包含政策中定義的資源清單。
4. Resource (資源) – 此欄列示政策為所選服務定義的資源。在此範例中，所有物件路徑都允許執行此PutObject動作，但只允許在 `developer_bucket` Amazon S3 儲存貯體資源上執行此動作。根

據服務提供給 IAM 的資訊，您可能會看到一個 ARN (例如 `arn:aws:s3:::developer_bucket/*`)，或者您可能會看到定義的資源類型 (例如 `BucketName = developer_bucket, ObjectPath = All`)。

5. Region (區域) – 此欄顯示定義資源的區域。可針對所有區域或單一區域定義資源。它們無法存在於一個以上的特定區域。
   - 所有區域 – 與資源關聯的動作適用於所有區域。在這個範例中，動作屬於全球服務 Amazon S3。屬於全球服務的動作適用於所有區域。
   - Region text (區域文字) – 與資源關聯的動作適用於一個區域。例如，政策可以指定資源的 `us-east-2` 區域。
6. Account (帳戶) – 此欄顯示與資源關聯的服務或動作是否套用於特定帳戶。資源可以存在於所有帳戶或單一帳戶。它們無法存在於一個以上的特定帳戶。
   - All accounts (所有帳戶) – 與資源關聯的動作適用於所有帳戶。在這個範例中，動作屬於全球服務 Amazon S3。屬於全球服務的動作適用於所有帳戶。
   - 這個帳戶 – 與資源關聯的動作只適用於目前的帳戶。
   - Account number (帳戶編號) – 與資源關聯的動作，套用於一個帳戶 (您目前未登入的帳戶)。例如，如果政策指定 `123456789012` 帳戶供資源使用，則帳號會出現在政策摘要中。
7. Request condition (請求條件) – 此欄顯示與資源關聯的動作是否受條件約束。此範例包含 `s3:x-amz-acl = public-read` 條件。若要進一步了解相關條件，請選擇 JSON 以檢視 JSON 政策文件。

政策摘要的範例

以下範例包括將 JSON 政策加入關聯的政策摘要、服務摘要、以及動作摘要，以協助您了解透過政策提供的許可。

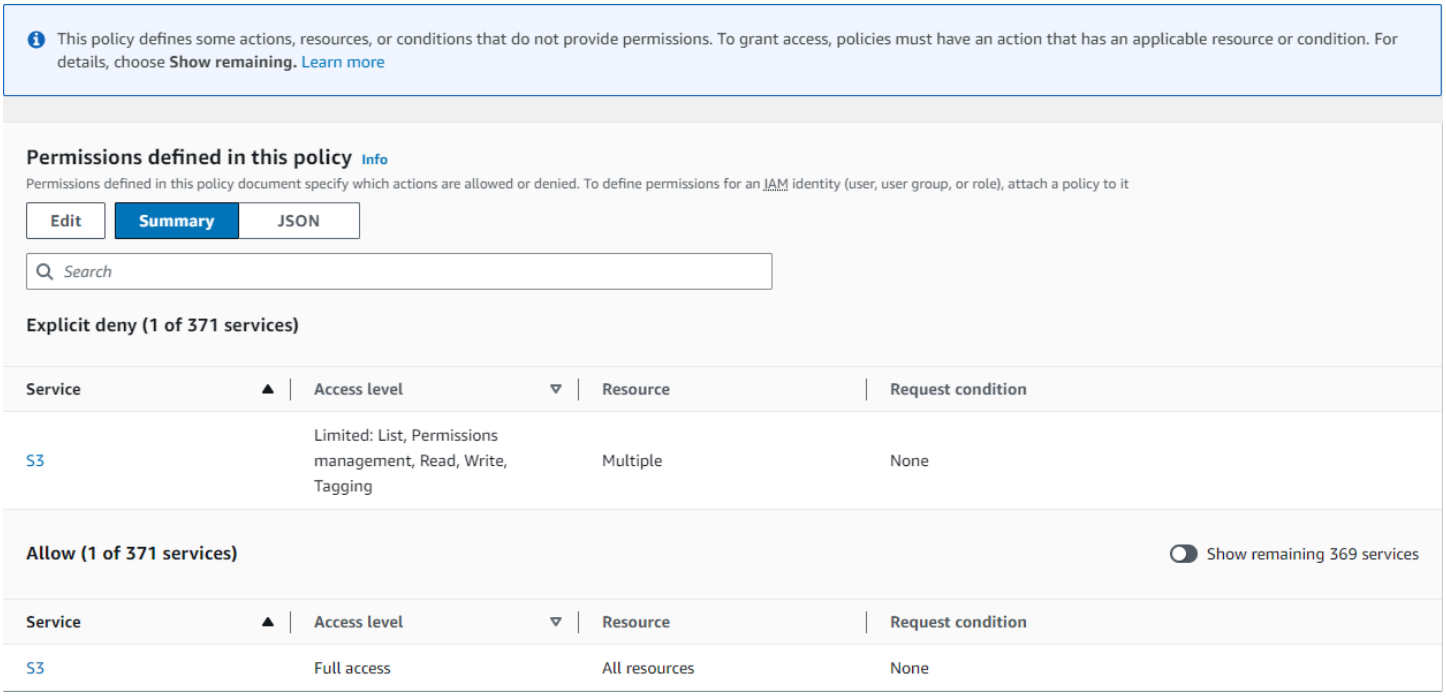
政策一： DenyCustomerBucket

此政策示範對相同服務的允許和拒絕。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccess",
            "Effect": "Allow",
            "Action": ["s3:*"],
```

```
            "Resource": ["*"]
        },
        {
            "Sid": "DenyCustomerBucket",
            "Action": ["s3:*"],
            "Effect": "Deny",
            "Resource": ["arn:aws:s3:::customer", "arn:aws:s3:::customer/*" ]
        }
    ]
}
```

DenyCustomerBucket政策摘要：



DenyCustomerBucket S3 (明確拒絕) 服務摘要：

< Services Actions in S3 (82 of 130) Show remaining 48 actions

Read (35 of 53)

Action	Resource	Request condition
GetAccelerateConfiguration	BucketName| string like |customer	None
GetAnalyticsConfiguration	BucketName| string like |customer	None
GetBucketAcl	BucketName| string like |customer	None
GetBucketCORS	BucketName| string like |customer	None
GetBucketLocation	BucketName| string like |customer	None
GetBucketLogging	BucketName| string like |customer	None
GetBucketNotification	BucketName| string like |customer	None
GetBucketObjectLockConfiguration	BucketName| string like |customer	None
GetBucketOwnershipControls	BucketName| string like |customer	None
GetBucketPolicy	BucketName| string like |customer	None
GetBucketPolicyStatus	BucketName| string like |customer	None
GetBucketPublicAccessBlock	BucketName| string like |customer	None
GetBucketRequestPayment	BucketName| string like |customer	None
GetBucketTagging	BucketName| string like |customer	None
GetBucketVersioning	BucketName| string like |customer	None
GetBucketWebsite	BucketName| string like |customer	None

GetObject (閱讀) 動作摘要：

< Actions GetObject action in S3

Resource	Region	Account	Request condition
BucketName| string like |customer, ObjectPath| string like |All	-	All accounts	None

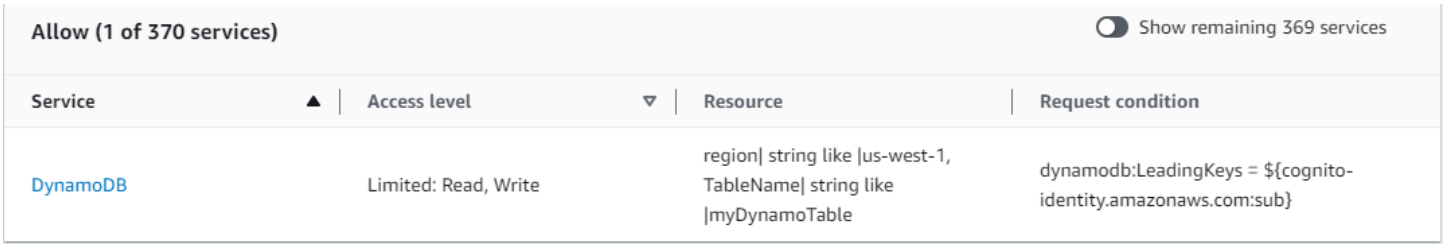
原則 2： DynamoDbRowCognito識別碼

此政策根據使用者的 Amazon Cognito ID 提供對 Amazon DynamoDB 的低層級存取權限。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-1:123456789012:table/myDynamoTable"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${cognito-identity.amazonaws.com:sub}"
                    ]
                }
            }
        }
    ]
}
```

DynamoDbRowCognitoID 原則摘要：



DynamoDbRowCognito識別碼 (允許) 服務摘要：

< Services Actions in DynamoDB (4 of 65) Show remaining 61 actions

Read (1 of 26)

Action	Resource	Request condition
GetItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

Write (3 of 33)

Action	Resource	Request condition
DeleteItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}
PutItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}
UpdateItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

GetItem (清單) 動作摘要：

< Actions GetItem action in DynamoDB

Resource	Region	Account	Request condition
region| string like |us-west-1, TableName| string like |myDynamoTable	us-west-1	123456789012	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

政策三： MultipleResourceCondition

此政策包含多個資源和條件。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Apple_bucket/*"],
            "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Orange_bucket/*"],
            "Condition": {"StringEquals": {
                "s3:x-amz-acl": ["custom"],
                "s3:x-amz-grant-full-control": ["1234"]
            }}
        }
    ]
}
```

MultipleResourceCondition政策摘要：

Allow (1 of 370 services) Show remaining 369 services

Service	Access level	Resource	Request condition
S3	Limited: Permissions management, Write	Multiple	Multiple

MultipleResourceCondition S3 (允許) 服務摘要：

< Services Actions in S3 (2 of 130) Show remaining 128 actions

Write (1 of 47)

Action	Resource	Request condition
PutObject	Multiple	Multiple

Permission Management (1 of 15)

Action	Resource	Request condition
PutObjectAcl	Multiple	Multiple

PutObject (寫入) 動作摘要：

< Actions PutObject action in S3

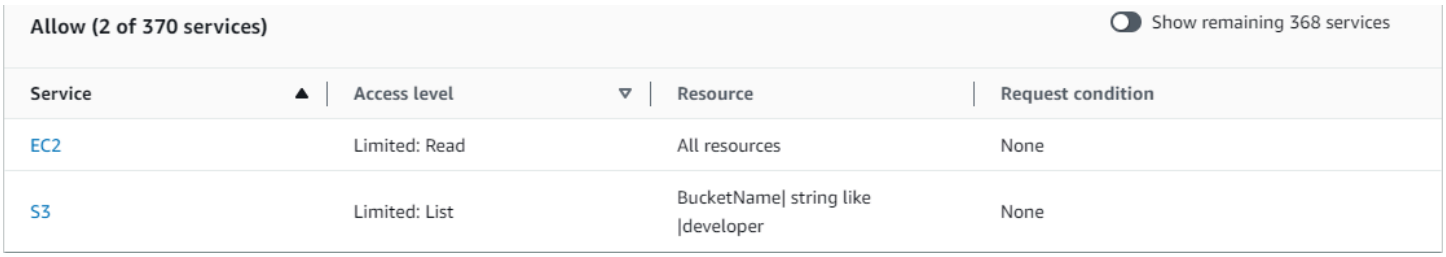
Resource	Region	Account	Request condition
Multiple	-	All accounts	Multiple

政策 4：EC2_troubleshoot

以下政策可讓使用者取得執行中的 Amazon EC2 執行個體螢幕截圖，可協助執行 EC2 故障排除。此政策也允許檢視 Amazon S3 開發人員儲存貯體中項目的相關資訊。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshot"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::developer"
            ]
        }
    ]
}
```

EC2_Troubleshoot 政策摘要：



Allow (2 of 370 services) — Show remaining 368 services

Service	Access level	Resource	Request condition
EC2	Limited: Read	All resources	None
S3	Limited: List	BucketName| string like |developer	None

EC2_Troubleshoot S3 (允許) 服務摘要：

< Services Actions in S3 (1 of 130) Show remaining 129 actions

List (1 of 7)

Action	Resource	Request condition
ListBucket	BucketName| string like |developer	None

ListBucket (清單) 動作摘要：

< Actions ListBucket action in S3

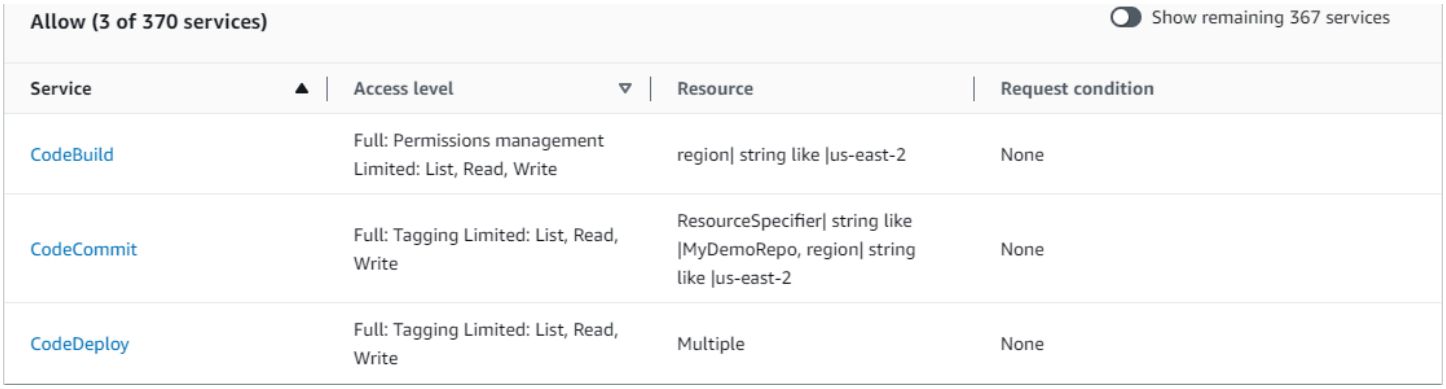
Resource	Region	Account	Request condition
BucketName| string like |developer	-	All accounts	None

政策五： CodeBuild_ CodeCommit _ CodeDeploy

此原則可讓您存取特定 CodeBuild CodeCommit、和 CodeDeploy 資源。由於這些資源對於每個服務來說都是專有的，因此只會在對應服務中顯示。如果您在 `Action` 元素中加入不符合任何服務的資源，那麼資源會顯示在所有動作摘要中。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487980617000",
            "Effect": "Allow",
            "Action": [
                "codebuild:*",
                "codecommit:*",
                "codedeploy:*"
            ],
            "Resource": [
                "arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project",
                "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
                "arn:aws:codedeploy:us-east-2:123456789012:application:WordPress_App",
                "arn:aws:codedeploy:us-east-2:123456789012:instance/AssetTag*"
            ]
        }
    ]
}
```

CodeBuildCodeCommit_ CodeDeploy 政策摘要：



Allow (3 of 370 services) Show remaining 367 services

Service	Access level	Resource	Request condition
CodeBuild	Full: Permissions management Limited: List, Read, Write	region| string like |us-east-2	None
CodeCommit	Full: Tagging Limited: List, Read, Write	ResourceSpecifier| string like |MyDemoRepo, region| string like |us-east-2	None
CodeDeploy	Full: Tagging Limited: List, Read, Write	Multiple	None

CodeBuild_ CodeCommit _ CodeDeploy CodeBuild (允許) 服務摘要：

< Services Actions in CodeBuild (24 of 53) Show remaining 29 actions

Read (4 of 9)

Action	Resource	Request condition
BatchGetBuildBatches	region| string like |us-east-2	None
BatchGetBuilds	region| string like |us-east-2	None
BatchGetProjects	region| string like |us-east-2	None
GetResourcePolicy	region| string like |us-east-2	None

Write (16 of 28)

Action	Resource	Request condition
BatchDeleteBuilds	region| string like |us-east-2	None
CreateProject	region| string like |us-east-2	None
CreateWebhook	region| string like |us-east-2	None
DeleteBuildBatch	region| string like |us-east-2	None
DeleteProject	region| string like |us-east-2	None
DeleteWebhook	region| string like |us-east-2	None
InvalidateProjectCache	region| string like |us-east-2	None
RetryBuild	region| string like |us-east-2	None
RetryBuildBatch	region| string like |us-east-2	None
StartBuild	region| string like |us-east-2	None
StartBuildBatch	region| string like |us-east-2	None
StopBuild	region| string like |us-east-2	None
StopBuildBatch	region| string like |us-east-2	None
UpdateProject	region| string like |us-east-2	None
UpdateProjectVisibility	region| string like |us-east-2	None
UpdateWebhook	region| string like |us-east-2	None

List (2 of 14)

CodeBuild_ CodeCommit _ CodeDeploy StartBuild (寫入) 動作摘要：

< Actions StartBuild action in CodeBuild

Resource	Region	Account	Request condition
region| string like |us-east-2	us-east-2	123456789012	None

存取 IAM 資源所需的許可

資源是服務中的物件。IAM 資源包括群組、使用者、角色和政策。如果您使用 AWS 帳戶根使用者 憑證登入，則對管理 IAM 憑證或 IAM 資源沒有任何限制。但是，必須明確授予 IAM 使用者管理憑證或 IAM 資源的許可。您可以透過將以身分為基礎的政策連接到使用者來執行此操作。

> **Note**
>
> 在整份 AWS 文件中，當我們參考 IAM 政策而未提及任何特定類別時，我們指的是以身分識別為基礎的客戶管理政策。如需政策類別的詳細資訊，請參閱the section called “政策和許可”。

管理 IAM 身分的許可

管理 IAM 群組、使用者、角色和憑證所需的許可通常對應於任務的 API 動作。例如，若要建立 IAM 使用者，您必須擁有具有對應的 API 命令的 `iam:CreateUser` 許可：`CreateUser`。若要允許 IAM 使用者建立其他 IAM 使用者，您可以將 IAM 政策 (如下所示) 連接到該使用者：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

在政策中，`Resource` 元素的值取決於動作以及動作可以影響的資源。在上述範例中，政策可讓使用者建立任何使用者 (`*` 是符合所有字串的萬用字元)。反之，讓使用者僅變更自己的存取金鑰 (API 動作 `CreateAccessKey` 和 `UpdateAccessKey`) 的政策通常具有 `Resource` 元素。在這種情況下，ARN 包含一個變數 (`${aws:username}`)，其解析至目前使用者名稱，如以下範例所示：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListUsersForConsole",
            "Effect": "Allow",
            "Action": "iam:ListUsers",
```

```
            "Resource": "arn:aws:iam::*:*"
        },
        {
            "Sid": "ViewAndUpdateAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:UpdateAccessKey",
                "iam:CreateAccessKey",
                "iam:ListAccessKeys"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

在上述範例中，${aws:username} 是一個解析為目前使用者的使用者名稱的變數。如需有關政策變數的詳細資訊，請參閱 IAM 政策元素：變數與標籤。

在動作名稱中使用萬用字元 (*) 通常可讓您更輕鬆地授予許可給與特定任務相關的所有動作。例如，若要允許使用者執行任何 IAM 動作，您可以使用 iam:* 執行動作。為了允許使用者能夠執行只與存取金鑰相關的任何動作，您可以在政策陳述式的 iam:*AccessKey* 元素中使用 Action。如此可提供使用者許可來執行 CreateAccessKey、DeleteAccessKey、GetAccessKeyLastUsed、ListAccessKeys 和 UpdateAccessKey 動作。（如果將 future 將某個動作添加到 IAM 中，名稱中包含 AccessKey「」，則使iam:*AccessKey*用 to Action 元素也將授予用戶對該新動作的權限。）下列範例顯示的原則可讓使用者執行與自己的存取金鑰相關的所有動作 (以您*account-id*的 AWS 帳戶 ID 取代)：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/${aws:username}"
  }
}
```

有些任務 (例如刪除群組) 涉及多個動作：您必須先從群組中移除使用者，然後分開或刪除群組的政策，然後實際刪除該群組。如果您希望使用者能夠刪除群組，則必須確保授予使用者執行所有相關動作的許可。

在 AWS Management Console工作的許可

上述範例顯示允許使用者使用 AWS CLI 或 AWS 開發套件執行動作的政策。

當使用者使用主控台，主控台會向 IAM 發出請求，以列出群組、使用者、角色和政策，並獲得與群組、使用者或角色關聯的政策。主控台也會發出要求，以取得主體的相關資 AWS 帳戶 訊和資訊。主體是使用者在主控台中提出請求。

一般而言，要執行動作，您必須擁有只包含在政策中的符合動作。要建立使用者，您需要有呼叫 `CreateUser` 動作的許可。通常，當您使用主控台來執行動作時，您必須具有在主控台中顯示、列出、取得或以其他方式查看資源的許可。這是必要的，以便您可以在主控台中導覽以執行指定的動作。例如，如果使用者 Jorge 想要使用主控台來變更自己的存取金鑰，他會進入 IAM 主控台並選擇 Users (使用者)。此動作會導致主控台發出 `ListUsers` 請求。如果 Jorge 沒有 `iam:ListUsers` 動作的許可，則在嘗試列出使用者時，主控台將被拒絕存取。因此，Jorge 無法取得自己的名稱和自己的存取金鑰，即使他擁有 `CreateAccessKey` 和 `UpdateAccessKey` 動作的許可。

如果您想要授與使用者管理群組、使用者、角色、原則和認證的權限 AWS Management Console，您必須包含控制台執行之動作的權限。如需可以用來授予使用者許可的一些政策範例，請參閱管理 IAM 資源的政策範例。

跨 AWS 帳戶授予許可

您可以直接向自己帳戶中的 IAM 使用者授予對您的資源的存取權限。如果其他帳戶中的使用者需要存取您的資源，您可以建立 IAM 角色，該角色是一個擁有許可的實體，但不與特定使用者相關聯。然後，其他帳戶中的使用者可以使用該角色，並根據您為該角色分配的許可存取資源。如需詳細資訊，請參閱 在您擁有的另一個 AWS 帳戶 IAM 使用者中提供存取權。

> **Note**
>
> 部分服務支援以資源為基礎的政策，如 以身分為基礎和以資源為基礎的政策 (例如 Amazon S3、Amazon SNS 和 Amazon SQS) 中所述。對於這些服務，使用角色的替代方法是將政策連接到您要共用的資源 (儲存貯體、主題或佇列)。以資源為基礎的策略可以指定具有資源存取權限的 AWS 帳號。

由一個服務來存取另一個服務的許可

許多 AWS 服務訪問其他 AWS 服務。例如，多種 AWS 服務 (包括 Amazon EMR、Elastic Load Balancing 和 Amazon EC2 Auto Scaling) 可管理 Amazon EC2 執行個體。其他 AWS 服務利用 Amazon S3 存儲桶，Amazon SNS 主題，Amazon SQS 隊列等。

在這些情況下管理許可的方案因服務而異。以下是一些如何處理不同服務許可的範例：

- 在 Amazon EC2 Auto Scaling 中，使用者必須具有使用 Auto Scaling 的許可，但不需要明確授予管理 Amazon EC2 執行個體的許可。
- 在中 AWS Data Pipeline，IAM 角色決定管道可執行的動作；使用者需要權限才能擔任該角色。(有關詳細資訊，請參閱在《AWS Data Pipeline 開發人員指南》中的使用 IAM 授予管道許可)。

如需有關如何正確設定權限，讓 AWS 服務能夠完成您想要的工作的詳細資訊，請參閱您呼叫之服務的說明文件。若要了解如何為服務建立角色的詳細資訊，請參閱建立角色以將許可委派給 AWS 服務。

設定具有 IAM 角色的服務以代表您的工作

當您想要將 AWS 服務設定為代表您工作時，通常會為 IAM 角色提供 ARN，該角色定義允許服務執行的動作。 AWS 檢查以確保您具有將角色傳遞給服務的權限。如需詳細資訊，請參閱 授予使用者將角色傳遞至 AWS 服務的許可。

必要的動作

動作是您可以對資源執行的動作，例如查看、建立、編輯和刪除該資源。動作由每個 AWS 服務定義。

若要允許某人執行動作，您必須在適用於呼叫身分或受影響資源的政策中包含必要的動作。一般而言，要提供執行動作所需的許可，您必須在政策中包含該動作。例如，若要建立使用者，您需要將 CreateUser 動作新增至您的政策。

在某些情況下，動作可能要求您在政策中包含其他相關動作。例如，若要提供某人許可，使其可以使用 AWS Directory Service 操作在 `ds:CreateDirectory` 中建立目錄，您必須在其政策中包含下列動作：

- `ds:CreateDirectory`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:CreateSecurityGroup`

- `ec2:CreateNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:AuthorizeSecurityGroupEgress`

使用視覺化編輯器建立或編輯政策時，會收到警告和提示，以幫助您選擇政策的所有必需操作。

如需在中建立目錄所需權限的詳細資訊 AWS Directory Service，請參閱範例 2：允許使用者建立目錄。

管理 IAM 資源的政策範例

以下 IAM 政策範例允許使用者執行與管理 IAM 使用者、群組和憑證相關的任務。這包括允許使用者管理自己的密碼、存取金鑰和多重要素驗證 (MFA) 裝置的政策。

如需可讓使用者使用其他 AWS 服務 (例如 Amazon S3、Amazon EC2 和 DynamoDB) 執行任務的政策範例，請參閱。以身分為基礎的 IAM 政策範例

主題

- 允許使用者列出帳戶的群組、使用者、政策等，以供報告之用
- 允許使用者管理群組的成員資格
- 允許使用者管理 IAM 使用者
- 允許使用者設定帳戶密碼政策
- 允許使用者產生和擷取 IAM 憑證報告
- 允許所有 IAM 動作 (管理員存取)

允許使用者列出帳戶的群組、使用者、政策等，以供報告之用

以下政策允許使用者呼叫以字串 `Get` 或 `List` 開頭的任何 IAM 動作來產生報告。若要檢視範例政策，請參閱 IAM：允許對 IAM 主控台的唯讀存取權。

允許使用者管理群組的成員資格

下列原則可讓使用者更新呼叫之群組的成員資格MarketingGroup。若要檢視範例政策，請參閱 IAM：允許以程式設計方式及在主控台中管理群組的成員資格。

允許使用者管理 IAM 使用者

以下政策允許使用者執行所有與管理 IAM 使用者相關的任務，但是不允許對其他實體執行操作，如建立群組或政策。允許的動作包括：

- 建立使用者 (CreateUser 動作)。
- 刪除使用者。此任務需要許可以執行下列動作：DeleteSigningCertificate、DeleteLoginProfile、RemoveUserFromGroup 和 DeleteUser。
- 列出帳戶中的使用者和群組 (GetUser、ListUsers 和 ListGroupsForUser 動作)。
- 列出並移除使用者的政策 (ListUserPolicies、ListAttachedUserPolicies、DetachUserPolicy、DeleteUserPolicy 動作)
- 更改或變更使用者路徑 (UpdateUser 動作)。Resource 元素必須包含涵蓋來源路徑和目標路徑的 ARN。如需有關路徑的詳細資訊，請參閱 易用名稱和路徑。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUsersToPerformUserActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListPolicies",
                "iam:GetPolicy",
                "iam:UpdateUser",
                "iam:AttachUserPolicy",
                "iam:ListEntitiesForPolicy",
                "iam:DeleteUserPolicy",
                "iam:DeleteUser",
                "iam:ListUserPolicies",
                "iam:CreateUser",
                "iam:RemoveUserFromGroup",
                "iam:AddUserToGroup",
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:PutUserPolicy",
                "iam:ListAttachedUserPolicies",
                "iam:ListUsers",
```

```
                "iam:GetUser",
                "iam:DetachUserPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUsersToSeeStatsOnIAMConsoleDashboard",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccount*",
                "iam:ListAccount*"
            ],
            "Resource": "*"
        }
    ]
}
```

前述政策中包含的多個許可，允許使用者在 AWS Management Console中執行任務。從 AWS CLI、AWS 開發套件或 IAM HTTP 查詢 API 執行使用者相關任務的使用者可能不需要特定許可。例如，如果使用者已知道從使用者取消連接的 ARN，則不需要 `iam:ListAttachedUserPolicies` 許可。使用者所需許可的確切清單取決於使用者管理其他使用者時必須執行的任務。

以下政策中的許可允許透過 AWS Management Console存取使用者任務：

- `iam:GetAccount*`
- `iam:ListAccount*`

允許使用者設定帳戶密碼政策

您可以授予某些使用者取得和更新您 AWS 帳戶密碼政策的許可。若要檢視範例政策，請參閱 IAM：允許以程式設計方式在主控台中設定帳戶密碼要求。

允許使用者產生和擷取 IAM 憑證報告

您可以授與使用者產生和下載列出您的所有使用者的報告的權限 AWS 帳戶。此報告也會列出各種使用者憑證的狀態，包括密碼、存取金鑰、MFA 裝置和簽章憑證。如需有關憑證報告的詳細資訊，請參閱取得 AWS 帳戶的憑證報告。若要檢視範例政策，請參閱 IAM：產生和擷取 IAM 憑證報告。

允許所有 IAM 動作 (管理員存取)

您可以授予某些使用者在 IAM 中執行所有操作的管理員許可，包括管理密碼、存取金鑰、MFA 裝置和使用者憑證。以下範例政策授予這些許可。

> **Warning**
>
> 當您授予使用者 IAM 的完全存取權時，對於使用者可以向自己或他人授予的許可則沒有限制。使用者可以建立新的 IAM 實體 (使用者或角色) 並授予這些實體對您 AWS 帳戶中所有資源的完全存取權限。您授予使用者對 IAM 的完全存取權限時，實際上是授予使用者對您的 AWS 帳戶中所有資源的完全存取權限。這包括可刪除所有資源的許可。您應只將這些許可授予信任的管理員，也應該對這些管理員強制採用多重要素驗證 (MFA)。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*",
    "Resource": "*"
  }
}
```

使用 AWS 軟體開發套件的 IAM 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 IAM。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

程式碼範例

- 搭DeleteAccountAlias配 AWS 開發套件或 CLI 使用
- 搭DeleteAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteGroup配 AWS 開發套件或 CLI 使用
- 搭DeleteGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭DeleteLoginProfile配 AWS 開發套件或 CLI 使用
- 搭DeleteOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭DeletePolicy配 AWS 開發套件或 CLI 使用
- 搭DeletePolicyVersion配 AWS 開發套件或 CLI 使用
- 搭DeleteRole配 AWS 開發套件或 CLI 使用
- 搭DeleteRolePermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭DeleteRolePolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteSAMLProvider配 AWS 開發套件或 CLI 使用
- 搭DeleteServerCertificate配 AWS 開發套件或 CLI 使用
- 搭DeleteServiceLinkedRole配 AWS 開發套件或 CLI 使用
- 搭DeleteSigningCertificate配 AWS 開發套件或 CLI 使用
- 搭DeleteUser配 AWS 開發套件或 CLI 使用
- 搭DeleteUserPermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭DeleteUserPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteVirtualMfaDevice配 AWS 開發套件或 CLI 使用
- 搭DetachGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭DetachRolePolicy配 AWS 開發套件或 CLI 使用
- 搭DetachUserPolicy配 AWS 開發套件或 CLI 使用
- 搭EnableMfaDevice配 AWS 開發套件或 CLI 使用
- 搭GenerateCredentialReport配 AWS 開發套件或 CLI 使用
- 搭GenerateServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用
- 搭GetAccessKeyLastUsed配 AWS 開發套件或 CLI 使用
- 搭GetAccountAuthorizationDetails配 AWS 開發套件或 CLI 使用
- 搭GetAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭GetAccountSummary配 AWS 開發套件或 CLI 使用

- 搭GetContextKeysForCustomPolicy配 AWS 開發套件或 CLI 使用
- 搭GetContextKeysForPrincipalPolicy配 AWS 開發套件或 CLI 使用
- 搭GetCredentialReport配 AWS 開發套件或 CLI 使用
- 搭GetGroup配 AWS 開發套件或 CLI 使用
- 搭GetGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭GetInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭GetLoginProfile配 AWS 開發套件或 CLI 使用
- 搭GetOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭GetPolicy配 AWS 開發套件或 CLI 使用
- 搭GetPolicyVersion配 AWS 開發套件或 CLI 使用
- 搭GetRole配 AWS 開發套件或 CLI 使用
- 搭GetRolePolicy配 AWS 開發套件或 CLI 使用
- 搭GetSamlProvider配 AWS 開發套件或 CLI 使用
- 搭GetServerCertificate配 AWS 開發套件或 CLI 使用
- 搭GetServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用
- 搭GetServiceLastAccessedDetailsWithEntities配 AWS 開發套件或 CLI 使用
- 搭GetServiceLinkedRoleDeletionStatus配 AWS 開發套件或 CLI 使用
- 搭GetUser配 AWS 開發套件或 CLI 使用
- 搭GetUserPolicy配 AWS 開發套件或 CLI 使用
- 搭ListAccessKeys配 AWS 開發套件或 CLI 使用
- 搭ListAccountAliases配 AWS 開發套件或 CLI 使用
- 搭ListAttachedGroupPolicies配 AWS 開發套件或 CLI 使用
- 搭ListAttachedRolePolicies配 AWS 開發套件或 CLI 使用
- 搭ListAttachedUserPolicies配 AWS 開發套件或 CLI 使用
- 搭ListEntitiesForPolicy配 AWS 開發套件或 CLI 使用
- 搭ListGroupPolicies配 AWS 開發套件或 CLI 使用
- 搭ListGroups配 AWS 開發套件或 CLI 使用
- 搭ListGroupsForUser配 AWS 開發套件或 CLI 使用
- 搭ListInstanceProfiles配 AWS 開發套件或 CLI 使用
- 搭ListInstanceProfilesForRole配 AWS 開發套件或 CLI 使用

- 搭ListMfaDevices配 AWS 開發套件或 CLI 使用
- 搭ListOpenIdConnectProviders配 AWS 開發套件或 CLI 使用
- 搭ListPolicies配 AWS 開發套件或 CLI 使用
- 搭ListPolicyVersions配 AWS 開發套件或 CLI 使用
- 搭ListRolePolicies配 AWS 開發套件或 CLI 使用
- 搭ListRoleTags配 AWS 開發套件或 CLI 使用
- 搭ListRoles配 AWS 開發套件或 CLI 使用
- 搭ListSAMLProviders配 AWS 開發套件或 CLI 使用
- 搭ListServerCertificates配 AWS 開發套件或 CLI 使用
- 搭ListSigningCertificates配 AWS 開發套件或 CLI 使用
- 搭ListUserPolicies配 AWS 開發套件或 CLI 使用
- 搭ListUserTags配 AWS 開發套件或 CLI 使用
- 搭ListUsers配 AWS 開發套件或 CLI 使用
- 搭ListVirtualMfaDevices配 AWS 開發套件或 CLI 使用
- 搭PutGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭PutRolePermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭PutRolePolicy配 AWS 開發套件或 CLI 使用
- 搭PutUserPermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭PutUserPolicy配 AWS 開發套件或 CLI 使用
- 搭RemoveClientIdFromOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭RemoveRoleFromInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭RemoveUserFromGroup配 AWS 開發套件或 CLI 使用
- 搭ResyncMfaDevice配 AWS 開發套件或 CLI 使用
- 搭SetDefaultPolicyVersion配 AWS 開發套件或 CLI 使用
- 搭TagRole配 AWS 開發套件或 CLI 使用
- 搭TagUser配 AWS 開發套件或 CLI 使用
- 搭UntagRole配 AWS 開發套件或 CLI 使用
- 搭UntagUser配 AWS 開發套件或 CLI 使用
- 搭UpdateAccessKey配 AWS 開發套件或 CLI 使用
- 搭UpdateAccountPasswordPolicy配 AWS 開發套件或 CLI 使用

- 搭UpdateAssumeRolePolicy配 AWS 開發套件或 CLI 使用
- 搭UpdateGroup配 AWS 開發套件或 CLI 使用
- 搭UpdateLoginProfile配 AWS 開發套件或 CLI 使用
- 搭UpdateOpenIdConnectProviderThumbprint配 AWS 開發套件或 CLI 使用
- 搭UpdateRole配 AWS 開發套件或 CLI 使用
- 搭UpdateRoleDescription配 AWS 開發套件或 CLI 使用
- 搭UpdateSamlProvider配 AWS 開發套件或 CLI 使用
- 搭UpdateServerCertificate配 AWS 開發套件或 CLI 使用
- 搭UpdateSigningCertificate配 AWS 開發套件或 CLI 使用
- 搭UpdateUser配 AWS 開發套件或 CLI 使用
- 搭UploadServerCertificate配 AWS 開發套件或 CLI 使用
- 搭UploadSigningCertificate配 AWS 開發套件或 CLI 使用

- 使用 AWS 軟體開發套件的 IAM 案例
  - 使用 AWS SDK 建置及管理彈性服務
  - 建立 IAM 群組，並使用 AWS SDK 將使用者新增至群組
  - 使用 AWS SDK 建立 IAM AWS STS 使用者並擔任角色
  - 使用 SDK 建立唯讀和讀寫 IAM 使用者 AWS
  - 使用 AWS SDK 管理 IAM 存取金鑰
  - 使用 AWS 開發套件管理 IAM 政策
  - 使用 AWS SDK 管理 IAM 角色
  - 使用 AWS SDK 管理您的 IAM 帳戶
  - 使用 AWS SDK 復原 IAM 政策版本
  - 使用 SDK 使用 IAM 政策產生器 AWS API

- AWS STS 使用 AWS SDK 的程式碼範例
  - AWS STS 使用 AWS SDK 的動作
    - 搭AssumeRole配 AWS 開發套件或 CLI 使用
    - 搭AssumeRoleWithWebIdentity配 AWS 開發套件或 CLI 使用
    - 搭DecodeAuthorizationMessage配 AWS 開發套件或 CLI 使用
    - 搭GetFederationToken配 AWS 開發套件或 CLI 使用
    - 搭GetSessionToken配 AWS 開發套件或 CLI 使用

- AWS STS 使用 AWS SDK 的案例
  - 假設需要使用 SDK 的 MFA 權杖的 AWS STS IAM AWS 角色
  - 使用 SDK AWS STS 為同盟使用者建構 URL AWS
  - AWS STS 使用 SDK 獲取需要 MFA 令牌的會話令 AWS 牌

使用 AWS 軟體開發套件的 IAM 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 IAM。

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。

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 IAM。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

Hello IAM

下列程式碼範例示範如何開始使用 IAM。

.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 IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
```

```
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListPolicies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 iam)
```

```
# Set this project's name.
project("hello_iam")

# 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_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

iam.cpp 來源檔案的程式碼。

```
#include <aws/core/Aws.h>
```

```
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and
 Access Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
 *
 */

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.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
```

```
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) <<
 policy.GetArn() <<
                          std::setw(64) << policy.GetDescription() <<
 std::setw(12) <<
                          policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                          std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ListPolicies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"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
 Management (IAM)
// client and list up to 10 policies 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
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(context.TODO(), &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
```

```
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListPolicies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.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
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 HelloIAM {
```

```
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListPolicies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。

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
```

```
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      page.Policies.forEach((p) => {
        console.log(`${p.PolicyName}`);
        policyCount++;
      });
    }
  }
  console.log(`Found ${policyCount} policies.`);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListPolicies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。

來自 src/bin/hello.rs。

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
```

```
struct HelloScenarioArgs {
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

來自 src/iam-service-lib.rs。

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
```

```
        .collect();

    Ok(policy_names)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListPolicies中的 Rust API 參考資料。

程式碼範例

- 使用 AWS 開發套件執行 IAM 的動作
  - 搭AddClientIdToOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
  - 搭AddRoleToInstanceProfile配 AWS 開發套件或 CLI 使用
  - 搭AddUserToGroup配 AWS 開發套件或 CLI 使用
  - 搭AttachGroupPolicy配 AWS 開發套件或 CLI 使用
  - 搭AttachRolePolicy配 AWS 開發套件或 CLI 使用
  - 搭AttachUserPolicy配 AWS 開發套件或 CLI 使用
  - 搭ChangePassword配 AWS 開發套件或 CLI 使用
  - 搭CreateAccessKey配 AWS 開發套件或 CLI 使用
  - 搭CreateAccountAlias配 AWS 開發套件或 CLI 使用
  - 搭CreateGroup配 AWS 開發套件或 CLI 使用
  - 搭CreateInstanceProfile配 AWS 開發套件或 CLI 使用
  - 搭CreateLoginProfile配 AWS 開發套件或 CLI 使用
  - 搭CreateOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
  - 搭CreatePolicy配 AWS 開發套件或 CLI 使用
  - 搭CreatePolicyVersion配 AWS 開發套件或 CLI 使用
  - 搭CreateRole配 AWS 開發套件或 CLI 使用
  - 搭CreateSAMLProvider配 AWS 開發套件或 CLI 使用
  - 搭CreateServiceLinkedRole配 AWS 開發套件或 CLI 使用
  - 搭CreateUser配 AWS 開發套件或 CLI 使用
  - 搭CreateVirtualMfaDevice配 AWS 開發套件或 CLI 使用
  - 搭DeactivateMfaDevice配 AWS 開發套件或 CLI 使用
  - 搭DeleteAccessKey配 AWS 開發套件或 CLI 使用

- 搭DeleteAccountAlias配 AWS 開發套件或 CLI 使用
- 搭DeleteAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteGroup配 AWS 開發套件或 CLI 使用
- 搭DeleteGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭DeleteLoginProfile配 AWS 開發套件或 CLI 使用
- 搭DeleteOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭DeletePolicy配 AWS 開發套件或 CLI 使用
- 搭DeletePolicyVersion配 AWS 開發套件或 CLI 使用
- 搭DeleteRole配 AWS 開發套件或 CLI 使用
- 搭DeleteRolePermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭DeleteRolePolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteSAMLProvider配 AWS 開發套件或 CLI 使用
- 搭DeleteServerCertificate配 AWS 開發套件或 CLI 使用
- 搭DeleteServiceLinkedRole配 AWS 開發套件或 CLI 使用
- 搭DeleteSigningCertificate配 AWS 開發套件或 CLI 使用
- 搭DeleteUser配 AWS 開發套件或 CLI 使用
- 搭DeleteUserPermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭DeleteUserPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteVirtualMfaDevice配 AWS 開發套件或 CLI 使用
- 搭DetachGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭DetachRolePolicy配 AWS 開發套件或 CLI 使用
- 搭DetachUserPolicy配 AWS 開發套件或 CLI 使用
- 搭EnableMfaDevice配 AWS 開發套件或 CLI 使用
- 搭GenerateCredentialReport配 AWS 開發套件或 CLI 使用
- 搭GenerateServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用
- 搭GetAccessKeyLastUsed配 AWS 開發套件或 CLI 使用
- 搭GetAccountAuthorizationDetails配 AWS 開發套件或 CLI 使用
- 搭GetAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭GetAccountSummary配 AWS 開發套件或 CLI 使用

- 搭GetContextKeysForCustomPolicy配 AWS 開發套件或 CLI 使用
- 搭GetContextKeysForPrincipalPolicy配 AWS 開發套件或 CLI 使用
- 搭GetCredentialReport配 AWS 開發套件或 CLI 使用
- 搭GetGroup配 AWS 開發套件或 CLI 使用
- 搭GetGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭GetInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭GetLoginProfile配 AWS 開發套件或 CLI 使用
- 搭GetOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭GetPolicy配 AWS 開發套件或 CLI 使用
- 搭GetPolicyVersion配 AWS 開發套件或 CLI 使用
- 搭GetRole配 AWS 開發套件或 CLI 使用
- 搭GetRolePolicy配 AWS 開發套件或 CLI 使用
- 搭GetSamlProvider配 AWS 開發套件或 CLI 使用
- 搭GetServerCertificate配 AWS 開發套件或 CLI 使用
- 搭GetServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用
- 搭GetServiceLastAccessedDetailsWithEntities配 AWS 開發套件或 CLI 使用
- 搭GetServiceLinkedRoleDeletionStatus配 AWS 開發套件或 CLI 使用
- 搭GetUser配 AWS 開發套件或 CLI 使用
- 搭GetUserPolicy配 AWS 開發套件或 CLI 使用
- 搭ListAccessKeys配 AWS 開發套件或 CLI 使用
- 搭ListAccountAliases配 AWS 開發套件或 CLI 使用
- 搭ListAttachedGroupPolicies配 AWS 開發套件或 CLI 使用
- 搭ListAttachedRolePolicies配 AWS 開發套件或 CLI 使用
- 搭ListAttachedUserPolicies配 AWS 開發套件或 CLI 使用
- 搭ListEntitiesForPolicy配 AWS 開發套件或 CLI 使用
- 搭ListGroupPolicies配 AWS 開發套件或 CLI 使用
- 搭ListGroups配 AWS 開發套件或 CLI 使用
- 搭ListGroupsForUser配 AWS 開發套件或 CLI 使用
- 搭ListInstanceProfiles配 AWS 開發套件或 CLI 使用
- 搭ListInstanceProfilesForRole配 AWS 開發套件或 CLI 使用

- 搭ListMfaDevices配 AWS 開發套件或 CLI 使用
- 搭ListOpenIdConnectProviders配 AWS 開發套件或 CLI 使用
- 搭ListPolicies配 AWS 開發套件或 CLI 使用
- 搭ListPolicyVersions配 AWS 開發套件或 CLI 使用
- 搭ListRolePolicies配 AWS 開發套件或 CLI 使用
- 搭ListRoleTags配 AWS 開發套件或 CLI 使用
- 搭ListRoles配 AWS 開發套件或 CLI 使用
- 搭ListSAMLProviders配 AWS 開發套件或 CLI 使用
- 搭ListServerCertificates配 AWS 開發套件或 CLI 使用
- 搭ListSigningCertificates配 AWS 開發套件或 CLI 使用
- 搭ListUserPolicies配 AWS 開發套件或 CLI 使用
- 搭ListUserTags配 AWS 開發套件或 CLI 使用
- 搭ListUsers配 AWS 開發套件或 CLI 使用
- 搭ListVirtualMfaDevices配 AWS 開發套件或 CLI 使用
- 搭PutGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭PutRolePermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭PutRolePolicy配 AWS 開發套件或 CLI 使用
- 搭PutUserPermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭PutUserPolicy配 AWS 開發套件或 CLI 使用
- 搭RemoveClientIdFromOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭RemoveRoleFromInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭RemoveUserFromGroup配 AWS 開發套件或 CLI 使用
- 搭ResyncMfaDevice配 AWS 開發套件或 CLI 使用
- 搭SetDefaultPolicyVersion配 AWS 開發套件或 CLI 使用
- 搭TagRole配 AWS 開發套件或 CLI 使用
- 搭TagUser配 AWS 開發套件或 CLI 使用
- 搭UntagRole配 AWS 開發套件或 CLI 使用
- 搭UntagUser配 AWS 開發套件或 CLI 使用
- 搭UpdateAccessKey配 AWS 開發套件或 CLI 使用
- 搭UpdateAccountPasswordPolicy配 AWS 開發套件或 CLI 使用

- 搭UpdateAssumeRolePolicy配 AWS 開發套件或 CLI 使用
- 搭UpdateGroup配 AWS 開發套件或 CLI 使用
- 搭UpdateLoginProfile配 AWS 開發套件或 CLI 使用
- 搭UpdateOpenIdConnectProviderThumbprint配 AWS 開發套件或 CLI 使用
- 搭UpdateRole配 AWS 開發套件或 CLI 使用
- 搭UpdateRoleDescription配 AWS 開發套件或 CLI 使用
- 搭UpdateSamlProvider配 AWS 開發套件或 CLI 使用
- 搭UpdateServerCertificate配 AWS 開發套件或 CLI 使用
- 搭UpdateSigningCertificate配 AWS 開發套件或 CLI 使用
- 搭UpdateUser配 AWS 開發套件或 CLI 使用
- 搭UploadServerCertificate配 AWS 開發套件或 CLI 使用
- 搭UploadSigningCertificate配 AWS 開發套件或 CLI 使用

- 使用 AWS 軟體開發套件的 IAM 案例
  - 使用 AWS SDK 建置及管理彈性服務
  - 建立 IAM 群組，並使用 AWS SDK 將使用者新增至群組
  - 使用 AWS SDK 建立 IAM AWS STS 使用者並擔任角色
  - 使用 SDK 建立唯讀和讀寫 IAM 使用者 AWS
  - 使用 AWS SDK 管理 IAM 存取金鑰
  - 使用 AWS 開發套件管理 IAM 政策
  - 使用 AWS SDK 管理 IAM 角色
  - 使用 AWS SDK 管理您的 IAM 帳戶
  - 使用 AWS SDK 復原 IAM 政策版本
  - 使用 SDK 使用 IAM 政策產生器 AWS API

使用 AWS 開發套件執行 IAM 的動作

下列程式碼範例示範如何使用 AWS SDK 執行個別 IAM 動作。這些摘錄會呼叫 IAM 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。如需完整清單，請參閱《AWS Identity and Access Management (IAM) API 參考》。

範例

- 搭AddClientIdToOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭AddRoleToInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭AddUserToGroup配 AWS 開發套件或 CLI 使用
- 搭AttachGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭AttachRolePolicy配 AWS 開發套件或 CLI 使用
- 搭AttachUserPolicy配 AWS 開發套件或 CLI 使用
- 搭ChangePassword配 AWS 開發套件或 CLI 使用
- 搭CreateAccessKey配 AWS 開發套件或 CLI 使用
- 搭CreateAccountAlias配 AWS 開發套件或 CLI 使用
- 搭CreateGroup配 AWS 開發套件或 CLI 使用
- 搭CreateInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭CreateLoginProfile配 AWS 開發套件或 CLI 使用
- 搭CreateOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭CreatePolicy配 AWS 開發套件或 CLI 使用
- 搭CreatePolicyVersion配 AWS 開發套件或 CLI 使用
- 搭CreateRole配 AWS 開發套件或 CLI 使用
- 搭CreateSAMLProvider配 AWS 開發套件或 CLI 使用
- 搭CreateServiceLinkedRole配 AWS 開發套件或 CLI 使用
- 搭CreateUser配 AWS 開發套件或 CLI 使用
- 搭CreateVirtualMfaDevice配 AWS 開發套件或 CLI 使用
- 搭DeactivateMfaDevice配 AWS 開發套件或 CLI 使用
- 搭DeleteAccessKey配 AWS 開發套件或 CLI 使用
- 搭DeleteAccountAlias配 AWS 開發套件或 CLI 使用
- 搭DeleteAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteGroup配 AWS 開發套件或 CLI 使用
- 搭DeleteGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteInstanceProfile配 AWS 開發套件或 CLI 使用

- 搭DeleteLoginProfile配 AWS 開發套件或 CLI 使用
- 搭DeleteOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭DeletePolicy配 AWS 開發套件或 CLI 使用
- 搭DeletePolicyVersion配 AWS 開發套件或 CLI 使用
- 搭DeleteRole配 AWS 開發套件或 CLI 使用
- 搭DeleteRolePermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭DeleteRolePolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteSAMLProvider配 AWS 開發套件或 CLI 使用
- 搭DeleteServerCertificate配 AWS 開發套件或 CLI 使用
- 搭DeleteServiceLinkedRole配 AWS 開發套件或 CLI 使用
- 搭DeleteSigningCertificate配 AWS 開發套件或 CLI 使用
- 搭DeleteUser配 AWS 開發套件或 CLI 使用
- 搭DeleteUserPermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭DeleteUserPolicy配 AWS 開發套件或 CLI 使用
- 搭DeleteVirtualMfaDevice配 AWS 開發套件或 CLI 使用
- 搭DetachGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭DetachRolePolicy配 AWS 開發套件或 CLI 使用
- 搭DetachUserPolicy配 AWS 開發套件或 CLI 使用
- 搭EnableMfaDevice配 AWS 開發套件或 CLI 使用
- 搭GenerateCredentialReport配 AWS 開發套件或 CLI 使用
- 搭GenerateServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用
- 搭GetAccessKeyLastUsed配 AWS 開發套件或 CLI 使用
- 搭GetAccountAuthorizationDetails配 AWS 開發套件或 CLI 使用
- 搭GetAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭GetAccountSummary配 AWS 開發套件或 CLI 使用
- 搭GetContextKeysForCustomPolicy配 AWS 開發套件或 CLI 使用
- 搭GetContextKeysForPrincipalPolicy配 AWS 開發套件或 CLI 使用
- 搭GetCredentialReport配 AWS 開發套件或 CLI 使用
- 搭GetGroup配 AWS 開發套件或 CLI 使用

- 搭GetGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭GetInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭GetLoginProfile配 AWS 開發套件或 CLI 使用
- 搭GetOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭GetPolicy配 AWS 開發套件或 CLI 使用
- 搭GetPolicyVersion配 AWS 開發套件或 CLI 使用
- 搭GetRole配 AWS 開發套件或 CLI 使用
- 搭GetRolePolicy配 AWS 開發套件或 CLI 使用
- 搭GetSamlProvider配 AWS 開發套件或 CLI 使用
- 搭GetServerCertificate配 AWS 開發套件或 CLI 使用
- 搭GetServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用
- 搭GetServiceLastAccessedDetailsWithEntities配 AWS 開發套件或 CLI 使用
- 搭GetServiceLinkedRoleDeletionStatus配 AWS 開發套件或 CLI 使用
- 搭GetUser配 AWS 開發套件或 CLI 使用
- 搭GetUserPolicy配 AWS 開發套件或 CLI 使用
- 搭ListAccessKeys配 AWS 開發套件或 CLI 使用
- 搭ListAccountAliases配 AWS 開發套件或 CLI 使用
- 搭ListAttachedGroupPolicies配 AWS 開發套件或 CLI 使用
- 搭ListAttachedRolePolicies配 AWS 開發套件或 CLI 使用
- 搭ListAttachedUserPolicies配 AWS 開發套件或 CLI 使用
- 搭ListEntitiesForPolicy配 AWS 開發套件或 CLI 使用
- 搭ListGroupPolicies配 AWS 開發套件或 CLI 使用
- 搭ListGroups配 AWS 開發套件或 CLI 使用
- 搭ListGroupsForUser配 AWS 開發套件或 CLI 使用
- 搭ListInstanceProfiles配 AWS 開發套件或 CLI 使用
- 搭ListInstanceProfilesForRole配 AWS 開發套件或 CLI 使用
- 搭ListMfaDevices配 AWS 開發套件或 CLI 使用
- 搭ListOpenIdConnectProviders配 AWS 開發套件或 CLI 使用
- 搭ListPolicies配 AWS 開發套件或 CLI 使用

- 搭ListPolicyVersions配 AWS 開發套件或 CLI 使用
- 搭ListRolePolicies配 AWS 開發套件或 CLI 使用
- 搭ListRoleTags配 AWS 開發套件或 CLI 使用
- 搭ListRoles配 AWS 開發套件或 CLI 使用
- 搭ListSAMLProviders配 AWS 開發套件或 CLI 使用
- 搭ListServerCertificates配 AWS 開發套件或 CLI 使用
- 搭ListSigningCertificates配 AWS 開發套件或 CLI 使用
- 搭ListUserPolicies配 AWS 開發套件或 CLI 使用
- 搭ListUserTags配 AWS 開發套件或 CLI 使用
- 搭ListUsers配 AWS 開發套件或 CLI 使用
- 搭ListVirtualMfaDevices配 AWS 開發套件或 CLI 使用
- 搭PutGroupPolicy配 AWS 開發套件或 CLI 使用
- 搭PutRolePermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭PutRolePolicy配 AWS 開發套件或 CLI 使用
- 搭PutUserPermissionsBoundary配 AWS 開發套件或 CLI 使用
- 搭PutUserPolicy配 AWS 開發套件或 CLI 使用
- 搭RemoveClientIdFromOpenIdConnectProvider配 AWS 開發套件或 CLI 使用
- 搭RemoveRoleFromInstanceProfile配 AWS 開發套件或 CLI 使用
- 搭RemoveUserFromGroup配 AWS 開發套件或 CLI 使用
- 搭ResyncMfaDevice配 AWS 開發套件或 CLI 使用
- 搭SetDefaultPolicyVersion配 AWS 開發套件或 CLI 使用
- 搭TagRole配 AWS 開發套件或 CLI 使用
- 搭TagUser配 AWS 開發套件或 CLI 使用
- 搭UntagRole配 AWS 開發套件或 CLI 使用
- 搭UntagUser配 AWS 開發套件或 CLI 使用
- 搭UpdateAccessKey配 AWS 開發套件或 CLI 使用
- 搭UpdateAccountPasswordPolicy配 AWS 開發套件或 CLI 使用
- 搭UpdateAssumeRolePolicy配 AWS 開發套件或 CLI 使用
- 搭UpdateGroup配 AWS 開發套件或 CLI 使用

- 搭UpdateLoginProfile配 AWS 開發套件或 CLI 使用
- 搭UpdateOpenIdConnectProviderThumbprint配 AWS 開發套件或 CLI 使用
- 搭UpdateRole配 AWS 開發套件或 CLI 使用
- 搭UpdateRoleDescription配 AWS 開發套件或 CLI 使用
- 搭UpdateSamlProvider配 AWS 開發套件或 CLI 使用
- 搭UpdateServerCertificate配 AWS 開發套件或 CLI 使用
- 搭UpdateSigningCertificate配 AWS 開發套件或 CLI 使用
- 搭UpdateUser配 AWS 開發套件或 CLI 使用
- 搭UploadServerCertificate配 AWS 開發套件或 CLI 使用
- 搭UploadSigningCertificate配 AWS 開發套件或 CLI 使用

搭**AddClientIdToOpenIdConnectProvider**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

將用戶端 ID (對象) 新增至開放式 ID Connect (OIDC) 提供者

下列add-client-id-to-open-id-connect-provider命令會將用戶端識別碼新增my-application-ID至名為的 OIDC 提供者。server.example.com

```
aws iam add-client-id-to-open-id-connect-provider \
    --client-id my-application-ID \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

此命令不會產生輸出。

若要建立 OIDC 提供者，請使用指create-open-id-connect-provider令。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的建立 OpenID Connect (OIDC) 身分識別提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AddClientIdToOpenIdConnectProvider中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會將用戶端 ID (或對象) 新增**my-application-ID**至名為的現有 OIDC 提供者。**server.example.com**

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程AddClientIdToOpenIdConnectProvider式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AddRoleToInstanceProfile**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

新增角色至執行個體設定檔

下列add-role-to-instance-profile命令會將名為的角色新增S3Access至名為的執行個體設定檔Webserver。

```
aws iam add-role-to-instance-profile \
    --role-name S3Access \
    --instance-profile-name Webserver
```

此命令不會產生輸出。

若要建立例證設定檔，請使用create-instance-profile指令。

如需更多資訊，請參閱 AWS 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 API 詳細資訊，請參閱AWS CLI 命令參考中的AddRoleToInstance設定檔。

PowerShell

用於的工具 PowerShell

範例 1：此命令會將名為的角色新增**S3Access**至名為的現有執行個體設定檔**webserver**。若要建立例證設定檔，請使用**New-IAMInstanceProfile**指令。建立執行個體設定檔並使用此命令將其與角色建立關聯後，您可以將其附加到 EC2 執行個體。若要這麼做，請使用**New-EC2Instance**指令程式搭配**InstanceProfile_Arn**或**InstanceProfile-Name**參數來啟動新的執行個體。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的AddRoleToInstance設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AddUserToGroup**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AddUserToGroup。

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：

- 建立群組並新增使用者。

.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。

```
    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AddUserToGroup中的。

CLI

AWS CLI

將使用者新增至 IAM 群組

下列 `add-user-to-group` 命令會將名為 Bob 的 IAM 使用者新增至名為 Admins 的 IAM 群組。

```
aws iam add-user-to-group \
    --user-name Bob \
    --group-name Admins
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的在 IAM 使用者群組中新增和移除使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AddUserToGroup中的。

PowerShell

適用的工具 PowerShell

範例 1：此指令會將名為的使用者新增**Bob**至名為的群組**Admins**。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程AddUserToGroup式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AttachGroupPolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AttachGroupPolicy。

CLI

AWS CLI

將受管政策附加到 IAM 群組

下列attach-group-policy命令會將名為的 AWS 受管政策附加ReadOnlyAccess到名為的 IAM 群組Finance。

```
aws iam attach-group-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --group-name Finance
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的受管政策和內嵌政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的AttachGroup政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將名為的客戶受管政策附加**TesterPolicy**至 IAM 群組**Testers**。該群組中的使用者會立即受到該原則預設版本中定義的權限影響。

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

範例 2：此範例會將名為的 AWS 受管政策附加**AdministratorAccess**至 IAM 群組**Admins**。該群組中的使用者會立即受到該原則最新版本中定義的權限影響。

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的AttachGroup原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AttachRolePolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用AttachRolePolicy。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 管理角色

.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。

```
    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的AttachRole政策。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
```

```
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam attach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
  return 1
```

```
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的AttachRole政策。

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。

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                      roleName << ": " << list_outcome.GetError().GetMessage() <<
                      std::endl;
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
                        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
```

```
                                return policy.GetPolicyArn() == policyArn;
                            })) {
                std::cout << "Policy " << policyArn <<
                          " is already attached to role " << roleName << std::endl;
                return true;
            }

            done = !list_outcome.GetResult().GetIsTruncated();
            list_request.SetMarker(list_outcome.GetResult().GetMarker());
        }

        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(roleName);
        request.SetPolicyArn(policyArn);

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
     iam.AttachRolePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                      roleName << ": " << outcome.GetError().GetMessage() <<
     std::endl;
        }
        else {
            std::cout << "Successfully attached policy " << policyArn << " to role "
     <<
                      roleName << std::endl;
        }

        return outcome.IsSuccess();
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的AttachRole政策。

CLI

AWS CLI

將受管政策連接至 IAM 角色

下列attach-role-policy命令會將名為的 AWS 受管政策附加ReadOnlyAccess到名為的 IAM 角色ReadOnlyRole。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --role-name ReadOnlyRole
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的受管政策和內嵌政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的AttachRole政策。

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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
 error {
 _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
 &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
```

```
 return err
}
```

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

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.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import
 software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
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 AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s
```

```
                Where:
                    roleName - A role name that you can obtain from the AWS
Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
Management Console.\s
                """;

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

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
```

```
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的AttachRole政策。

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。

連接政策。

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 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AttachRole政策。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
      }
    });
  }
});
```

- 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AttachRole政策。

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 attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String
) {
```

```
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

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

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。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
```

```
    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的AttachRole政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將名為的 AWS 受管政策附加SecurityAudit至 IAM 角色CoSecurityAuditors。承擔該角色的使用者會立即受到該原則最新版本中定義的權限影響。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的AttachRole原則。

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。

使用 Boto3 Policy 物件將政策連接至角色。

```
def attach_to_role(role_name, policy_arn):
```

```
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

使用 Boto3 Role 物件將政策連接至角色。

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的AttachRole政策 (Boto3) 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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、附加和解除連結角色原則。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的AttachRole政策。

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。

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的 Rust API 參考AttachRole政策。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func attachRolePolicy(role: String, policyArn: String) async throws {
    let input = AttachRolePolicyInput(
        policyArn: policyArn,
        roleName: role
    )
    do {
        _ = try await client.attachRolePolicy(input: input)
    } catch {
        throw error
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的AttachRole政策以取得 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`AttachUserPolicy`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`AttachUserPolicy`。

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：

- 建立唯讀和讀寫的使用者

CLI

AWS CLI

將受管政策連接至 IAM 使用者

下列attach-user-policy命令會將名為的 AWS 受管政策附加AdministratorAccess至名為的 IAM 使用者Alice。

```
aws iam attach-user-policy \
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
    --user-name Alice
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的受管政策和內嵌政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的AttachUser政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將名為的 AWS 受管政策附加**AmazonCognitoPowerUser**至 IAM 使用者**Bob**。使用者會立即受到該原則最新版本中定義的權限影響。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的AttachUser原則。

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。

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的AttachUser政策 (Boto3) 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 設定和執行程式碼範例儲存庫。

```
  # Attaches a policy to a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_user(user_name, policy_arn)
    @iam_client.attach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的AttachUser政策。

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。

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;
```

```
    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的 Rust API 參考AttachUser政策。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ChangePassword配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ChangePassword。

CLI

AWS CLI

變更 IAM 使用者的密碼

若要變更 IAM 使用者的密碼，建議您使用--cli-input-json參數傳遞包含新舊密碼的 JSON 檔案。使用此方法，您可以使用含非英數字元的強式密碼。當您將密碼作為命令行參數傳遞時，將密碼與非字母數字字元一起使用可能很困難。若要使用--cli-input-json參數，請先使用指change-password令搭配--generate-cli-skeleton參數，如下列範例所示。

```
aws iam change-password \
    --generate-cli-skeleton > change-password.json
```

上一個命令創建一個名為更改密碼 .json 的 JSON 文件，您可以使用該文件來填寫舊密碼和新密碼。例如，檔案可能如下所示。

```
{
    "OldPassword": "3s0K_;xh4~8XXI",
    "NewPassword": "]35d/{pB9Fo9wJ"
}
```

接下來，要更改您的密碼，請再次使用change-password命令，這次傳遞--cli-input-json參數來指定您的 JSON 文件。下列change-password命令使用名為變更密碼 .json 的 JSON 檔案的--cli-input-json參數。

```
aws iam change-password \
```

```
    --cli-input-json file://change-password.json
```

此命令不會產生輸出。

此命令只能由 IAM 使用者呼叫。如果使用 AWS 帳戶 (root) 認證呼叫此命令，則命令會傳回錯`InvalidUserType`誤。

如需詳細資訊，請參閱 IAM 使用者指南中的AWS IAM 使用者如何變更自己的密碼。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ChangePassword中的。

PowerShell

用於的工具 PowerShell

範例 1：此命令會變更執行命令之使用者的密碼。此命令只能由 IAM 使用者呼叫。如果在您使用 AWS 帳戶 (root) 認證登入時呼叫此命令，則命令會傳回錯誤**`InvalidUserType`**。

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ChangePassword式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**`CreateAccessKey`**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CreateAccessKey`。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 建立唯讀和讀寫的使用者
- 管理存取金鑰

.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。

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的CreateAccess金鑰。

Bash

AWS CLI 與 bash 腳本

> **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。

```
########################################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
########################################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
########################################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
```

```
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- 如需 API 詳細資訊，請參閱CreateAccess輸入AWS CLI 命令參考。

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::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                         const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome =
 iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey()
 <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的CreateAccess金鑰。

CLI

AWS CLI

為 IAM 使用者建立存取金鑰

下列 `create-access-key` 命令會為名為 Bob 的 IAM 使用者建立存取金鑰 (存取金鑰 ID 與私密存取金鑰)。

```
aws iam create-access-key \
    --user-name Bob
```

輸出：

```
{
    "AccessKey": {
        "UserName": "Bob",
        "Status": "Active",
        "CreateDate": "2015-03-09T18:39:23.411Z",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

請將私密存取金鑰存放在安全之處。遺失的金鑰無法復原，您必須建立新的存取金鑰。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

- 如需 API 詳細資訊，請參閱CreateAccess輸入AWS CLI 命令參考。

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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
 &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}
```

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

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.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

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

```
        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException 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CreateAccess金鑰。

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。

建立存取金鑰。

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- 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CreateAccess金鑰。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- 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CreateAccess金鑰。

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 createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的CreateAccess密鑰以獲取 Kotlin API 參考。

PowerShell

用於的工具 PowerShell

範例 1：此範例會建立新的存取金鑰和秘密存取 key pair，並將其指派給使用者**David**。請務必將**AccessKeyId**和**SecretAccessKey**值儲存至檔案，因為這是您唯一可以取得的時間**SecretAccessKey**。您稍後便無法擷取它。如果您遺失了密碼金鑰，您必須建立新的存取 key pair。

```
New-IAMAccessKey -UserName David
```

輸出：

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的CreateAccess金鑰。

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。

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- 如需 API 詳細資訊，請參閱AWS 開發套件中的CreateAccess金鑰 (Boto3) 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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、停用及刪除存取金鑰。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
```

```
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
```

```
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的CreateAccess金鑰。

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。

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
```

```
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的CreateAccess金鑰以取得 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
   public func createAccessKey(userName: String) async throws ->
IAMClientTypes.AccessKey {
       let input = CreateAccessKeyInput(
           userName: userName
       )
       do {
           let output = try await iamClient.createAccessKey(input: input)
           guard let accessKey = output.accessKey else {
               throw ServiceHandlerError.keyError
           }
           return accessKey
       } catch {
           throw error
       }
   }
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的CreateAccess密鑰以獲取 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateAccountAlias配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateAccountAlias。

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：

- 管理您的帳戶

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。

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }
```

```
    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的CreateAccount別名。

CLI

AWS CLI

建立帳戶別名

下列create-account-alias指令會examplecorp為您的 AWS 帳戶建立別名。

```
aws iam create-account-alias \
    --account-alias examplecorp
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的您的 AWS 帳戶 ID 及其別名。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的CreateAccount別名。

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.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to create (for example,
 myawsaccount).\s
                """;

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

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request =
 CreateAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);
```

```
        } catch (IamException 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CreateAccount別名。

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。

建立帳戶別名。

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- 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CreateAccount別名。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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CreateAccount別名。

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 createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的CreateAccount別名以獲取 Kotlin API 參考。

PowerShell

用於的工具 PowerShell

範例 1：此範例會將您帳戶的 AWS 帳戶別名變更為**mycompanyaws**。使用者登入頁面的位址會追蹤到 https://mycompanyaws.signin.aws.amazon.com/console。使用您的帳戶 ID 號碼而不是別名的原始網址（HTTPS://<accountidnumber>.signin.aws.amazon.com）仍會繼續運作。不過，任何先前定義的別名式 URL 都會停止運作。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的CreateAccount別名。

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。

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的CreateAccount別名，以供 Python (博多 3) API 參考使用。

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。

列出、建立及刪除帳戶別名。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
```

```
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的CreateAccount別名。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateGroup配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateGroup。

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：

- 建立群組並新增使用者。

.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。

```
    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
{ GroupName = groupName });
        return response.Group;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateGroup中的。

CLI

AWS CLI

建立 IAM 群組

下列 `create-group` 命令會建立名為 `Admins` 的 IAM 群組。

```
aws iam create-group \
    --group-name Admins
```

輸出：

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-03-09T20:30:24.940Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateGroup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。

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CreateGroup中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立名為的新 IAM 群組**Developers**。

```
New-IAMGroup -GroupName Developers
```

輸出：

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreateGroup式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CreateInstanceProfile**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateInstanceProfile。

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：

- 建置及管理彈性服務

.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。

```
   /// <summary>
   /// Create a policy, role, and profile that is associated with instances with
a specified name.
   /// An instance's associated profile defines a role that is assumed by the
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                        "\"Service\": [" +
                                            "\"ec2.amazonaws.com\"" +
```

```
                                    "]" +
                                    "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
```

```
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
```

```
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- 有關 API 詳細信息，請參閱 AWS SDK for .NET API 參考中的CreateInstance配置文件。

CLI

AWS CLI

建立執行個體設定檔

下列 `create-instance-profile` 命令會建立名為 `Webserver` 的執行個體設定檔。

```
aws iam create-instance-profile \
    --instance-profile-name Webserver
```

輸出：

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJMBYC7DLSPEXAMPLE",
        "Roles": [],
        "CreateDate": "2015-03-09T20:33:19.626Z",
        "InstanceProfileName": "Webserver",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"
    }
```

```
}
```

若要將角色新增至執行個體設定檔，請使用 `add-role-to-instance-profile` 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的CreateInstance設定檔。

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。

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- 有關 API 詳細信息，請參閱 AWS SDK for JavaScript API 參考中的CreateInstance配置文件。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立名為的新 IAM 執行個體設定檔**ProfileForDevEC2Instance**。您必須單獨執行命**Add-IAMRoleToInstanceProfile**令，將執行個體設定檔與提供執行個體許可的現有 IAM 角色建立關聯。最後，在啟動 EC2 執行個體時，將執行個體設定檔附加到該執行個

體。若要這麼做，請搭配**InstanceProfile_Arn**或**InstanceProfile_Name**參數使用**New-EC2Instance**指令程式。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

輸出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的CreateInstance設定檔。

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。

此範例會建立政策、角色和執行個體設定檔，並將它們全部連結在一起。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
```

```
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is
                          created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def create_instance_profile(
        self, policy_file, policy_name, role_name, profile_name,
aws_managed_policies=()
    ):
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
```

```
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        with open(policy_file) as file:
            instance_policy_doc = file.read()

        policy_arn = None
        try:
            pol_response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=instance_policy_doc
            )
            policy_arn = pol_response["Policy"]["Arn"]
            log.info("Created policy with ARN %s.", policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Policy %s already exists, nothing to do.", policy_name)
                list_pol_response = self.iam_client.list_policies(Scope="Local")
                for pol in list_pol_response["Policies"]:
                    if pol["PolicyName"] == policy_name:
                        policy_arn = pol["Arn"]
                        break
```

```
            if policy_arn is None:
                raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

        try:
            self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info("Created role %s and attached policy %s.", role_name,
policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Role %s already exists, nothing to do.", role_name)
            else:
                raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
```

```
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create profile {profile_name} and attach it to
role\n"
                    f"{role_name}: {err}"
                )
        return profile_arn
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的CreateInstance配置文件進行 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateLoginProfile配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateLoginProfile。

CLI

AWS CLI

建立 IAM 使用者的密碼

若要為 IAM 使用者建立密碼，建議您使用--cli-input-json參數傳遞包含密碼的 JSON 檔案。使用此方法，您可以建立含有非英數字元的強式密碼。當您將密碼作為命令行參數傳遞時，建立包含非英數字元的密碼可能很困難。

若要使用--cli-input-json參數，請先使用指create-login-profile令搭配--generate-cli-skeleton參數，如下列範例所示。

```
aws iam create-login-profile \
    --generate-cli-skeleton > create-login-profile.json
```

上一個命令會建立名為 create-login-profile .json 的 JSON 檔案，您可以使用該檔案填入後續create-login-profile指令的資訊。例如：

```
{
    "UserName": "Bob",
    "Password": "&1-3a6u:RA0djs",
    "PasswordResetRequired": true
}
```

接下來，要為 IAM 用戶創建密碼，請再次使用該`create-login-profile`命令，這次傳遞`--cli-input-json`參數來指定您的 JSON 文件。下列`create-login-profile`命令會使用名為 create-login-profile .json 的 JSON 檔案的`--cli-input-json`參數。

```
aws iam create-login-profile \
    --cli-input-json file://create-login-profile.json
```

輸出：

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2015-03-10T20:55:40.274Z",
        "PasswordResetRequired": true
    }
}
```

如果新密碼違反了帳號密碼策略，則命令會傳回`PasswordPolicyViolation`錯誤。

若要變更已有密碼的使用者密碼，請使用`update-login-profile`。若要設定帳號的密碼策略，請使用`update-account-password-policy`指令。

如果帳戶密碼政策允許他們，IAM 使用者可以使用`change-password`命令變更自己的密碼。

如需詳細資訊，請參閱 IAM 使用者指南中的管理AWS IAM 使用者的密碼。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的CreateLogin設定檔。

PowerShell

用於的工具 PowerShell

範例 1：此範例會為名為 Bob 的 IAM 使用者建立 (暫時) 密碼，並設定要求使用者在下次登入時**Bob**變更密碼的旗標。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

輸出：

```
CreateDate                  PasswordResetRequired             UserName
----------                  ---------------------             --------
4/14/2015 12:26:30 PM       True                              Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的CreateLogin設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`CreateOpenIdConnectProvider`配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要建立 OpenID Connect (OIDC) 提供者

若要建立 OpenID Connect (OIDC) 提供者，建議您使用`--cli-input-json`參數來傳遞包含必要參數的 JSON 檔案。當您建立 OIDC 提供者時，您必須傳遞提供者的 URL，且 URL 必須以開頭。`https://`將 URL 作為命令行參數傳遞可能很困難，因為冒號 (:) 和正斜線 (//) 字元在某些命令列環境中具有特殊意義。使用`--cli-input-json`參數可以繞過此限制。

若要使用`--cli-input-json`參數，請先使用指`create-open-id-connect-provider`令搭配`--generate-cli-skeleton`參數，如下列範例所示。

```
aws iam create-open-id-connect-provider \
    --generate-cli-skeleton > create-open-id-connect-provider.json
```

上一個命令會建立名為 create-open-id-connect-provider.json 的 JSON 檔案，您可以使用該檔案來填入後續指令的資訊。`create-open-id-connect-provider`例如：

```
{
```

```
    "Url": "https://server.example.com",
    "ClientIDList": [
        "example-application-ID"
    ],
    "ThumbprintList": [
        "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"
    ]
}
```

接下來，要創建 OpenID Connect（OIDC）提供程序，請再次使用該`create-open-id-connect-provider`命令，這次傳遞`--cli-input-json`參數來指定您的 JSON 文件。以下`create-open-id-connect-provider`命令使用名為 create-open-id-connect-provider.json 的 JSON 文件的`--cli-input-json`參數。

```
aws iam create-open-id-connect-provider \
    --cli-input-json file://create-open-id-connect-provider.json
```

輸出：

```
{
    "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
}
```

如需有關 OIDC 提供者的詳細資訊，請參閱 IAM 使用者指AWS 南中的建立 OpenID Connect (OIDC) 身分識別提供者。

如需取得 OIDC 提供者指紋的詳細資訊，請參閱 IAM 使用者指南中的為 OpenID Connect 身分提供者取得指AWS 紋。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的CreateOpenIdConnect提供者。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立與位於 URL **`https://example.oidcprovider.com`** 和用戶端 ID 的 OIDC 相容提供者服務相關聯的 IAM OIDC 提供者。**`my-testapp-1`**OIDC 提供者會提供指紋。要驗證指紋，請按照以下步驟操作：http://docs.aws.amazon.com/IAM/latest/ UserGuide /身份提供者-OIDC-獲取的指紋。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

輸出：

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的CreateOpenIdConnect提供者。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`CreatePolicy`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`CreatePolicy`。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 建立唯讀和讀寫的使用者
- 管理政策
- 使用 IAM 政策產生器 API

.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。

```
    /// <summary>
    /// Create an IAM policy.
```

```
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreatePolicy中的。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
```

```
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
```

```
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreatePolicy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。

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
```

```
                                    const Aws::String &rsrcArn,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                  std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
```

```
                << "       ],"
                << "       \"Resource\": \""
                << rsrc_arn
                << "\""
                << "    }"
                << "   ]"
                << "}";

    return stringStream.str();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考CreatePolicy中的。

CLI

AWS CLI

範例 1：建立客戶管理政策

下列命令會建立名為 `my-policy` 的客戶管理政策。

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy
```

檔案 `policy` 是目前資料夾中的 JSON 文件，在名為 `my-bucket` 的 Amazon S3 儲存貯體中授予 `shared` 資料夾的唯讀存取權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/shared/*"
            ]
        }
```

```
    ]
}
```

輸出：

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "CreateDate": "2015-06-01T19:31:18.620Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
        "UpdateDate": "2015-06-01T19:31:18.620Z"
    }
}
```

如需有關使用檔案作為字串參數輸入的詳細資訊，請參閱 AWS CLI 使用者指南中的指定AWS CLI 的參數值。

範例 2：建立內含描述的客戶管理政策

下列命令會建立名為 my-policy 的客戶管理政策，其中包含不可變的描述：

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --description "This policy grants access to all Put, Get, and List actions
 for my-bucket"
```

檔案 policy.json 是目前資料夾中的 JSON 文件，可針對名為 my-bucket 的 Amazon S3 儲存貯體，授予所有 Put、List 和 Get 動作的存取權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        }
    ]
}
```

輸出：

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T22:38:47+00:00",
        "UpdateDate": "2023-05-24T22:38:47+00:00"
    }
}
```

如需有關以身分為基礎之政策的詳細資訊，請參閱《AWS IAM 使用者指南》中的以身分為基礎和以資源為基礎的政策。

範例 3：建立內含標籤的客戶管理政策

下列命令會建立名為 my-policy 的客戶管理政策，其中包含標籤。此範例使用具有下列 JSON 格式標記的 --tags 參數旗標：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 旗標可以與速記格式的標籤一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
```

```
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

檔案 policy.json 是目前資料夾中的 JSON 文件，可針對名為 my-bucket 的 Amazon S3 儲存貯體，授予所有 Put、List 和 Get 動作的存取權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                 "s3:ListBucket*",
                 "s3:PutBucket*",
                 "s3:GetBucket*"
             ],
             "Resource": [
                 "arn:aws:s3:::my-bucket"
             ]
         }
     ]
 }
```

輸出：

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::12345678012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T23:16:39+00:00",
        "UpdateDate": "2023-05-24T23:16:39+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
                "Key": "Location",
```

```
                "Value": "Seattle"
            {

        ]
    }
}
```

如需有關標記政策的詳細資訊，請參閱《AWS IAM 使用者指南》中的標記客戶管理政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreatePolicy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。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
```

```
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
 &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CreatePolicy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.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
```

```
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * 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 CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;
```

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

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: "
+ result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException 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CreatePolicy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。

建立政策。

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyName
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });
```

```
  return client.send(command);
};
```

- 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CreatePolicy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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
```

```
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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CreatePolicy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 createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
```

```
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "        ]," +
            "        \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreatePolicy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。

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
```

```
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    public function createPolicy(string $policyName, string $policyDocument)
    {
        $result = $this->customWaiter(function () use ($policyName,
 $policyDocument) {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreatePolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會在名為「檔案」的目前 AWS 帳戶中建立新MySamplePolicy的 IAM 政策，MySamplePolicy.json提供政策內容。請注意，您必須使用 -Raw switch 參數才能成功處理 JSON 原則檔案。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

輸出：

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
```

```
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreatePolicy式參考中的。

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。

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::my-bucket/*' to allow actions on all
 objects
                         in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
```

```
            logger.info("Created policy %s.", policy.arn)
        except ClientError:
            logger.exception("Couldn't create policy %s.", name)
            raise
        else:
            return policy
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreatePolicy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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、附加和解除連結角色原則。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
```

```
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end
```

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreatePolicy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。

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CreatePolicy中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func createPolicy(name: String, policyDocument: String) async throws -
> IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
```

```
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreatePolicy中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreatePolicyVersion配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreatePolicyVersion。

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：

- 管理政策

CLI

AWS CLI

建立新版本的受管政策

此範例會建立新 v2 版的 IAM 政策 (其 ARN 為 arn:aws:iam::123456789012:policy/MyPolicy)，並將該版本設為預設版本。

```
aws iam create-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --policy-document file://NewPolicyVersion.json \
    --set-as-default
```

輸出：

```
{
```

```
    "PolicyVersion": {
        "CreateDate": "2015-06-16T18:56:03.721Z",
        "VersionId": "v2",
        "IsDefaultVersion": true
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 政策的版本控制。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的CreatePolicy版本。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立新的「v2」版本的 IAM 政策，其 ARN 為，**arn:aws:iam::123456789012:policy/MyPolicy**並使其成為預設版本。**NewPolicyVersion.json**檔案會提供原則內容。請注意，您必須使用 **-Raw** switch 參數才能成功處理 JSON 原則檔案。

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

輸出：

```
CreateDate                          Document                IsDefaultVersion
          VersionId
----------                          --------                ----------------
          ---------
4/15/2015 10:54:54 AM                                       True
          v2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的CreatePolicy版本。

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。

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
```

```
        return policy_version
```

- 如需 API 的詳細資訊，請參閱適用於 Python (Boto3) API 參考的AWS SDK CreatePolicy 版本。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateRole配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateRole。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 管理角色

.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。

```
/// <summary>
/// Create a new IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role.</param>
/// <param name="rolePolicyDocument">The name of the IAM policy document
/// for the new role.</param>
/// <returns>The Amazon Resource Name (ARN) of the role.</returns>
```

```
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateRole中的。

Bash

AWS CLI 與 bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
```

```
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
```

```
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateRole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。

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
```

```
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考CreateRole中的。

CLI

AWS CLI

範例 1：建立 IAM 角色

下列 `create-role` 命令會建立名為 `Test-Role` 的角色，並將信任政策連接至該角色。

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

輸出：

```
{
    "Role": {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
```

```
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "CreateDate": "2013-06-07T20:43:32.821Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

在 Test-Role-Trust-Policy.json 檔案中，將信任政策定義為 JSON 文件。(檔案名稱和副檔名沒有意義。) 信任政策必須指定主體。

若要將許可政策連接至角色，請使用 `put-role-policy` 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 角色。

範例 2：建立具有指定最長工作階段持續時間的 IAM 角色

下列 `create-role` 命令會建立名為 `Test-Role` 的角色，並設定 7200 秒 (2 小時) 的最長工作階段持續時間。

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --max-session-duration 7200
```

輸出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:role/Test-Role",
        "CreateDate": "2023-05-24T23:50:25+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::12345678012:root"
                    },
```

```
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色最長工作階段持續時間 (AWS API)。

範例 3：建立內含標籤的 IAM 角色

下列命令會建立內含標籤的 IAM 角色 Test-Role。此範例使用具有下列 JSON 格式標記的 --tags 參數旗標：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 旗標可以與速記格式的標籤一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

輸出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role",
        "CreateDate": "2023-05-25T23:29:41+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole"
```

```
                }
            ]
        },
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的標記 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateRole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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
```

```
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
 (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action: []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(context.TODO(),
 &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CreateRole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 org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-
roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
```

```
                    fileLocation - The location of the JSON document that
represents the trust policy.\s
                """;

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

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject)
readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());

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

    public static Object readJsonSimpleDemo(String filename) throws Exception {
```

```
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考CreateRole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。

建立角色。

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
```

```
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考CreateRole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。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
```

```
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
            ]);
        });
        return $result['Role'];
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreateRole中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立名為的新角色，MyNewRole並將檔案中找到的原則附加至該角色NewRoleTrustPolicy.json。請注意，您必須使用 -Raw switch 參數才能成功處理 JSON 原則檔案。輸出中顯示的策略文件經過 URL 編碼。它在這個例子中使用 UrlDecode .NET 方法進行解碼。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

輸出：

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
```

```
                              %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                              %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreateRole式參考中的。

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。

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
```

```
    :return: The newly created role.
    """
    trust_policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateRole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 設定和執行程式碼範例儲存庫。

```
# Creates a role and attaches policies to it.
#
```

```
  # @param role_name [String] The name of the role.
  # @param assume_role_policy_document [Hash] The trust relationship policy
document.
  # @param policy_arns [Array<String>] The ARNs of the policies to attach.
  # @return [String, nil] The ARN of the new role if successful, or nil if an
error occurred.
  def create_role(role_name, assume_role_policy_document, policy_arns)
    response = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: assume_role_policy_document.to_json
    )
    role_arn = response.role.arn

    policy_arns.each do |policy_arn|
      @iam_client.attach_role_policy(
        role_name: role_name,
        policy_arn: policy_arn
      )
    end

    role_arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating role: #{e.message}")
    nil
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreateRole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。

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
```

```
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CreateRole中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
   public func createRole(name: String, policyDocument: String) async throws ->
String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
```

```
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
                throw ServiceHandlerError.noSuchRole
            }
            return id
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateRole中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CreateSAMLProvider**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateSAMLProvider。

CLI

AWS CLI

建立 SAML 提供者

此範例會在 IAM 中建立名為 MySAMLProvider 的新 SAML 提供者。它會由在檔案 SAMLMetaData.xml 中找到的 SAML 中繼資料文件進行描述。

```
aws iam create-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --name MySAMLProvider
```

輸出：

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM SAML 身分提供者。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 CreateSAMLProvider。

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。

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "fs";
import * as path from "path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
```

```
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的 CreateSAMLProvider。

PowerShell

適用的工具 PowerShell

範例 1：此範例會在 IAM 中建立新的 SAML 提供者實體。檔案中的 SAML 中繼資料文件會命名**MySAMLProvider**並加以描述**SAMLMetaData.xml**，該文件是從 SAML 服務提供者的網站分開下載的。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

輸出：

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- 如需 API 詳細資訊，請參閱在指令程式參考中建立 SAML 提供者。AWS Tools for PowerShell

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**CreateServiceLinkedRole**配 AWS 開發套件或 CLI 使用

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

.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。

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateServiceLinkedRole中的。

CLI

AWS CLI

建立服務連結角色

下列create-service-linked-role範例會為指定的服務建立 AWS 服務連結角色，並附加指定的描述。

```
aws iam create-service-linked-role \
    --aws-service-name lex.amazonaws.com \
    --description "My service-linked role to support Lex"
```

輸出：

```
{
    "Role": {
        "Path": "/aws-service-role/lex.amazonaws.com/",
        "RoleName": "AWSServiceRoleForLexBots",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
        "CreateDate": "2019-04-17T20:34:14+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "lex.amazonaws.com"
                        ]
                    }
                }
            ]
        }
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用服務連結角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateServiceLinkedRole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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
 description string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

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

建立服務連結角色。

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-
services-that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/
general/latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
```

```
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in
 this account",
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    } else {
      throw caught;
    }
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考[CreateServiceLinkedRole](#)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](#)。

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' =>
 $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
```

```
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreateServiceLinkedRole中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立自動調度資源服務的服務連結角色。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreateServiceLinkedRole式參考中的。

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。

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
```

```
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
service_name)
        raise
    else:
        return role
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateServiceLinkedRole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 設定和執行程式碼範例儲存庫。

```
 # Creates a service-linked role
 #
 # @param service_name [String] The service name to create the role for.
 # @param description [String] The description of the service-linked role.
 # @param suffix [String] Suffix for customizing role name.
 # @return [String] The name of the created role
 def create_service_linked_role(service_name, description, suffix)
   response = @iam_client.create_service_linked_role(
     aws_service_name: service_name, description: description, custom_suffix:
suffix,)
   role_name = response.role.role_name
   @logger.info("Created service-linked role #{role_name}.")
   role_name
```

```
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't create service-linked role for #{service_name}.
 Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreateServiceLinkedRole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。

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput,
 SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CreateServiceLinkedRole中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func createServiceLinkedRole(service: String, suffix: String? = nil,
description: String?)
                    async throws -> IAMClientTypes.Role {
        let input = CreateServiceLinkedRoleInput(
            awsServiceName: service,
            customSuffix: suffix,
            description: description
        )
        do {
            let output = try await client.createServiceLinkedRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateServiceLinkedRole中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateUser配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateUser。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 建立唯讀和讀寫的使用者

.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。

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateUser中的。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
```

```
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""
```

```
  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateUser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。

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateUserRequest create_request;
    create_request.SetUserName(userName);
```

```
    auto create_outcome = iam.CreateUser(create_request);
    if (!create_outcome.IsSuccess()) {
        std::cerr << "Error creating IAM user " << userName << ":" <<
                  create_outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }

    return create_outcome.IsSuccess();
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考CreateUser中的。

CLI

AWS CLI

範例 1：建立 IAM 使用者

下列 `create-user` 命令會建立目前帳戶中名為 Bob 的 IAM 使用者。

```
aws iam create-user \
    --user-name Bob
```

輸出：

```
{
    "User": {
        "UserName": "Bob",
        "Path": "/",
        "CreateDate": "2023-06-08T03:20:41.270Z",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Bob"
    }
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的在 AWS 帳戶中建立AWS IAM 使用者。

範例 2：在指定路徑建立 IAM 使用者

下列 `create-user` 命令會在指定路徑建立名為 Bob 的 IAM 使用者。

```
aws iam create-user \
    --user-name Bob \
    --path /division_abc/subdivision_xyz/
```

輸出：

```
{
    "User": {
        "Path": "/division_abc/subdivision_xyz/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",
        "CreateDate": "2023-05-24T18:20:17+00:00"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 識別符。

範例 3：建立內含標籤的 IAM 使用者

下列 create-user 命令會建立內含標籤、名為 Bob 的 IAM 使用者。此範例使用具有下列 JSON 格式標記的 --tags 參數旗標：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 旗標可以與速記格式的標籤一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-user \
    --user-name Bob \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

輸出：

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-25T17:14:21+00:00",
        "Tags": [
```

```
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的標記 IAM 使用者。

範例 3：建立具有設定許可界限的 IAM 使用者

以下create-user命令會建立一個以 Amazon FullAccess S3 許可界限命名Bob的 IAM 使用者。

```
aws iam create-user \
    --user-name Bob \
    --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

輸出：

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-24T17:50:53+00:00",
        "PermissionsBoundary": {
        "PermissionsBoundaryType": "Policy",
        "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
        }
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 實體許可界限。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateUser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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(context.TODO(),
 &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考CreateUser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.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * 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 CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

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

```
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);

 waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException 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CreateUser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。

建立使用者。

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- 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CreateUser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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
  }
});
```

- 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CreateUser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 createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateUser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。

```
$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
```

```
        ]);

        return $result['User'];
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考CreateUser中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立名為的 IAM 使用者**Bob**。如果 Bob 需要登入 AWS 主控台，則您必須個別執行命令，**New-IAMLoginProfile**以使用密碼建立登入設定檔。如果 Bob 需要執行 AWS PowerShell 或跨平台 CLI 命令或進行 AWS API 呼叫，您必須分別執行命**New-IAMAccessKey**令來建立存取金鑰。

```
New-IAMUser -UserName Bob
```

輸出：

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
UserName         : Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreateUser式參考中的。

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。

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateUser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 設定和執行程式碼範例儲存庫。

```
 # Creates a user and their login profile
 #
 # @param user_name [String] The name of the user
 # @param initial_password [String] The initial password for the user
 # @return [String, nil] The ID of the user if created, or nil if an error
occurred
 def create_user(user_name, initial_password)
   response = @iam_client.create_user(user_name: user_name)
   @iam_client.wait_until(:user_exists, user_name: user_name)
   @iam_client.create_login_profile(
```

```
      user_name: user_name,
      password: initial_password,
      password_reset_required: true
    )
    @logger.info("User '#{user_name}' created successfully.")
    response.user.user_id
  rescue Aws::IAM::Errors::EntityAlreadyExists
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考CreateUser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。

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK CreateUser中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func createUser(name: String) async throws -> String {
    let input = CreateUserInput(
        userName: name
    )
    do {
        let output = try await client.createUser(input: input)
        guard let user = output.user else {
            throw ServiceHandlerError.noSuchUser
        }
        guard let id = user.userId else {
            throw ServiceHandlerError.noSuchUser
        }
        return id
    } catch {
        throw error
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateUser中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateVirtualMfaDevice配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateVirtualMfaDevice。

CLI

AWS CLI

若要建立虛擬 MFA 裝置

此範例會建立名BobsMFADevice為的新虛擬 MFA 裝置。它創建一個包含稱為引導信息的文件，QRCode.png並將其放置在C:/目錄中。在這個例子中使用的引導方法是QRCodePNG。

```
aws iam create-virtual-mfa-device \
    --virtual-mfa-device-name BobsMFADevice \
    --outfile C:/QRCode.png \
    --bootstrap-method QRCodePNG
```

輸出：

```
{
    "VirtualMFADevice": {
        "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"
}
```

如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考CreateVirtualMfaDevice中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立新的虛擬 MFA 裝置。第 2 行和第 3 行提取虛擬 MFA 軟件程序創建帳戶所需的**Base32StringSeed**值（作為 QR 碼的替代方法）。使用值配置程序後，從程序中獲取兩個連續的身份驗證代碼。最後，使用最後一個命令將虛擬 MFA 裝置連結至 IAM 使用者，**Bob**並使用這兩個驗證碼同步處理帳戶。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
```

```
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

輸出：

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

範例 2：此範例會建立新的虛擬 MFA 裝置。第 2 行和第 3 行會擷取**QRCodePNG**值並將其寫入檔案。虛擬 MFA 軟體程式可掃描此影像，以建立帳戶 (作為手動輸入 Base32 StringSeed 值的替代方法)。在虛擬 MFA 程式中建立帳戶後，取得兩個循序驗證代碼，並在最後一個命令中輸入這些代碼，以將虛擬 MFA 裝置連結至 IAM 使用者**Bob**並同步處理帳戶。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

輸出：

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程CreateVirtualMfaDevice式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeactivateMfaDevice**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeactivateMfaDevice。

CLI

AWS CLI

若要停用 MFA 裝置

此指令會停用與使用者相關聯之 ARN arn:aws:iam::210987654321:mfa/BobsMFADevice 的虛擬 MFA 裝置。Bob

```
aws iam deactivate-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

此命令不會產生輸出。

如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeactivateMfa裝置。

PowerShell

適用的工具 PowerShell

範例 1：此指令會停用與具有序號**123456789012**之使用者**Bob**相關聯的硬體 MFA 裝置。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

範例 2：此指令會停用與具有 AR **arn:aws:iam::210987654321:mfa/David** N 之使用者**David**相關聯的虛擬 MFA 裝置。請注意，虛擬 MFA 裝置不會從帳戶中刪除。虛擬裝置仍然存在，並出現在**Get-IAMVirtualMFADevice**指令的輸出中。您必須先使用**Remove-IAMVirtualMFADevice**指令刪除舊的虛擬 MFA 裝置，才能為相同使用者建立新的虛擬 MFA 裝置。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeactivateMfa裝置。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeleteAccessKey**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteAccessKey。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 建立唯讀和讀寫的使用者
- 管理存取金鑰

.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。

```
    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
```

```
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的DeleteAccess金鑰。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_access_key() {
```

```
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an WS Identity and Access Management (IAM) access key for the
specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
```

```
  iecho "    Username:   $user_name"
  iecho "    Access key:   $access_key"
  iecho ""

  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- 如需 API 詳細資訊，請參閱DeleteAccess輸入AWS CLI 命令參考。

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。

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
```

```
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的DeleteAccess金鑰。

CLI

AWS CLI

刪除 IAM 使用者的存取金鑰

下列 `delete-access-key` 命令會為名為 Bob 的 IAM 使用者刪除指定的存取金鑰 (存取金鑰 ID 與私密存取金鑰)。

```
aws iam delete-access-key \
    --access-key-id AKIDPMS9RO4H3FEXAMPLE \
    --user-name Bob
```

此命令不會產生輸出。

若要列出為 IAM 使用者定義的存取金鑰，請使用 `list-access-keys` 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

- 如需 API 詳細資訊，請參閱DeleteAccess輸入AWS CLI 命令參考。

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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
 &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}
```

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

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.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
                    accessKey - The access key ID for the secret access key you
 want to delete.\s
                """;

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

```
        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String
 accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException 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DeleteAccess金鑰。

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。

刪除存取金鑰。

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- 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DeleteAccess金鑰。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
```

```
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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DeleteAccess金鑰。

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 deleteKey(
    userNameVal: String,
    accessKey: String
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的DeleteAccess密鑰以獲取 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：此範例會**AKIAIOSFODNN7EXAMPLE**從名為的使用者刪除 AWS 存取 key pair 及金鑰 ID **Bob**。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteAccess金鑰。

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。

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- 如需 API 詳細資訊，請參閱AWS 開發套件中的DeleteAccess金鑰 (Boto3) 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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、停用及刪除存取金鑰。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end
```

```
 # Creates an access key for a user
 #
 # @param user_name [String] The name of the user.
 # @return [Boolean]
 def create_access_key(user_name)
   response = @iam_client.create_access_key(user_name: user_name)
   access_key = response.access_key
   @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
   access_key
 rescue Aws::IAM::Errors::LimitExceeded => e
   @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
   nil
 rescue StandardError => e
   @logger.error("Error creating access key: #{e.message}")
   nil
 end

 # Deactivates an access key
 #
 # @param user_name [String] The name of the user.
 # @param access_key_id [String] The ID for the access key.
 # @return [Boolean]
 def deactivate_access_key(user_name, access_key_id)
   @iam_client.update_access_key(
     user_name: user_name,
     access_key_id: access_key_id,
     status: "Inactive"
   )
   true
 rescue StandardError => e
   @logger.error("Error deactivating access key: #{e.message}")
   false
 end

 # Deletes an access key
 #
 # @param user_name [String] The name of the user.
 # @param access_key_id [String] The ID for the access key.
 # @return [Boolean]
 def delete_access_key(user_name, access_key_id)
   @iam_client.delete_access_key(
```

```
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的DeleteAccess金鑰。

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。

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
```

```
            }
        }
    }
    Ok(())
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的DeleteAccess金鑰以取得 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                                key: IAMClientTypes.AccessKey) async throws {
        let userName: String?

        if user != nil {
            userName = user!.userName
        } else {
            userName = nil
        }

        let input = DeleteAccessKeyInput(
            accessKeyId: key.accessKeyId,
            userName: userName
        )
        do {
            _ = try await iamClient.deleteAccessKey(input: input)
        } catch {
```

```
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的DeleteAccess密鑰以獲取 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteAccountAlias配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteAccountAlias。

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：

- 管理您的帳戶

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。

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
```

```
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的DeleteAccount別名。

CLI

AWS CLI

刪除帳戶別名

下列 delete-account-alias 命令會移除目前帳戶的別名 mycompany。

```
aws iam delete-account-alias \
    --account-alias mycompany
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的您的 AWS 帳戶 ID 及其別名。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteAccount別名。

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.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <alias>\s

                Where:
                    alias - The account alias to delete.\s
                """;

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

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request =
 DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();
```

```
            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考中的DeleteAccount別名。

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。

刪除帳戶別名。

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- 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DeleteAccount別名。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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DeleteAccount別名。

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 deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的DeleteAccount別名以獲取 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：此範例會從您的 AWS 帳戶. 用戶登錄頁面與別名 https://mycompanyaws.signin.aws.amazon.com/console 不再有效。您必須改為使用原始網址與您的 AWS 帳戶 ID 號碼在 HTTPS://. <accountidnumber>

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteAccount別名。

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。

```
def delete_alias(alias):
```

```
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的DeleteAccount別名，以供 Python (博多 3) API 參考使用。

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。

列出、建立及刪除帳戶別名。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
```

```
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的DeleteAccount別名。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteAccountPasswordPolicy配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要刪除目前的帳號密碼策略

下列delete-account-password-policy命令會移除目前帳戶的密碼策略。

```
aws iam delete-account-password-policy
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的設定 IAM 使用者的帳戶密碼政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteAccountPasswordPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除的密碼原則， AWS 帳戶 並將所有值重設為其原始預設值。如果密碼策略目前不存在，將顯示以下錯誤訊息：找 PasswordPolicy 不到具有名稱的帳號策略。

```
Remove-IAMAccountPasswordPolicy
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteAccountPasswordPolicy式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteGroup配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteGroup。

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：

- 建立群組並新增使用者。

.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。

```
    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteGroup中的。

CLI

AWS CLI

刪除 IAM 群組

下列 `delete-group` 命令會刪除名為 `MyTestGroup` 的 IAM 群組。

```
aws iam delete-group \
    --group-name MyTestGroup
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的刪除 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteGroup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。

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteGroup中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除名為的 IAM 群組**MyTestGroup**。第一個命令會移除群組成員的所有 IAM 使用者，第二個命令會刪除 IAM 群組。這兩個命令都可以在沒有任何確認提示的情

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteGroup式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteGroupPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteGroupPolicy。

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：

- 建立群組並新增使用者。

.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。

```
    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
```

```
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的DeleteGroup政策。

CLI

AWS CLI

從 IAM 群組中刪除政策

下列 `delete-group-policy` 命令會將名為 `ExamplePolicy` 的政策從名為 `Admins` 的群組中刪除。

```
aws iam delete-group-policy \
    --group-name Admins \
    --policy-name ExamplePolicy
```

此命令不會產生輸出。

若要查看連接至群組的政策，請使用 `list-group-policies` 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteGroup政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會移除 IAM 群組**TesterPolicy**中名為的內嵌政策**Testers**。該群組中的使用者會立即失去該原則中定義的權限。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteGroup原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteInstanceProfile配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteInstanceProfile。

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：

- 建置及管理彈性服務

.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。

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
```

```
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }
```

- 有關 API 詳細信息，請參閱 AWS SDK for .NET API 參考中的DeleteInstance配置文件。

CLI

AWS CLI

刪除執行個體設定檔

下列 delete-instance-profile 命令會刪除名為 ExampleInstanceProfile 的執行個體設定檔。

```
aws iam delete-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用執行個體設定檔。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteInstance設定檔。

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。

```
const client = new IAMClient({});
await client.send(
  new DeleteInstanceProfileCommand({
    InstanceProfileName: NAMES.instanceProfileName,
  }),
);
```

- 有關 API 詳細信息，請參閱 AWS SDK for JavaScript API 參考中的DeleteInstance配置文件。

PowerShell

用於的工具 PowerShell

範例 1：此範例會刪除名為的 EC2 執行個體設定檔**MyAppInstanceProfile**。第一個指令會從執行個體設定檔中分離任何角色，然後第二個指令會刪除執行個體設定檔。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteInstance設定檔。

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。

此範例會從執行個體設定檔中移除角色、分離所有附加到該角色的政策並刪除全部資源。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is
                          created.
```

```
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def delete_instance_profile(self, profile_name, role_name):
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
```

```
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete instance profile {profile_name} or detach "
                    f"policies and delete role {role_name}: {err}"
                )
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的DeleteInstance配置文件進行 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteLoginProfile配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteLoginProfile。

CLI

AWS CLI

刪除 IAM 使用者的密碼

下列delete-login-profile命令會刪除名為的 IAM 使用者的密碼Bob。

```
aws iam delete-login-profile \
    --user-name Bob
```

此命令不會產生輸出。

如需詳細資訊，請參閱 IAM 使用者指南中的管理AWS IAM 使用者的密碼。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteLogin設定檔。

PowerShell

適用的工具 PowerShell

範例 1：此範例會從名為的 IAM 使用者刪除登入設定檔**Bob**。這樣可以防止用戶登錄到控制台。 AWS 它不會阻止使用者使用可能仍附加至使用者帳戶的 AWS 存取金鑰執行任何 AWS CLI PowerShell、或 API 呼叫。

```
Remove-IAMLoginProfile -UserName Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteLogin設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeleteOpenIdConnectProvider**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要刪除身分識別提供者

此範例會刪除連線至提供者的 IAM OIDC 提供者。example.oidcprovider.com

```
aws iam delete-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的建立 OpenID Connect (OIDC) 身分識別提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteOpenIdConnect提供者。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除連線至提供者的 IAM OIDC 提供者。**example.oidcprovider.com**請務必更新或刪除角色信任原則**Principal**元素中參照此提供者的任何角色。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteOpenIdConnect提供者。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeletePolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeletePolicy。

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：

- 建立使用者並擔任角色
- 建立唯讀和讀寫的使用者
- 管理政策

.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。

```
    /// <summary>
    /// Delete an IAM policy.
```

```
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
 DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeletePolicy中的。

Bash

AWS CLI 與 bash 腳本

> **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。

```
#############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

#############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
```

```
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeletePolicy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。

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeletePolicy中的。

CLI

AWS CLI

刪除 IAM 政策

此範例會刪除 ARN 為 `arn:aws:iam::123456789012:policy/MySamplePolicy` 的政策。

```
aws iam delete-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeletePolicy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。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
 _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeletePolicy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.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

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

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeletePolicy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。

刪除策略。

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeletePolicy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 deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除 ARN 為**arn:aws:iam::123456789012:policy/MySamplePolicy**的原則。在刪除原則之前，您必須先刪除除預設值以外的所有版本**Remove-IAMPolicyVersion**。您也必須將政策從任何 IAM 使用者、群組或角色中斷連結。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

範例 2：此範例會刪除政策，方法是先刪除所有非預設政策版本，從所有附加的 IAM 實體中斷連結，最後刪除政策本身。第一行會擷取政策物件。第二行會擷取未標記為預設版本的所有原則版本到集合中，然後刪除集合中的每個原則。第三行會擷取政策所附加的所有 IAM 使用者、群組和角色。第四到第六行會從每個附加的實體中斷原則。最後一行使用此命令來移除受管理的策略以及剩餘的預設版本。此範例包括任何需要它來抑制確認提示的行上的 **-Force** switch 參數。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeletePolicy式參考中的。

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。

```
def delete_policy(policy_arn):
```

```
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

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

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。

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeletePolicy中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
    let input = DeletePolicyInput(
        policyArn: policy.arn
    )
    do {
        _ = try await iamClient.deletePolicy(input: input)
    } catch {
        throw error
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeletePolicy中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DeletePolicyVersion`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`DeletePolicyVersion`。

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：

- 管理政策

- 復原政策版本

CLI

AWS CLI

刪除受管策略的版本

此範例會v2從 ARN 所屬原則刪除識別為的版本。arn:aws:iam::123456789012:policy/MySamplePolicy

```
aws iam delete-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeletePolicy版本。

PowerShell

適用的工具 PowerShell

範例 1：此範例會**v2**從 ARN 所屬原則刪除識別為的版本。**arn:aws:iam::123456789012:policy/MySamplePolicy**

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

範例 2：此範例會刪除原則，方法是先刪除所有非預設原則版本，然後刪除原則本身。第一行會擷取政策物件。第二行會擷取未標記為預設值的所有原則版本至集合中，然後使用此命令刪除集合中的每個原則。最後一行會移除原則本身以及剩餘的預設版本。請注意，若要成功刪除受管理的策略，您還必須使用、和**Unregister-IAMRolePolicy**命令將策略從任何使用者**Unregister-IAMUserPolicyUnregister-IAMGroupPolicy**、群組或角色中斷連結。請參閱**Remove-IAMPolicy**指令程式的範例。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
```

```
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeletePolicy版本。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteRole配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteRole。

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：

- 建立使用者並擔任角色
- 管理角色

.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。

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteRole中的。

Bash

AWS CLI 與 bash 腳本

> **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。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
```

```
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an WS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
```

```
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteRole中的。

CLI

AWS CLI

刪除 IAM 角色

下列 delete-role 命令會將名為 Test-Role 的角色移除。

```
aws iam delete-role \
    --role-name Test-Role
```

此命令不會產生輸出。

刪除角色之前，您必須先從任何執行個體設定檔 (remove-role-from-instance-profile) 移除該角色、分離任何受管政策 (detach-role-policy)，並刪除任何連接至該角色的內嵌政策 (delete-role-policy)。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 角色和使用執行個體設定檔。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteRole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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

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

刪除角色。

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteRole中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例會刪除目前 IAM 帳戶**MyNewRole**中指定的角色。您必須先使用**Unregister-IAMRolePolicy**命令中斷連結任何受管理的策略，才能刪除角色。內嵌原則會與角色一起刪除。

```
Remove-IAMRole -RoleName MyNewRole
```

範例 2：此範例會將任何受管理的策略從名為的角色中斷連結，**MyNewRole**然後刪除該角色。第一行會擷取任何作為集合附加至該角色的受管理原則，然後將集合中的每個原則與角色分離。第二行刪除角色本身。內嵌原則會與角色一起刪除。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteRole式參考中的。

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。

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteRole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 設定和執行程式碼範例儲存庫。

```
  # Deletes a role and its attached policies.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    begin
      # Detach and delete attached policies
      @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
        response.attached_policies.each do |policy|
          @iam_client.detach_role_policy({
                                       role_name: role_name,
                                       policy_arn: policy.policy_arn
                                     })
          # Check if the policy is a customer managed policy (not AWS managed)
          unless policy.policy_arn.include?("aws:policy/")
            @iam_client.delete_policy({ policy_arn: policy.policy_arn })
            @logger.info("Deleted customer managed policy
 #{policy.policy_name}.")
          end
        end
      end

      # Delete the role
      @iam_client.delete_role({ role_name: role_name })
      @logger.info("Deleted role #{role_name}.")
    rescue Aws::IAM::Errors::ServiceError => e
      @logger.error("Couldn't detach policies and delete role #{role_name}.
 Here's why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteRole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。

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError>
 {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeleteRole中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func deleteRole(role: IAMClientTypes.Role) async throws {
        let input = DeleteRoleInput(
            roleName: role.roleName
        )
        do {
            _ = try await iamClient.deleteRole(input: input)
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteRole中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DeleteRolePermissionsBoundary`配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

從 IAM 角色刪除許可界限

下列`delete-role-permissions-boundary`範例會刪除指定 IAM 角色的許可界限。若要將權限界限套用至角色，請使用`put-role-permissions-boundary`指令。

```
aws iam delete-role-permissions-boundary \
    --role-name lambda-application-role
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteRolePermissionsBoundary中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例顯示如何移除附加至 IAM 角色的權限界限。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteRolePermissionsBoundary式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteRolePolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteRolePolicy。

.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。

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
```

```
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的DeleteRole政策。

CLI

AWS CLI

從 IAM 角色中移除政策

下列 `delete-role-policy` 命令會將名為 `ExamplePolicy` 的政策從名為 `Test-Role` 的角色中移除。

```
aws iam delete-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteRole政策。

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。

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的DeleteRole政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除 IAM 角色中內嵌的內嵌政策**S3AccessPolicyS3BackupRole**。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteRole原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteSAMLProvider配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteSAMLProvider。

CLI

AWS CLI

刪除 SAML 提供者

此範例會刪除 ARN 為 arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider 的 IAM SAML 2.0 提供者。

```
aws iam delete-saml-provider \
--saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM SAML 身分提供者。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 DeleteSAMLProvider。

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。

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的 DeleteSAMLProvider。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除其 ARN 所屬的 IAM SAML 2.0 提供者。**arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider**

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- 如需 API 詳細資訊，請參閱指令程式參考中的刪除 SAML 提供者。AWS Tools for PowerShell

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeleteServerCertificate**配 AWS 開發套件或 CLI 使用

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

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。

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " <<
 certificateName
                  << std::endl;
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的DeleteServer憑證。

CLI

AWS CLI

從您的 AWS 帳戶刪除伺服器憑證

下列delete-server-certificate命令會從您的 AWS 帳戶中移除指定的伺服器憑證。

```
aws iam delete-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

此命令不會產生輸出。

若要列出您 AWS 帳戶中可用的伺服器憑證，請使用list-server-certificates指令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的在 IAM 中管理伺服器憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteServer憑證。

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。

刪除伺服器憑證。

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 */
```

```
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- 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DeleteServer憑證。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 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DeleteServer憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除名為的伺服器憑證**MyServerCert**。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteServer憑證。

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。

列出、更新和刪除伺服器憑證。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
```

```
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的DeleteServer憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteServiceLinkedRole配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

刪除服務連結角色

下列 delete-service-linked-role 範例會刪除您不再需要的指定服務連結角色。刪除會以非同步方式發生。您可以使用 get-service-linked-role-deletion-status 命令，檢查刪除狀態並確認刪除的時間。

```
aws iam delete-service-linked-role \
    --role-name AWSServiceRoleForLexBots
```

輸出：

```
{
    "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用服務連結角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteServiceLinkedRole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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}
```

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

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考DeleteServiceLinkedRole中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例刪除服務連結的角色。請注意，如果服務仍在使用此角色，則此命令會導致失敗。

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteServiceLinkedRole式參考中的。

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 設定和執行程式碼範例儲存庫。

```
# Deletes a service-linked role.
#
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)
    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
```

```
  def handle_deletion_error(e, role_name)
    unless e.code == "NoSuchEntity"
      @logger.error("Couldn't delete #{role_name}. Here's why:")
      @logger.error("\t#{e.code}: #{e.message}")
      raise
    end
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteServiceLinkedRole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。

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeleteServiceLinkedRole中的 Rust 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteSigningCertificate配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

刪除 IAM 使用者的簽署憑證

下列delete-signing-certificate命令會刪除名為 IAM 使用者的指定簽署憑證Bob。

```
aws iam delete-signing-certificate \
    --user-name Bob \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE
```

此命令不會產生輸出。

若要取得簽署憑證的 ID，請使用list-signing-certificates指令。

如需詳細資訊，請參閱 Amazon EC2 使用者指南中的管理簽署憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteSigning憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會**Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**從名為的 IAM 使用者刪除具有 ID 的簽署憑證**Bob**。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteSigning憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeleteUser**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteUser。

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：

- 建立群組並新增使用者。
- 建立使用者並擔任角色
- 建立唯讀和讀寫的使用者

.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。

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteUser中的。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
########################################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
```

```
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

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

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteUserRequest request;
    request.SetUserName(userName);
    auto outcome = iam.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting IAM user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;;
    }
```

```
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考DeleteUser中的。

CLI

AWS CLI

刪除 IAM 使用者

下列 `delete-user` 命令會將名為 Bob 的 IAM 使用者從目前帳戶中移除。

```
aws iam delete-user \
    --user-name Bob
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的刪除 IAM 使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteUser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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
```

```
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
 _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考DeleteUser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.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

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

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException 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DeleteUser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。

刪除使用者。

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- 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DeleteUser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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- 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DeleteUser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 deleteIAMUser(userNameVal: String) {
```

```
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除名為的 IAM 使用者**Bob**。

```
Remove-IAMUser -UserName Bob
```

範例 2：此範例會刪除名為的 IAM 使用者，以**Theresa**及必須先刪除的所有元素。

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }
```

```
# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteUser式參考中的。

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。

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.
```

```
    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteUser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 設定和執行程式碼範例儲存庫。

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考DeleteUser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。

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    response
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK DeleteUser中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func deleteUser(user: IAMClientTypes.User) async throws {
    let input = DeleteUserInput(
        userName: user.userName
    )
    do {
        _ = try await iamClient.deleteUser(input: input)
    } catch {
        throw error
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteUser中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DeleteUserPermissionsBoundary**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

從 IAM 使用者刪除許可界限

下列delete-user-permissions-boundary範例會刪除附加至名為 IAM 使用者的許可界限intern。若要將權限界限套用至使用者，請使用put-user-permissions-boundary指令。

```
aws iam delete-user-permissions-boundary \
    --user-name intern
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteUserPermissionsBoundary中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例顯示如何移除附加至 IAM 使用者的權限界限。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteUserPermissionsBoundary式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteUserPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteUserPolicy。

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：

- 建立使用者並擔任角色

.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。

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的DeleteUser政策。

CLI

AWS CLI

從 IAM 使用者中移除政策

下列 `delete-user-policy` 命令會將指定的政策從名為 Bob 的 IAM 使用者中移除。

```
aws iam delete-user-policy \
    --user-name Bob \
```

```
    --policy-name ExamplePolicy
```

此命令不會產生輸出。

若要取得 IAM 使用者的政策清單，請使用 `list-user-policies` 命令。

如需詳細資訊，請參閱 IAM 使用者指南中的在 AWS 帳戶中建立AWS IAM 使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DeleteUser政策。

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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
 error {
 _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
 &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
```

```
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除內嵌於名**AccessToEC2Policy**為 IAM 使用者的內嵌政策**Bob**。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

範例 2：此範例會尋找名為的 IAM 使用者中內嵌的所有內嵌政策，**Theresa**然後將其刪除。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DeleteUser原則。

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。

```
  # Deletes a user and their associated resources
  #
  # @param user_name [String] The name of the user to delete
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
```

```
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的DeleteUser政策。

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。

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的 Rust API 參考DeleteUser政策。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            throw error
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的DeleteUser政策以取得 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`DeleteVirtualMfaDevice`配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

移除虛擬 MFA 裝置

下列delete-virtual-mfa-device命令會從目前帳戶移除指定的 MFA 裝置。

```
aws iam delete-virtual-mfa-device \
    --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的停用 MFA 裝置。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考DeleteVirtualMfaDevice中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除其 ARN 所在的 IAM 虛擬 MFA 裝置。**arn:aws:iam::123456789012:mfa/bob**

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

範例 2：此範例會檢查 IAM 使用者 Theresa 是否已指派 MFA 裝置。如果找到該裝置，IAM 使用者就會停用該裝置。如果設備是虛擬的，則也將其刪除。

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程DeleteVirtualMfaDevice式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DetachGroupPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DetachGroupPolicy。

CLI

AWS CLI

從群組中斷連結原則

此範例會從名Testers為的群組中移除具有 ARN 的arn:aws:iam::123456789012:policy/TesterAccessPolicy受管理原則。

```
aws iam detach-group-policy \
    --group-name Testers \
    --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DetachGroup政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將 ARN 所屬的受管理群組原則與名為的群組**arn:aws:iam::123456789012:policy/TesterAccessPolicy**中斷連結。**Testers**

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

範例 2：此範例會尋找附加至名為群組的所有受管理策略，**Testers**並將它們從群組中分離。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DetachGroup原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DetachRolePolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DetachRolePolicy。

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：

- 建立使用者並擔任角色
- 管理角色

.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。

```
    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的DetachRole政策。

Bash

AWS CLI 與 bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.
```

```
 # bashsupport disable=BP5008
 function usage() {
   echo "function iam_detach_role_policy"
   echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
   echo "  -n role_name   The name of the IAM role."
   echo "  -p policy_ARN -- The IAM policy document ARN."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "n:p:h" option; do
   case "${option}" in
     n) role_name="${OPTARG}" ;;
     p) policy_arn="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$role_name" ]]; then
   errecho "ERROR: You must provide a role name with the -n parameter."
   usage
   return 1
 fi

 if [[ -z "$policy_arn" ]]; then
   errecho "ERROR: You must provide a policy ARN with the -p parameter."
   usage
   return 1
 fi

 response=$(aws iam detach-role-policy \
   --role-name "$role_name" \
   --policy-arn "$policy_arn")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DetachRole政策。

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::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully detached policy " << policyArn << " from role
 "
                  << roleName << std::endl;
```

```
    }

    return detachOutcome.IsSuccess();
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的DetachRole政策。

CLI

AWS CLI

將政策與角色分離

此範例會將具有 ARN arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy 的受管政策從名為 FedTesterRole 的角色中移除。

```
aws iam detach-role-policy \
    --role-name FedTesterRole \
    --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DetachRole政策。

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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
```

```
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
 error {
 _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
 &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}
```

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

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.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * 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 DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

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

        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();
```

```
            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException 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DetachRole政策。

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。

分離政策。

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
```

```
};
```

- 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DetachRole政策。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
```

```
            process.exit();
          }
        });
      }
    });
  }
});
```

- 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DetachRole政策。

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 detachPolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")
    }
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會將 ARN 所屬的受管理群組原則與名為的角色**arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy**中斷連結。**FedTesterRole**

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

範例 2：此範例會尋找附加至名為角色的所有受管理策略，**FedTesterRole**並將其與角色中斷連結。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DetachRole原則。

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。

使用 Boto3 Policy 物件將政策與角色分離。

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
```

```
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

使用 Boto3 Role 物件將政策與角色分離。

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的DetachRole政策 (Boto3) 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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、附加和解除連結角色原則。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的DetachRole政策。

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。

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
```

```
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的 Rust API 參考DetachRole政策。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
IAMClientTypes.Role) async throws {
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            throw error
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的DetachRole政策以取得 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DetachUserPolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DetachUserPolicy。

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：

- 建立唯讀和讀寫的使用者

CLI

AWS CLI

將政策與使用者分離

此範例會將具有 ARN arn:aws:iam::123456789012:policy/TesterPolicy 的受管政策從使用者 Bob 中移除。

```
aws iam detach-user-policy \
    --user-name Bob \
    --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的變更 IAM 使用者的許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DetachUser政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將 ARN 所屬的受管政策與名為**arn:aws:iam::123456789012:policy/TesterPolicy**的 IAM 使用者分離。**Bob**

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

範例 2：此範例會尋找附加至名為 IAM 使用者的所有受管政策，**Theresa**並將這些政策與使用者分離。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DetachUser原則。

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。

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的DetachUser政策 (Boto3) 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 設定和執行程式碼範例儲存庫。

```
 # Detaches a policy from a user
 #
 # @param user_name [String] The name of the user
 # @param policy_arn [String] The ARN of the policy to detach
 # @return [Boolean] true if the policy was successfully detached, false
otherwise
 def detach_user_policy(user_name, policy_arn)
   @iam_client.detach_user_policy(
     user_name: user_name,
     policy_arn: policy_arn
   )
   @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
successfully.")
   true
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Error detaching policy: Policy or user does not exist.")
   false
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error detaching policy from user '#{user_name}':
#{e.message}")
   false
 end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的DetachUser政策。

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。

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的 Rust API 參考DetachUser政策。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`EnableMfaDevice`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`EnableMfaDevice`。

CLI

AWS CLI

若要啟用 MFA 裝置

使用指`create-virtual-mfa-device`令建立新的虛擬 MFA 裝置之後，您可以將 MFA 裝置指派給使用者。下列`enable-mfa-device`範例會將含有序號的 MFA 裝置指派`arn:aws:iam::210987654321:mfa/BobsMFADevice`給使用者Bob。該命令還 AWS 通過從虛擬 MFA 設備中按順序包含前兩個代碼來同步設備。

```
aws iam enable-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 789012
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的啟用虛擬多重要素身份驗證 (MFA) 裝置。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的EnableMfa裝置。

PowerShell

適用的工具 PowerShell

範例 1：此指令會啟用硬體 MFA 裝置與序號，**987654321098**並將裝置與使用者**Bob**建立關聯。它包括從設備順序的前兩個代碼。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

範例 2：此範例會建立並啟用虛擬 MFA 裝置。第一個指令會建立虛擬裝置，並在變數中傳回裝置的物件表示法**$MFADevice**。您可以使用**.Base32StringSeed**或內**QRCodePng**容來設定使用者的軟體應用程式。最後一個指令會將裝置指派給使用者**David**，並透過其序號識別裝置。該命令還 AWS 通過從虛擬 MFA 設備中按順序包含前兩個代碼來同步設備。

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的EnableMfa裝置。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GenerateCredentialReport配 AWS 開發套件或 CLI 使用

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

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：

- 管理您的帳戶

CLI

AWS CLI

產生憑證報告

下列範例會嘗試產生 AWS 帳戶的認證報告。

```
aws iam generate-credential-report
```

輸出：

```
{
    "State":  "STARTED",
    "Description": "No report exists. Starting a new report generation task"
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的取得 AWS 帳戶的登入資料報告。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GenerateCredential報告。

PowerShell

適用的工具 PowerShell

範例 1：此範例要求產生新報告，每四小時可完成一次。如果最後一個報告仍然是最近的報告，則「狀態」欄位**COMPLETE**會 用**Get-IAMCredentialReport**於檢視完成的報告。

```
Request-IAMCredentialReport
```

輸出：

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GenerateCredential報告。

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。

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response
```

```
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的GenerateCredential報表 (Boto3) 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GenerateServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

範例 1：產生自訂原則的服務存取報告

下列generate-service-last-accessed-details範例會啟動背景工作以產生報告，其中列出 IAM 使用者和其他實體存取的服務，其中包含名為的自訂政策intern-boundary。您可以透過執行get-service-last-accessed-details指令來在建立報告後顯示報告。

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::123456789012:policy/intern-boundary
```

輸出：

```
{
    "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"
}
```

範例 2：產生 AWS 受管理 AdministratorAccess 策略的服務存取報告

下列generate-service-last-accessed-details範例會啟動背景工作以產生報告，其中列出 IAM 使用者和具有 AWS 受管AdministratorAccess政策的其他實體存取的服務。您可以透過執行get-service-last-accessed-details指令來在建立報告後顯示報告。

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::aws:policy/AdministratorAccess
```

輸出：

```
{
    "JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的 AWS 使用上次存取的資訊中的精簡許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GenerateServiceLastAccessed詳細資料

PowerShell

適用的工具 PowerShell

範例 1：此範例是 GenerateServiceLastAccessedDetails API 的對等指令程式。這提供了一個可用於獲取 IAM 和 Get-IAM ServiceLastAccessedDetail 的作業 ID ServiceLast AccessedDetail WithEntity

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GenerateServiceLastAccessed詳細資料

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetAccessKeyLastUsed配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetAccessKeyLastUsed。

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：

- 管理存取金鑰

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。

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
 iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱在 AWS SDK for C++ API 參考中GetAccessKeyLast使用。

CLI

AWS CLI

擷取最後一次使用指定存取金鑰之時機的相關資訊

下列範例會擷取最後一次使用存取金鑰 ABCDEXAMPLE 之時機的相關資訊。

```
aws iam get-access-key-last-used \
    --access-key-id ABCDEXAMPLE
```

輸出：

```
{
    "UserName":  "Bob",
    "AccessKeyLastUsed": {
        "Region": "us-east-1",
        "ServiceName": "iam",
        "LastUsedDate": "2015-06-16T22:45:00Z"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

- 如需 API 詳細資訊，請參閱在AWS CLI 命令參考中GetAccessKeyLast使用。

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。

獲取存取金鑰。

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });

  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  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，請參閱在 AWS SDK for JavaScript API 參考中GetAccessKeyLast使用。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });
```

```
iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- 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，請參閱在 AWS SDK for JavaScript API 參考中GetAccessKeyLast使用。

PowerShell

適用的工具 PowerShell

示例 1：返回所提供的訪問密鑰的擁有用戶名和上次使用信息。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- 如需 API 詳細資訊，請參閱在AWS Tools for PowerShell 指令程式參考中GetAccessKeyLast使用。

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。

```
def get_last_use(key_id):
```

```
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- 有關 API 的詳細信息，請參閱在 AWS SDK 中GetAccessKeyLast使用的 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetAccountAuthorizationDetails配 AWS 開發套件或 CLI 使用

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

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：

- 管理您的帳戶

CLI

AWS CLI

列出 AWS 帳戶 IAM 使用者、群組、角色和政策

下列get-account-authorization-details命令會傳回帳 AWS 戶中所有 IAM 使用者、群組、角色和政策的相關資訊。

```
aws iam get-account-authorization-details
```

輸出：

```
{
    "RoleDetailList": [
        {
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "RoleId": "AROA1234567890EXAMPLE",
            "CreateDate": "2014-07-30T17:09:20Z",
            "InstanceProfileList": [
                {
                    "InstanceProfileId": "AIPA1234567890EXAMPLE",
                    "Roles": [
                        {
                            "AssumeRolePolicyDocument": {
                                "Version": "2012-10-17",
                                "Statement": [
                                    {
                                        "Sid": "",
                                        "Effect": "Allow",
                                        "Principal": {
```

```
                                        "Service": "ec2.amazonaws.com"
                                    },
                                    "Action": "sts:AssumeRole"
                                }
                            ]
                        },
                        "RoleId": "AROA1234567890EXAMPLE",
                        "CreateDate": "2014-07-30T17:09:20Z",
                        "RoleName": "EC2role",
                        "Path": "/",
                        "Arn": "arn:aws:iam::123456789012:role/EC2role"
                    }
                ],
                "CreateDate": "2014-07-30T17:09:20Z",
                "InstanceProfileName": "EC2role",
                "Path": "/",
                "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
            }
        ],
        "RoleName": "EC2role",
        "Path": "/",
        "AttachedManagedPolicies": [
            {
                "PolicyName": "AmazonS3FullAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
            },
            {
                "PolicyName": "AmazonDynamoDBFullAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess"
            }
        ],
        "RoleLastUsed": {
            "Region": "us-west-2",
            "LastUsedDate": "2019-11-13T17:30:00Z"
        },
        "RolePolicyList": [],
        "Arn": "arn:aws:iam::123456789012:role/EC2role"
    }
],
"GroupDetailList": [
    {
        "GroupId": "AIDA1234567890EXAMPLE",
        "AttachedManagedPolicies": {
```

```
                "PolicyName": "AdministratorAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
            },
            "GroupName": "Admins",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "CreateDate": "2013-10-14T18:32:24Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "PowerUserAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
            },
            "GroupName": "Dev",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Dev",
            "CreateDate": "2013-10-14T18:33:55Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": [],
            "GroupName": "Finance",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Finance",
            "CreateDate": "2013-10-14T18:57:48Z",
            "GroupPolicyList": [
                {
                    "PolicyName": "policygen-201310141157",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Action": "aws-portal:*",
                                "Sid": "Stmt1381777017000",
                                "Resource": "*",
                                "Effect": "Allow"
                            }
                        ]
                    }
                }
            ]
```

```
        }
    ],
    "UserDetailList": [
        {
            "UserName": "Alice",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:24Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Alice"
        },
        {
            "UserName": "Bob",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:25Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [
                {
                    "PolicyName": "DenyBillingAndIAMPolicy",
                    "PolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": {
                            "Effect": "Deny",
                            "Action": [
                                "aws-portal:*",
                                "iam:*"
                            ],
                            "Resource": "*"
                        }
                    }
                }
            ],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        },
        {
            "UserName": "Charlie",
```

```
            "GroupList": [
                "Dev"
            ],
            "CreateDate": "2013-10-14T18:33:56Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Charlie"
        }
    ],
    "Policies": [
        {
            "PolicyName": "create-update-delete-set-managed-policies",
            "CreateDate": "2015-02-06T19:58:34Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-02-06T19:58:34Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version": "2012-10-17",
                        "Statement": {
                            "Effect": "Allow",
                            "Action": [
                                "iam:CreatePolicy",
                                "iam:CreatePolicyVersion",
                                "iam:DeletePolicy",
                                "iam:DeletePolicyVersion",
                                "iam:GetPolicy",
                                "iam:GetPolicyVersion",
                                "iam:ListPolicies",
                                "iam:ListPolicyVersions",
                                "iam:SetDefaultPolicyVersion"
                            ],
                            "Resource": "*"
                        }
                    },
                    "IsDefaultVersion": true
                }
            ],
```

```
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
            "UpdateDate": "2015-02-06T19:58:34Z"
        },
        {
            "PolicyName": "S3-read-only-specific-bucket",
            "CreateDate": "2015-01-21T21:39:41Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-01-21T21:39:41Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:Get*",
                                    "s3:List*"
                                ],
                                "Resource": [
                                    "arn:aws:s3:::example-bucket",
                                    "arn:aws:s3:::example-bucket/*"
                                ]
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-
bucket",
            "UpdateDate": "2015-01-21T23:39:41Z"
        },
        {
            "PolicyName": "AmazonEC2FullAccess",
            "CreateDate": "2015-02-06T18:40:15Z",
            "AttachmentCount": 1,
```

```
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2014-10-30T20:59:46Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Action": "ec2:*",
                                "Effect": "Allow",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "elasticloadbalancing:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "cloudwatch:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "autoscaling:*",
                                "Resource": "*"
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
            "UpdateDate": "2015-02-06T18:40:15Z"
        }
    ],
    "Marker": "EXAMPLEkakv9BCuUNFDtxWSyfzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
    "IsTruncated": true
```

```
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 AWS 安全性稽核指南。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetAccountAuthorizationDetails中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會取得 AWS 帳戶中身分識別的授權詳細資料，並顯示傳回物件的元素清單，包括使用者、群組和角色。例如，該**UserDetailList**屬性顯示有關用戶的詳細信息。和屬性中提供了類似**RoleDetailList**的**GroupDetailList**資訊。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

輸出：

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

輸出：

```
Arn             : arn:aws:iam::123456789012:user/Administrator
CreateDate      : 10/16/2014 9:03:09 AM
GroupList       : {Administrators}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE1
UserName        : Administrator
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
GroupList       : {Developers}
Path            : /
```

```
UserId          : AIDACKCEVSQ6CEXAMPLE2
UserName        : bab
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/BackupToS3
CreateDate      : 1/27/2015 10:15:08 AM
GroupList       : {Backup}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE3
UserName        : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetAccountAuthorizationDetails式參考中的。

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。

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
```

```
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetAccountAuthorizationDetails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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetAccountPasswordPolicy配 AWS 開發套件或 CLI 使用

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

.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。

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetAccountPasswordPolicy中的。

CLI

AWS CLI

查看目前的帳戶密碼政策

下列 `get-account-password-policy` 命令會顯示有關目前帳戶密碼政策的詳細資訊。

```
aws iam get-account-password-policy
```

輸出：

```
{
    "PasswordPolicy": {
        "AllowUsersToChangePassword": false,
        "RequireLowercaseCharacters": false,
        "RequireUppercaseCharacters": false,
        "MinimumPasswordLength": 8,
        "RequireNumbers": true,
        "RequireSymbols": true
    }
}
```

如果沒有為帳戶定義密碼政策，則該命令會傳回 `NoSuchEntity` 錯誤。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的設定 IAM 使用者的帳戶密碼政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetAccountPasswordPolicy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。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
 error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

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

取得帳戶密碼政策。

```
import {
  GetAccountPasswordPolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});

  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetAccountPasswordPolicy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。

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考GetAccountPasswordPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回目前帳戶之密碼策略的詳細資料。如果沒有為帳戶定義密碼策略，命令會傳回NoSuchEntity錯誤。

```
Get-IAMAccountPasswordPolicy
```

輸出：

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetAccountPasswordPolicy式參考中的。

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。

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
```

```
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
{pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention:
{pw_policy.password_reuse_prevention}")
        print(
            f"\trequire_lowercase_characters:
{pw_policy.require_lowercase_characters}"
        )
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
            f"\trequire_uppercase_characters:
{pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetAccountPasswordPolicy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 設定和執行程式碼範例儲存庫。

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "IAMPolicyManager"
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
    begin
      response = @iam_client.get_account_password_policy
      @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
    rescue Aws::IAM::Errors::NoSuchEntity
      @logger.info("The account does not have a password policy.")
    rescue Aws::Errors::ServiceError => e
      @logger.error("Couldn't print the account password policy. Error: #{e.code}
 - #{e.message}")
      raise
    end
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetAccountPasswordPolicy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。

```
pub async fn get_account_password_policy(
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
 SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK GetAccountPasswordPolicy中的 Rust 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**GetAccountSummary**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`GetAccountSummary`。

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：

- 管理您的帳戶

CLI

AWS CLI

取得有關目前帳戶中 IAM 實體用量和 IAM 配額的資訊

下列 get-account-summary 命令會傳回有關帳戶中目前 IAM 實體用量和目前 IAM 實體配額的資訊。

```
aws iam get-account-summary
```

輸出：

```
{
    "SummaryMap": {
        "UsersQuota": 5000,
        "GroupsQuota": 100,
        "InstanceProfiles": 6,
        "SigningCertificatesPerUserQuota": 2,
        "AccountAccessKeysPresent": 0,
        "RolesQuota": 250,
        "RolePolicySizeQuota": 10240,
        "AccountSigningCertificatesPresent": 0,
        "Users": 27,
        "ServerCertificatesQuota": 20,
        "ServerCertificates": 0,
        "AssumeRolePolicySizeQuota": 2048,
        "Groups": 7,
        "MFADevicesInUse": 1,
        "Roles": 3,
        "AccountMFAEnabled": 1,
        "MFADevices": 3,
        "GroupsPerUserQuota": 10,
        "GroupPolicySizeQuota": 5120,
        "InstanceProfilesQuota": 100,
        "AccessKeysPerUserQuota": 2,
        "Providers": 0,
        "UserPolicySizeQuota": 2048
    }
}
```

如需實體限制的詳細資訊，請參閱 IAM 使用者指南中的AWS IAM 和 AWS STS 配額。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetAccount摘要。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回中目前 IAM 實體用量和目前 IAM 實體配額的相關資訊 AWS 帳戶。

```
Get-IAMAccountSummary
```

輸出：

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
InstanceProfilesQuota                   100
MFADevices                              4
AccessKeysPerUserQuota                  2
AttachedPoliciesPerUserQuota            2
SigningCertificatesPerUserQuota         2
PolicyVersionsInUse                     4
```

```
InstanceProfiles                          1
...
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetAccount摘要。

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。

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- 如需 API 的詳細資訊，請參閱 AWS SDK 中的GetAccount摘要 (Boto3) 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetContextKeysForCustomPolicy配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

範例 1：若要在命令列上列出由一或多個自訂 JSON 原則所參照的內容索引鍵，做為參數提供

下列get-context-keys-for-custom-policy命令會剖析每個提供的原則，並列出這些原則使用的內容索引鍵。使用此命令可識別您必須提供哪些內容索引鍵值，才能順利使用原則模擬器命令simulate-custom-policy和simulate-custom-policy。您也可以使用get-context-keys-for-custom-policy命令擷取 IAM 使用者或角色關聯的所有政策使用的內容金鑰清單。以開頭的參數值會file://指示命令讀取檔案，並使用內容做為參數的值，而不是檔案名稱本身。

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

輸出：

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

範例 2：列出以檔案輸入形式提供的一或多個自訂 JSON 原則所參考的內容索引鍵

下列get-context-keys-for-custom-policy命令與上一個範例相同，不同之處在於原則是以檔案而非參數的形式提供。由於該命令需要 JSON 字符串列表，而不是 JSON 結構列表，因此文件必須按以下方式構造，儘管您可以將其折疊為一個。

```
[
    "Policy1",
    "Policy2"
```

```
]
```

例如，包含上一個範例中原則的檔案必須如下所示。您必須在原則字串中加上反斜線」，以逸出原則字串內嵌的每個內嵌雙引號。

```
[ "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow
\", \"Action\": \"dynamodb:*\", \"Resource\": \"arn:aws:dynamodb:us-
west-2:128716708097:table/${aws:username}\", \"Condition\": {\"DateGreaterThan\":
 {\"aws:CurrentTime\": \"2015-08-16T12:00:00Z\"}}}}" ]
```

然後可以將此檔案提交至下列命令。

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list file://policyfile.json
```

輸出：

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的使用AWS IAM 政策模擬器 (AWS CLI 和 AWS API)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetContextKeysForCustomPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取所提供原則 JSON 中存在的所有內容索引鍵。為了提供多個原則，您可以以逗號分隔的值清單提供。

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

```
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetContextKeysForCustomPolicy式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetContextKeysForPrincipalPolicy配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

列出與 IAM 主體相關聯的所有政策所參考的內容金鑰

下列get-context-keys-for-principal-policy命令會擷取附加至使用者的所有策略，以saanvi及她所屬的任何群組。然後它會剖析每個項目，並列出這些原則使用的內容索引鍵。使用此指令可識別您必須提供哪些前後關聯索引鍵值，才能順利使用simulate-custom-policy和simulate-principal-policy指令。您也可以使用get-context-keys-for-custom-policy命令擷取任意 JSON 政策使用的內容金鑰清單。

```
aws iam get-context-keys-for-principal-policy \
   --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

輸出：

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的使用AWS IAM 政策模擬器 (AWS CLI 和 AWS API)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetContextKeysForPrincipalPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取所提供政策 json 中存在的所有內容金鑰，以及附加至 IAM 實體的政策 (使用者/角色等)。For-PolicyInputList 您可以將多個值清單提供為逗號分隔值。

```
$policy1 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17","Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetContextKeysForPrincipalPolicy式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetCredentialReport配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetCredentialReport。

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：

- 管理您的帳戶

CLI

AWS CLI

取得憑證報告

此範例會開啟傳回的報告，並以文字行陣列的形式將其輸出至管道。

```
aws iam get-credential-report
```

輸出：

```
{
    "GeneratedTime":  "2015-06-17T19:11:50Z",
    "ReportFormat": "text/csv"
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的取得 AWS 帳戶的登入資料報告。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetCredential報告。

PowerShell

適用的工具 PowerShell

範例 1：此範例會開啟傳回的報表，並以文字行陣列的形式將其輸出至管線。第一行是帶有逗號分隔列名的標題。連續的每一列都是一位使用者的詳細資料列，每個欄位都以逗號分隔。您必須先使用**Request-IAMCredentialReport**指令程式產生報告，才能檢視報告。若要將報表擷取為單一字串，請使用**-Raw**而非**-AsTextArray**。**-AsTextArray**交換**-SplitLines**器也接受別名。如需輸出中資料行的完整清單，請參閱服務 API 參考資料。請注意，如果您不使用**-AsTextArray**或**-SplitLines**，則必須使用 .NET **StreamReader** 類別從**.Content**屬性中擷取文字。

```
Request-IAMCredentialReport
```

輸出：

```
Description                                                     State
-----------                                                     -----
No report exists. Starting a new report generation task         STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

輸出：

```
user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
```

```
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetCredential報告。

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。

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的GetCredential報表 (Boto3) 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetGroup配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetGroup。

CLI

AWS CLI

若要取得 IAM 群組

此範例會傳回 IAM 群組的詳細資訊Admins。

```
aws iam get-group \
    --group-name Admins
```

輸出：

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-06-16T19:41:48Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    },
    "Users": []
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 身分 (使用者、使用者群組和角色)。AWS

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetGroup中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例會傳回 IAM 群組的詳細資訊`Testers`，包括屬於該群組的所有 IAM 使用者的集合。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

輸出：

```
Group                                    IsTruncated           Marker
     Users
-----                                    -----------           ------
     -----
Amazon.IdentityManagement.Model.Group    False
     {Theresa, David}
```

```
$results.Group
```

輸出：

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

輸出：

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa
```

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetGroup式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetGroupPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetGroupPolicy。

CLI

AWS CLI

取得附加至 IAM 群組之政策的相關資訊

下列get-group-policy命令會取得附加至名為群組之指定原則的相關資訊Test-Group。

```
aws iam get-group-policy \
    --group-name Test-Group \
    --policy-name S3-ReadOnly-Policy
```

輸出：

```
{
    "GroupName": "Test-Group",
    "PolicyDocument": {
        "Statement": [
            {
                "Action": [
                    "s3:Get*",
                    "s3:List*"
                ],
                "Resource": "*",
                "Effect": "Allow"
            }
```

```
        ]
    },
    "PolicyName": "S3-ReadOnly-Policy"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetGroup政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回**PowerUserAccess-Testers**針對群組命名的內嵌內嵌原則的詳細資料**Testers**。該**PolicyDocument**屬性是 URL 編碼的。它在這個例子中使用 **UrlDecode** .NET 方法進行解碼。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

輸出：

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetGroup原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetInstanceProfile配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetInstanceProfile。

CLI

AWS CLI

取得執行個體設定檔的相關資訊

下列get-instance-profile命令會取得名為之執行個體設定檔的相關資訊ExampleInstanceProfile。

```
aws iam get-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

輸出：

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AID2MAB8DPLSRHEXAMPLE",
        "Roles": [
            {
                "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                "RoleId": "AIDGPMS9RO4H3FEXAMPLE",
                "CreateDate": "2013-01-09T06:33:26Z",
                "RoleName": "Test-Role",
                "Path": "/",
                "Arn": "arn:aws:iam::336924118301:role/Test-Role"
            }
        ],
        "CreateDate": "2013-06-12T23:52:02Z",
        "InstanceProfileName": "ExampleInstanceProfile",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:instance-profile/
ExampleInstanceProfile"
```

```
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用執行個體設定檔。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetInstance設定檔。

PowerShell

用於的工具 PowerShell

範例 1：此範例會傳回目前 AWS 帳戶中定義的名稱為**ec2instancerole**之執行個體設定檔的詳細資訊。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

輸出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetInstance設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**GetLoginProfile**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetLoginProfile。

CLI

AWS CLI

取得 IAM 使用者的密碼資訊

下列get-login-profile命令會取得名為 IAM 使用者之密碼的相關資訊Bob。

```
aws iam get-login-profile \
    --user-name Bob
```

輸出：

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2012-09-21T23:03:39Z"
    }
}
```

此命get-login-profile令可用來驗證 IAM 使用者是否擁有密碼。如果沒有為使用者定義密碼，命令會傳回NoSuchEntity錯誤。

您無法使用此命令檢視密碼。如果密碼遺失，您可以重設使用者的密碼 (update-login-profile)。或者，您可以刪除使用者的登入設定檔 (delete-login-profile)，然後建立新的 (create-login-profile)。

如需詳細資訊，請參閱 IAM 使用者指南中的管理AWS IAM 使用者的密碼。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetLogin設定檔。

PowerShell

用於的工具 PowerShell

範例 1：此範例會傳回密碼建立日期，以及 IAM 使用者是否需要重設密碼**David**。

```
Get-IAMLoginProfile -UserName David
```

輸出：

```
CreateDate                 PasswordResetRequired                UserName
----------                 ---------------------                --------
12/10/2014 3:39:44 PM      False                                David
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetLogin設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetOpenIdConnectProvider配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要傳回有關指定 OpenID Connect 提供者的資訊

此範例會傳回有關其 ARN 為 arn:aws:iam::123456789012:oidc-provider/server.example.com OpenID Connect 提供者的詳細資訊。

```
aws iam get-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

輸出：

```
{
    "Url": "server.example.com"
        "CreateDate": "2015-06-16T19:41:48Z",
        "ThumbprintList": [
        "12345abcdefghijk67890lmnopqrst987example"
        ],
        "ClientIDList": [
        "example-application-ID"
        ]
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的建立 OpenID Connect (OIDC) 身分識別提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetOpenIdConnect提供者。

PowerShell

用於的工具 PowerShell

示例 1：此示例返回有關其 ARN 為 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** OpenID Connect 提供程序的詳細信息。該**ClientIDList**屬性是一個集合，其中包含為此提供程序定義的所有客戶端 ID。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

輸出：

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetOpenIdConnect提供者。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**GetPolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetPolicy。

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：

- 使用 IAM 政策產生器 API

.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。

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetPolicy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。

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
```

```
                                const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
                  policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考GetPolicy中的。

CLI

AWS CLI

擷取指定受管政策的相關資訊

此範例會傳回其 ARN 為 `arn:aws:iam::123456789012:policy/MySamplePolicy` 之受管政策的相關詳細資訊。

```
aws iam get-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

輸出：

```
{
    "Policy": {
        "PolicyName": "MySamplePolicy",
        "CreateDate": "2015-06-17T19:23;32Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "Z27SI6FQMGNQ2EXAMPLE1",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",
        "UpdateDate": "2015-06-17T19:23:32Z"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetPolicy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。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
```

```
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

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

取得政策。

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
  });

  return client.send(command);
```

```
};
```

- 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GetPolicy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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- 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GetPolicy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 getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetPolicy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。

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
```

```
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考GetPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回其 ARN 為arn:aws:iam::123456789012:policy/MySamplePolicy受管理原則的詳細資料。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

輸出：

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetPolicy式參考中的。

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。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetPolicy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 設定和執行程式碼範例儲存庫。

```
  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
```

```
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetPolicy中的。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
```

```
            }
            return policy
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetPolicy中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetPolicyVersion配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetPolicyVersion。

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：

- 管理政策
- 使用 IAM 政策產生器 API

CLI

AWS CLI

擷取指定受管政策指定版本的相關資訊

此範例會傳回 ARN 為 arn:aws:iam::123456789012:policy/MyManagedPolicy 之 v2 政策版本的政策文件。

```
aws iam get-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

輸出：

```
{
```

```
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "iam:*",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2023-04-11T00:22:54+00:00"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetPolicy版本。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回其 ARN 為原則**v2**版本的原則文件。**arn:aws:iam::123456789012:policy/MyManagedPolicyDocument**屬性中的原則文件經過 URL 編碼，並在此範例中使用 **UrlDecode** .NET 方法解碼。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

輸出：

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version": "2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    }
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetPolicy版本。

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。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
```

```
        raise
    else:
        return policy_statement
```

- 如需 API 的詳細資訊，請參閱適用於 Python (Boto3) API 參考的AWS SDK GetPolicy 版本。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetRole配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetRole。

.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。

```
/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
        RoleName = roleName,
    });

    return response.Role;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetRole中的。

CLI

AWS CLI

取得 IAM 角色的相關資訊

下列 get-role 命令會取得名為 Test-Role 之角色的相關資訊。

```
aws iam get-role \
    --role-name Test-Role
```

輸出：

```
{
    "Role": {
        "Description": "Test Role",
        "AssumeRolePolicyDocument":"<URL-encoded-JSON>",
        "MaxSessionDuration": 3600,
        "RoleId": "AROA1234567890EXAMPLE",
        "CreateDate": "2019-11-13T16:45:56Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "RoleLastUsed": {
            "Region": "us-east-1",
            "LastUsedDate": "2019-11-13T17:14:00Z"
        },
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

該命令會顯示連接至角色的信任政策。若要列出連接至角色的許可政策，請使用 list-role-policies 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetRole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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(context.TODO(),
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

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

取得角色。

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetRole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。

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考GetRole中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例會傳回的詳細資訊lamda_exec_role。其中包含指定誰可以擔任此角色的信任原則文件。原則文件經過 URL 編碼，而且可以使用 .NET UrlDecode 方法來解碼。在此範例中，原始策略在上傳到策略之前已移除所有空格。若要查看決定擔任該角色之人員可執行的權限原則文件，請使Get-IAMRolePolicy用內嵌原則和附加Get-IAMPolicyVersion的受管理原則。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

輸出：

```
Arn                       : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument  : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                            %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                            %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                            %22%7D%5D%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

輸出：

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

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetRole式參考中的。

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。

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetRole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 設定和執行程式碼範例儲存庫。

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                               role_name: name,
                             }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetRole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。

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK GetRole中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
            roleName: name
```

```
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            throw error
        }
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetRole中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**GetRolePolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetRolePolicy。

CLI

AWS CLI

取得附加至 IAM 角色之政策的相關資訊

下列get-role-policy命令會取得附加至名為之角色之指定原則的相關資訊Test-Role。

```
aws iam get-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

輸出：

```
{
  "RoleName": "Test-Role",
  "PolicyDocument": {
      "Statement": [
          {
```

```
                "Action": [
                    "s3:ListBucket",
                    "s3:Put*",
                    "s3:Get*",
                    "s3:*MultipartUpload*"
                ],
                "Resource": "*",
                "Effect": "Allow",
                "Sid": "1"
            }
        ]
    }
    "PolicyName": "ExamplePolicy"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetRole政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回 IAM 角色中內嵌之名為**oneClick_lambda_exec_role_policy**之政策的許可政策文件**lamda_exec_role**。產生的策略文件會經過 URL 編碼。它在這個例子中使用 **UrlDecode** .NET 方法進行解碼。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

輸出：

```
PolicyDocument                                                  PolicyName
              UserName
--------------                                                  ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
```

```
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

輸出：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::*"
      ]
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetRole原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**GetSamlProvider**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`GetSamlProvider`。

CLI

AWS CLI

若要擷取 SAML 提供者中繼資料

此範例會擷取有關 ARM 所在之 SAML 2.0 提供者的詳細資料。arn:aws:iam::123456789012:saml-provider/SAMLADFS回應包括您從身分識別提供者取得的中繼資料文件，以建立 AWS SAML 提供者實體，以及建立和到期日期。

```
aws iam get-saml-provider \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

輸出：

```
{
    "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",
    "CreateDate": "2017-03-06T22:29:46+00:00",
    "ValidUntil": "2117-03-06T22:29:46.433000+00:00",
    "Tags": [
        {
            "Key": "DeptID",
            "Value": "123456"
        },
        {
            "Key": "Department",
            "Value": "Accounting"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM SAML 身分提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetSaml提供者。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取有關其 ARM 為 arn: aw: iam:: 123456789012：Saml 提供者/SamlaDFS 提供者的詳細資料。回應包括您從身分識別提供者取得的中繼資料文件，以建立 AWS SAML 提供者實體，以及建立和到期日期。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

輸出：

```
CreateDate                  SAMLMetadataDocument
        ValidUntil
----------                  --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetSamlI提供者。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetServerCertificate配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetServerCertificate。

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。

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
```

```
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                  << std::endl << "Body: " << certificate.GetCertificateBody() <<
                  std::endl << "Chain: " << certificate.GetCertificateChain() <<
                  std::endl;
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的GetServer憑證。

CLI

AWS CLI

取得 AWS 帳戶中伺服器憑證的詳細資料

下列get-server-certificate命令會擷取有關您 AWS 帳戶中指定伺服器憑證的所有詳細資料。

```
aws iam get-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

輸出：

```
{
    "ServerCertificate": {
        "ServerCertificateMetadata": {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
```

```
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        "CertificateBody": "-----BEGIN CERTIFICATE-----
            MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
            VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
            b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
            BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
            MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
            VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
            b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
            rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
            Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
            nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
            FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
            NYiytVbZPQUQ5Yaxu2jXnimvrszlaEXAMPLE=-----END CERTIFICATE-----",
        "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIICiTCCAfICCQD6md
            7oRw0uXOjANBgkqhkiG9w0BAqQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgT
            AldBMRAwDgYDVQQHEwdTZWF0drGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAs
            TC0lBTSBDb25zb2xlMRIwEAYDVsQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQ
            jb20wHhcNMTEwNDI1MjA0NTIxWhtcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
            MCVVMxCzAJBgNVBAgTAldBMRAwDgsYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
            WF6b24xFDASBgNVBAsTC0lBTSBDb2d5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
            HzAdBgkqhkiG9w0BCQEWEG5vb25lQGfFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
            BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
            k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8mh9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
            ITxOUSQv7c7ugFFDzQGBzZswY6786m86gjpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
            AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUhVVxYUntneD9+h8Mg9q6q+auN
            KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FlkbFFBjvSfpJIlJ00zbhNYS5f6Guo
            EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
            3rrszlaEWEG5vb25lQGFtsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
    }
}
```

若要列出您 AWS 帳戶中可用的伺服器憑證，請使用`list-server-certificates`指令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的在 IAM 中管理伺服器憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetServer憑證。

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。

獲取伺服器憑證。

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 * @returns
 */
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  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GetServer憑證。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 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GetServer憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例擷取名為的伺服器憑證的詳細資料MyServerCertificate。您可以在CertificateBody和內容中找到憑證詳細資ServerCertificateMetadata料。

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

輸出：

```
CertificateBody          : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
```

```
 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

輸出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetServer憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetServiceLastAccessedDetails配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

擷取服務存取報告

下列get-service-last-accessed-details範例會擷取先前產生的報表，其中列出 IAM 實體存取的服務。若要產生報告，請使用generate-service-last-accessed-details指令。

```
aws iam get-service-last-accessed-details \
    --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

輸出：

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:50:35.929Z",
    "ServicesLastAccessed": [
        ...
        {
            "ServiceName": "AWS Lambda",
            "LastAuthenticated": "2019-09-30T23:02:00Z",
            "ServiceNamespace": "lambda",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",
            "TotalAuthenticatedEntities": 6
        },
    ]
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的 AWS 使用上次存取資訊中的精簡許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetServiceLastAccessed詳細資料

PowerShell

適用的工具 PowerShell

範例 1：此範例提供要求呼叫中關聯的 IAM 實體 (使用者、群組、角色或政策) 上次存取之服務的詳細資料。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

輸出：

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetServiceLastAccessed詳細資料

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetServiceLastAccessedDetailsWithEntities配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

擷取包含服務詳細資訊的服務存取報告

下列get-service-last-accessed-details-with-entities範例會擷取報表，其中包含存取指定服務的 IAM 使用者和其他實體的詳細資料。若要產生報告，請使用generate-service-last-accessed-details指令。若要取得使用命名空間存取的服務清單，請使用get-service-last-accessed-details.

```
aws iam get-service-last-accessed-details-with-entities \
    --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \
    --service-namespace lambda
```

輸出：

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:55:41.756Z",
    "JobCompletionDate": "2019-10-01T03:55:42.533Z",
    "EntityDetailsList": [
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/admin",
                "Name": "admin",
                "Type": "USER",
                "Id": "AIDAIO2XMPLENQEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-30T23:02:00Z"
        },
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/developer",
                "Name": "developer",
                "Type": "USER",
                "Id": "AIDAIBEYXMPL2YEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-16T19:34:00Z"
        }
    ]
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的 AWS 使用上次存取的資訊中的精簡許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetServiceLastAccessedDetailsWith實體。

PowerShell

適用的工具 PowerShell

範例 1：此範例會針對該個別 IAM 實體在要求中提供服務上次存取的時間戳記。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
```

```
$results
```

輸出：

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

輸出：

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

輸出：

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetServiceLastAccessedDetailsWith實體。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetServiceLinkedRoleDeletionStatus配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

檢查刪除服務連結角色的請求狀態

下列 `get-service-linked-role-deletion-status` 範例會顯示刪除服務連結角色之先前請求的狀態。刪除操作會以非同步方式發生。提出請求時，就會取得您提供作為此命令參數的 `DeletionTaskId` 值。

```
aws iam get-service-linked-role-deletion-status \
    --deletion-task-id task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

輸出：

```
{
"Status": "SUCCEEDED"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用服務連結角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetServiceLinkedRoleDeletionStatus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。

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});
```

```
/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考GetServiceLinkedRoleDeletionStatus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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetUser配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetUser。

.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。

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
```

```
        var response = await _IAMService.GetUserAsync(new GetUserRequest
 { UserName = userName });
        return response.User;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetUser中的。

Bash

AWS CLI 使用 Bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
##############################################################################
function iam_user_exists() {
```

```
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}
```

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

CLI

AWS CLI

取得關於 IAM 使用者的資訊

下列 `get-user` 命令會取得名為 `Paulo` 之 IAM 使用者的相關資訊。

```
aws iam get-user \
    --user-name Paulo
```

輸出：

```
{
    "User": {
```

```
        "UserName": "Paulo",
        "Path": "/",
        "CreateDate": "2019-09-21T23:03:13Z",
        "UserId": "AIDA123456789EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Paulo"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考GetUser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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
```

```
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取有關具名使用者的詳細資料David。

```
Get-IAMUser -UserName David
```

輸出：

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

範例 2：此範例會擷取有關目前登入的 IAM 使用者的詳細資料。

```
Get-IAMUser
```

輸出：

```
Arn              : arn:aws:iam::123456789012:user/Bob
```

```
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程GetUser式參考中的。

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 設定和執行程式碼範例儲存庫。

```
 # Retrieves a user's details
 #
 # @param user_name [String] The name of the user to retrieve
 # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考GetUser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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetUserPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetUserPolicy。

CLI

AWS CLI

列出 IAM 使用者的政策詳細資料

下列get-user-policy命令會列出附加至名為 IAM 使用者之指定政策的詳細資料Bob。

```
aws iam get-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

輸出：

```
{
    "UserName": "Bob",
    "PolicyName": "ExamplePolicy",
    "PolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Action": "*",
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    }
}
```

若要取得 IAM 使用者的政策清單，請使用 list-user-policies 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetUser政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例擷取內嵌於名為 **Davids_IAM_Admin_Policy** IAM 使用者的內嵌政策詳細資料**David**。策略文件是 URL 編碼的。

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

輸出：

```
PolicyDocument                                                  PolicyName
       UserName
--------------                                                  ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的GetUser原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListAccessKeys**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListAccessKeys。

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：

- 管理存取金鑰

Bash

AWS CLI 與 Bash 腳本

> **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。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
```

```
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for
 the specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam list-access-keys \
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListAccess金鑰。

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。

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
```

```
                        std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                        std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的ListAccess金鑰。

CLI

AWS CLI

列出 IAM 使用者的存取金鑰 ID

下列 `list-access-keys` 命令會列出名為 Bob 之 IAM 使用者的存取金鑰 ID。

```
aws iam list-access-keys \
```

```
    --user-name Bob
```

輸出：

```
{
    "AccessKeyMetadata": [
        {
            "UserName": "Bob",
            "Status": "Active",
            "CreateDate": "2013-06-04T18:17:34Z",
            "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
        },
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CreateDate": "2013-06-06T20:42:26Z",
            "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"
        }
    ]
}
```

您無法列出 IAM 使用者的私密存取金鑰。如果私密存取金鑰遺失，您必須使用 `create-access-keys` 命令建立新的存取金鑰。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListAccess金鑰。

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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
```

```
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
 &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

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

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.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.iam.IamClient;

/**
 * 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 ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
 retrieved.\s
                """;

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

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;
```

```
            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request =
 ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request =
 ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException 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ListAccess金鑰。

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。

列出存取金鑰。

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput |
 undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
```

```
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 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ListAccess金鑰。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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ListAccess金鑰。

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 listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的ListAccess密鑰以獲取 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：此命令列出名為的 IAM 使用者的存取金鑰**Bob**。請注意，您無法列出 IAM 使用者的秘密存取金鑰。如果密碼存取金鑰遺失，您必須使用**New-IAMAccessKey**指令程式建立新的存取金鑰。

```
Get-IAMAccessKey -UserName "Bob"
```

輸出：

```
AccessKeyId                CreateDate                  Status
 UserName
-----------                ----------                  ------
 --------
AKIAIOSFODNN7EXAMPLE       12/3/2014 10:53:41 AM       Active              Bob
AKIAI44QH8DHBEXAMPLE       6/6/2013 8:42:26 PM         Inactive            Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListAccess金鑰。

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。

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- 如需 API 詳細資訊，請參閱AWS 開發套件中的ListAccess金鑰 (Boto3) 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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、停用及刪除存取金鑰。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end
```

```
  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded => e
    @logger.error("Error creating access key: limit exceeded. Cannot create
more.")
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: "Inactive"
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
```

```
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的ListAccess金鑰。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListAccountAliases配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListAccountAliases。

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：

- 管理您的帳戶

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。

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;
```

```
    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的ListAccount別名。

CLI

AWS CLI

列出帳戶別名

下列 `list-account-aliases` 命令會列出目前帳戶的別名。

```
aws iam list-account-aliases
```

輸出：

```
{
    "AccountAliases": [
    "mycompany"
    ]
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的您的 AWS 帳戶 ID 及其別名。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListAccount別名。

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.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * 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 ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

        } catch (IamException 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ListAccount別名。

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。

列出帳戶別名。

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 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ListAccount別名。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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ListAccount別名。

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 listAliases() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK 中的ListAccount別名以獲取 Kotlin API 參考。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回的帳戶別名 AWS 帳戶。

```
Get-IAMAccountAlias
```

輸出：

```
ExampleCo
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListAccount別名。

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。

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
```

```
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的ListAccount別名，以供 Python (Boto3) API 參考使用。

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。

列出、建立及刪除帳戶別名。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
```

```
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的ListAccount別名。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListAttachedGroupPolicies**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

列出附加至指定群組的所有受管理策略

此範例會傳回附加至 AWS 帳戶Admins中指定之 IAM 群組之受管政策的名稱和 ARN。

```
aws iam list-attached-group-policies \
    --group-name Admins
```

輸出：

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListAttachedGroupPolicies中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回附加至 AWS 帳戶**Admins**中指定的 IAM 群組之受管政策的名稱和 ARN。若要查看內嵌在群組中的內嵌原則清單，請使用**Get-IAMGroupPolicyList**指令。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

輸出：

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/SecurityAudit                      SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess                AdministratorAccess
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListAttachedGroupPolicies式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListAttachedRolePolicies**配 AWS 開發套件或 CLI 使用

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

.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。

```
    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
```

```
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListAttachedRolePolicies中的。

CLI

AWS CLI

列出連接至指定角色的所有受管政策

此命令會傳回附加至 AWS 帳戶SecurityAuditRole中指定之 IAM 角色之受管政策的名稱和 ARN。

```
aws iam list-attached-role-policies \
    --role-name SecurityAuditRole
```

輸出：

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListAttachedRolePolicies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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
 ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

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

列出連接至角色的政策。

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
```

```
        }),
      );
    } else {
      break;
    }
  }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListAttachedRolePolicies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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker
 = "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

```
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListAttachedRolePolicies中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會傳回附加至 AWS 帳戶**SecurityAuditRole**中指定之 IAM 角色之受管政策的名稱和 ARN。若要查看內嵌在角色中的內嵌原則清單，請使用**Get-IAMRolePolicyList**命令。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

輸出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListAttachedRolePolicies式參考中的。

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。

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
```

```
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListAttachedRolePolicies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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、附加和解除連結角色原則。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
```

```
 def create_policy(policy_name, policy_document)
   response = @iam_client.create_policy(
     policy_name: policy_name,
     policy_document: policy_document.to_json
   )
   response.policy.arn
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating policy: #{e.message}")
   nil
 end

 # Fetches an IAM policy by its ARN
 # @param policy_arn [String] the ARN of the IAM policy to retrieve
 # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
 def get_policy(policy_arn)
   response = @iam_client.get_policy(policy_arn: policy_arn)
   policy = response.policy
   @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
   policy
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
   raise
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
   raise
 end

 # Attaches a policy to a role
 #
 # @param role_name [String] The name of the role
 # @param policy_arn [String] The policy ARN
 # @return [Boolean] true if successful, false otherwise
 def attach_policy_to_role(role_name, policy_arn)
   @iam_client.attach_role_policy(
     role_name: role_name,
     policy_arn: policy_arn
   )
   true
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error attaching policy to role: #{e.message}")
   false
```

```
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考[ListAttachedRolePolicies](#)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](#)。

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
 SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListAttachedRolePolicies中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    /// Returns a list of AWS Identity and Access Management (IAM) policies
```

```
    /// that are attached to the role.
    ///
    /// - Parameter role: The IAM role to return the policy list for.
    ///
    /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
    ///   describing each managed policy that's attached to the role.
    public func listAttachedRolePolicies(role: String) async throws ->
[IAMClientTypes.AttachedPolicy] {
        var policyList: [IAMClientTypes.AttachedPolicy] = []
        var marker: String? = nil
        var isTruncated: Bool

        repeat {
            let input = ListAttachedRolePoliciesInput(
                marker: marker,
                roleName: role
            )
            let output = try await client.listAttachedRolePolicies(input: input)

            guard let attachedPolicies = output.attachedPolicies else {
                return policyList
            }

            for attachedPolicy in attachedPolicies {
                policyList.append(attachedPolicy)
            }
            marker = output.marker
            isTruncated = output.isTruncated
        } while isTruncated == true
        return policyList
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListAttachedRolePolicies中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListAttachedUserPolicies**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

列出附加至指定使用者的所有受管理策略

此命令會針對 AWS 帳戶Bob中指定的 IAM 使用者傳回受管政策的名稱和 ARN。

```
aws iam list-attached-user-policies \
    --user-name Bob
```

輸出：

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListAttachedUserPolicies中的。

PowerShell

適用的工具 PowerShell

範例 1：此命令會針對 AWS 帳戶**Bob**中指定的 IAM 使用者傳回受管政策的名稱和 ARN。若要查看 IAM 使用者中內嵌的內嵌政策清單，請使用**Get-IAMUserPolicyList**命令。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

輸出：

```
PolicyArn                                                  PolicyName
---------                                                  ----------
arn:aws:iam::aws:policy/TesterPolicy                       TesterPolicy
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListAttachedUserPolicies式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListEntitiesForPolicy**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListEntitiesForPolicy。

CLI

AWS CLI

列出指定受管理策略所附加的所有使用者、群組和角色

此範例會傳回已arn:aws:iam::123456789012:policy/TestPolicy附加政策的 IAM 群組、角色和使用者清單。

```
aws iam list-entities-for-policy \
    --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

輸出：

```
{
    "PolicyGroups": [
        {
            "GroupName": "Admins",
            "GroupId": "AGPACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyUsers": [
        {
            "UserName": "Alice",
            "UserId": "AIDACKCEVSQ6C2EXAMPLE"
        }
```

```
    ],
    "PolicyRoles": [
        {
            "RoleName": "DevRole",
            "RoleId": "AROADBQP57FF2AEXAMPLE"
        }
    ],
    "IsTruncated": false
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListEntitiesForPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回已**arn:aws:iam::123456789012:policy/TestPolicy**附加政策的 IAM 群組、角色和使用者清單。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

輸出：

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListEntitiesForPolicy式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListGroupPolicies**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListGroupPolicies。

CLI

AWS CLI

列出附加至指定群組的所有內嵌政策

下列`list-group-policies`命令會列出附加至目前帳戶`Admins`中指定之 IAM 群組的內嵌政策名稱。

```
aws iam list-group-policies \
    --group-name Admins
```

輸出：

```
{
    "PolicyNames": [
        "AdminRoot",
        "ExamplePolicy"
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListGroup政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回內嵌在群組中的內嵌原則清單**Testers**。若要取得附加至群組的受管理原則，請使用命令**Get-IAMAttachedGroupPolicyList**。

```
Get-IAMGroupPolicyList -GroupName Testers
```

輸出：

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListGroup原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListGroups配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListGroups。

.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。

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListGroups中的。

CLI

AWS CLI

列出目前帳戶的 IAM 群組

下列 `list-groups` 命令會列出目前帳戶中的 IAM 群組。

```
aws iam list-groups
```

輸出：

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-06-04T20:27:27.972Z",
            "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "GroupName": "Admins"
        },
        {
            "Path": "/",
            "CreateDate": "2013-04-16T20:30:42Z",
            "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
            "GroupName": "S3-Admins"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListGroups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。

```
// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group
 actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(maxGroups int32) ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(context.TODO(),
 &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
  groups = result.Groups
 }
 return groups, err
}
```

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

列出群組。

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
```

```
    }
  }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListGroups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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListGroups中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例會傳回目前所定義之所有 IAM 群組的集合 AWS 帳戶。

```
Get-IAMGroupList
```

輸出：

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListGroups式參考中的。

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。

```
def list_groups(count):
```

```
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListGroups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 設定和執行程式碼範例儲存庫。

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
```

```
      @logger.info("\t#{group.group_name}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListGroups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。

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListGroups中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func listGroups() async throws -> [String] {
    var groupList: [String] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListGroupsInput(marker: marker)
        let output = try await client.listGroups(input: input)

        guard let groups = output.groups else {
            return groupList
        }

        for group in groups {
            if let name = group.groupName {
                groupList.append(name)
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return groupList
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListGroups中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListGroupsForUser`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListGroupsForUser`。

CLI

AWS CLI

列出 IAM 使用者所屬的群組

下列`list-groups-for-user`命令會顯示 IAM 使用者Bob所屬的群組。

```
aws iam list-groups-for-user \
    --user-name Bob
```

輸出：

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:18:08Z",
            "GroupId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admin",
            "GroupName": "Admin"
        },
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:37:28Z",
            "GroupId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/s3-Users",
            "GroupName": "s3-Users"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListGroupsForUser中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回 IAM 使用者**David**所屬的 IAM 群組清單。

```
Get-IAMGroupForUser -UserName David
```

輸出：

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListGroupsForUser式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListInstanceProfiles**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListInstanceProfiles。

CLI

AWS CLI

列出帳戶的執行個體設定檔

下列list-instance-profiles指令會列出與目前帳戶相關聯的執行個體設定檔。

```
aws iam list-instance-profiles
```

輸出：

```
{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "example-dev-role",
            "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
            "CreateDate": "2023-09-21T18:17:41+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-dev-role",
                    "RoleId": "AROAJ52OTH4H7LEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
                    "CreateDate": "2023-09-21T18:17:40+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
        {
```

```
            "Path": "/",
            "InstanceProfileName": "example-s3-role",
            "InstanceProfileId": "AIPAJVJVNRIQFREXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
            "CreateDate": "2023-09-21T18:18:50+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-s3-role",
                    "RoleId": "AROAINUBC5O7XLEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
                    "CreateDate": "2023-09-21T18:18:49+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version": "2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用執行個體設定檔。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListInstance設定檔。

PowerShell

用於的工具 PowerShell

範例 1：此範例會傳回目前定義的執行個體設定檔集合 AWS 帳戶。

```
Get-IAMInstanceProfileList
```

輸出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListInstance設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListInstanceProfilesForRole**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

列出 IAM 角色的執行個體設定檔

下列`list-instance-profiles-for-role`命令會列出與角色相關聯的執行個體設定檔`Test-Role`。

```
aws iam list-instance-profiles-for-role \
    --role-name Test-Role
```

輸出：

```
{
    "InstanceProfiles": [
        {
            "InstanceProfileId": "AIDGPMS9RO4H3FEXAMPLE",
            "Roles": [
                {
                    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                    "RoleId": "AIDACKCEVSQ6C2EXAMPLE",
                    "CreateDate": "2013-06-07T20:42:15Z",
```

```
                    "RoleName": "Test-Role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:role/Test-Role"
                }
            ],
            "CreateDate": "2013-06-07T21:05:24Z",
            "InstanceProfileName": "ExampleInstanceProfile",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:instance-profile/
ExampleInstanceProfile"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的使用執行個體設定檔。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListInstanceProfilesFor角色。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回與角色相關聯之執行個體設定檔的詳細資訊**ec2instancerole**。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

輸出：

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListInstanceProfilesFor角色。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListMfaDevices**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListMfaDevices。

CLI

AWS CLI

列出指定使用者的所有 MFA 裝置

此範例會傳回指派給 IAM 使用者Bob之 MFA 裝置的詳細資料。

```
aws iam list-mfa-devices \
    --user-name Bob
```

輸出：

```
{
    "MFADevices": [
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
            "EnableDate": "2019-10-28T20:37:09+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "GAKT12345678",
            "EnableDate": "2023-02-18T21:44:42+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
            "EnableDate": "2023-09-19T02:25:35+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
            "EnableDate": "2023-09-19T01:49:18+00:00"
        }
    ]
}
```

如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListMfa裝置。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回指派給 IAM 使用者**David**之 MFA 裝置的相關詳細資料。在此範例中，您可以判斷它是虛擬裝置，因為它**SerialNumber**是 ARN，而不是實體裝置的實際序號。

```
Get-IAMMFADevice -UserName David
```

輸出：

```
EnableDate                SerialNumber                              UserName
----------                ------------                              --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David       David
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListMfa裝置。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListOpenIdConnectProviders**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

列出有關帳戶中 OpenID Connect 提供商的 AWS 信息

此範例會傳回目前 AWS 帳戶中定義之所有 OpenID Connect 提供者的 ARNS 清單。

```
aws iam list-open-id-connect-providers
```

輸出：

```
{
```

```
    "OpenIDConnectProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com"
        }
    ]
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的建立 OpenID Connect (OIDC) 身分識別提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListOpenIdConnect提供者。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回目 AWS 帳戶前定義之所有 OpenID Connect 提供者的 ARNS 清單。

```
Get-IAMOpenIDConnectProviderList
```

輸出：

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListOpenIdConnect提供者。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListPolicies**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListPolicies。

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：

- 管理政策

.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。

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListPolicies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。

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(64) << "Description" << std::setw(12) <<
                      "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                      policy.GetPolicyName() << std::setw(30) <<
                      policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                      std::setw(64) << policy.GetDescription() << std::setw(12)
 <<
                      policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
```

```
                        std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ListPolicies中的。

CLI

AWS CLI

列出您 AWS 帳戶可用的受管理策略

此範例會傳回目前 AWS 帳戶中可用的前兩個受管理策略的集合。

```
aws iam list-policies \
    --max-items 3
```

輸出：

```
{
    "Policies": [
        {
            "PolicyName": "AWSCloudTrailAccessPolicy",
            "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 0,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2019-09-04T17:43:42+00:00",
```

```
            "UpdateDate": "2019-09-04T17:43:42+00:00"
        },
        {
            "PolicyName": "AdministratorAccess",
            "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
            "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 6,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:46+00:00",
            "UpdateDate": "2015-02-06T18:39:46+00:00"
        },
        {
            "PolicyName": "PowerUserAccess",
            "PolicyId": "ANPAJYRXTHIB4FOVS3ZXS",
            "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",
            "Path": "/",
            "DefaultVersionId": "v5",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:47+00:00",
            "UpdateDate": "2023-07-06T22:04:00+00:00"
        }
    ],
    "NextToken": "EXAMPLErZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiA4fQ=="
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListPolicies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。

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
 error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(context.TODO(),
 &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}
```

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

列出政策。

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services
 account.
    Scope: "Local",
  });

  let response = await client.send(command);

  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListPolicies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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListPolicies中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回目前 AWS 帳戶中可用的前三個受管理策略的集合。由-scope於未指定，因此預設為all並包含 AWS 受管政策和客戶管理的策略。

```
Get-IAMPolicyList -MaxItem 3
```

輸出：

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
```

```
UpdateDate       : 2/11/2015 9:21:45 AM
```

範例 2：此範例會傳回目前 AWS 帳戶中可用的前兩個客戶管理策略的集合。它用`-Scope local`來將輸出限制為只有客戶管理的策略。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

輸出：

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListPolicies式參考中的。

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。

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListPolicies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 設定和執行程式碼範例儲存庫。

此範例模組會列出、建立、附加和解除連結角色原則。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListPolicies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。

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;
```

```
    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListPolicies中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func listPolicies() async throws -> [MyPolicyRecord] {
        var policyList: [MyPolicyRecord] = []
        var marker: String? = nil
        var isTruncated: Bool

        repeat {
            let input = ListPoliciesInput(marker: marker)
```

```
            let output = try await client.listPolicies(input: input)

            guard let policies = output.policies else {
                return policyList
            }

            for policy in policies {
                guard   let name = policy.policyName,
                        let id = policy.policyId,
                        let arn = policy.arn else {
                    throw ServiceHandlerError.noSuchPolicy
                }
                policyList.append(MyPolicyRecord(name: name, id: id, arn: arn))
            }
            marker = output.marker
            isTruncated = output.isTruncated
        } while isTruncated == true
        return policyList
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListPolicies中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListPolicyVersions**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListPolicyVersions`。

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：

- 管理政策
- 復原政策版本

CLI

AWS CLI

列出指定受管理策略版本的相關資訊

此範例會傳回其 ARN 為arn:aws:iam::123456789012:policy/MySamplePolicy原則的可用版本清單。

```
aws iam list-policy-versions \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

輸出：

```
{
    "IsTruncated": false,
    "Versions": [
        {
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2015-06-02T23:19:44Z"
        },
        {
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2015-06-02T22:30:47Z"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListPolicy版本。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回其 ARN 為arn:aws:iam::123456789012:policy/MyManagedPolicy原則的可用版本清單。若要取得特定版本的原則文件，請使用指Get-IAMPolicyVersion令並指定所需版本VersionId的原則文件。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

輸出：

```
CreateDate                  Document                IsDefaultVersion
       VersionId
----------                  --------                ---------------
       ---------
2/12/2015 9:39:53 AM                                True
       v2
2/12/2015 9:39:09 AM                                False
       v1
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListPolicy版本。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListRolePolicies配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListRolePolicies。

.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。

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();
```

```
        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的ListRole政策。

CLI

AWS CLI

列出連接至 IAM 角色的政策

下列 `list-role-policies` 命令會列出指定 IAM 角色的許可政策名稱。

```
aws iam list-role-policies \
    --role-name Test-Role
```

輸出：

```
{
    "PolicyNames": [
        "ExamplePolicy"
    ]
}
```

若要查看連接至角色的信任政策，請使用 `get-role` 命令。若要查看許可政策的詳細資訊，請使用 `get-role-policy` 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListRole政策。

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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

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

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。

列出政策。

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
```

```
        }),
      );
    } else {
      break;
    }
  }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的ListRole政策。

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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient-
>listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考中的ListRole政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會傳回 IAM 角色中內嵌的內嵌政策名稱清單**lamda_exec_role**。若要查看內嵌政策的詳細資料，請使用命令**Get-IAMRolePolicy**。

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

輸出：

```
oneClick_lambda_exec_role_policy
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListRole原則。

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。

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件中的ListRole政策 (Boto3) 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 設定和執行程式碼範例儲存庫。

```
# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)
  response = @iam_client.list_attached_role_policies(role_name: role_name)
  response.attached_policies.map(&:policy_arn)
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的ListRole政策。

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。

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的ListRole政策以取得 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func listRolePolicies(role: String) async throws -> [String] {
        var policyList: [String] = []
        var marker: String? = nil
        var isTruncated: Bool
```

```
        repeat {
            let input = ListRolePoliciesInput(
                marker: marker,
                roleName: role
            )
            let output = try await client.listRolePolicies(input: input)

            guard let policies = output.policyNames else {
                return policyList
            }

            for policy in policies {
                policyList.append(policy)
            }
            marker = output.marker
            isTruncated = output.isTruncated
        } while isTruncated == true
        return policyList
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的ListRole政策以取得 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListRoleTags**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListRoleTags`。

CLI

AWS CLI

若要列出附加至角色的標籤

下列`list-role-tags`命令會擷取與指定角色相關聯的標籤清單。

```
aws iam list-role-tags \
    --role-name production-role
```

輸出：

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的標記AWS IAM 資源。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListRole標籤。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取與角色相關聯的標籤。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListRole標籤。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListRoles**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListRoles。

.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。

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListRoles中的。

CLI

AWS CLI

列出目前帳戶的 IAM 角色

下列 `list-roles` 命令會列出目前帳戶的 IAM 角色。

```
aws iam list-roles
```

輸出：

```
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "ExampleRole",
            "RoleId": "AROAJ52OTH4H7LEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
            "CreateDate": "2017-09-12T19:23:36+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        },
        {
            "Path": "/example_path/",
            "RoleName": "ExampleRoleWithPath",
            "RoleId": "AROAI4QRP7UFT7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
            "CreateDate": "2023-09-21T20:29:38+00:00",
            "AssumeRolePolicyDocument": {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
```

```
            },
            "MaxSessionDuration": 3600
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListRoles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。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(context.TODO(),
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
```

```
}
```

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

列出角色。

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
```

```
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考ListRoles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。

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
```

```
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListRoles中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例擷取中所有 IAM 角色的清單 AWS 帳戶。

```
Get-IAMRoleList
```

範例 2：此範例程式碼片段會擷取 AWS 帳戶中的 IAM 角色清單，並一次顯示三個角色，並等待您在每個群組之間按 Enter。它傳遞從上一個調用的**Marker**值，以指定下一個組應該從哪裡開始。

```
$nextMarker = $null
Do
{
    $results = Get-IAMRoleList -MaxItem 3 -Marker $nextMarker
    $nextMarker = $AWSHistory.LastServiceResponse.Marker
    $results
    Read-Host
} while ($nextMarker -ne $null)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListRoles式參考中的。

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。

```
def list_roles(count):
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListRoles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 設定和執行程式碼範例儲存庫。

```
  # Lists IAM roles up to a specified count.
  # @param count [Integer] the maximum number of roles to list.
  # @return [Array<String>] the names of the roles.
  def list_roles(count)
    role_names = []
    roles_counted = 0

    @iam_client.list_roles.each_page do |page|
      page.roles.each do |role|
        break if roles_counted >= count
        @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
        role_names << role.role_name
        roles_counted += 1
      end
      break if roles_counted >= count
    end

    role_names
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't list roles for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListRoles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。

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
```

```
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListRoles中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
    public func listRoles() async throws -> [String] {
        var roleList: [String] = []
        var marker: String? = nil
        var isTruncated: Bool

        repeat {
            let input = ListRolesInput(marker: marker)
            let output = try await client.listRoles(input: input)

            guard let roles = output.roles else {
                return roleList
```

```
            }

            for role in roles {
                if let name = role.roleName {
                    roleList.append(name)
                }
            }
            marker = output.marker
            isTruncated = output.isTruncated
        } while isTruncated == true
        return roleList
    }
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListRoles中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListSAMLProviders**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListSAMLProviders`。

.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。

```
    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
```

```
        return response.SAMLProviderList;
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的 ListSAMLProviders。

CLI

AWS CLI

列出帳戶中的 SAML 提供者 AWS

此範例會擷取在目前 AWS 帳戶中建立的 SAML 2.0 提供者清單。

```
aws iam list-saml-providers
```

輸出：

```
{
    "SAMLProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",
            "ValidUntil": "2015-06-05T22:45:14Z",
            "CreateDate": "2015-06-05T22:45:14Z"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM SAML 身分提供者。

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的 ListSAMLProviders。

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。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
 error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

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

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。

列出 SAML 供應商。

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的 ListSAMLProviders。

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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- 如需 API 詳細資訊，請參閱《AWS SDK for PHP API 參考》中的 ListSAMLProviders。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取在目前 AWS 帳戶建立的 SAML 2.0 提供者清單。它會傳回每個 SAML 提供者的 ARN、建立日期和到期日。

```
Get-IAMSAMLProviderList
```

輸出：

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- 如需 API 詳細資訊，請參閱指令程式參考中AWS Tools for PowerShell 的清單 SAML 提供者。

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。

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
```

```
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 ListSAMLProviders。

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。

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
```

```
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Ruby API 參考》中的 ListSAMLProviders。

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。

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Rust API 參考》中的 ListSAMLProviders。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListServerCertificates`配 AWS 開發套件或 CLI 使用

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

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。

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
                      certificate.GetServerCertificateId() << std::setw(80) <<
```

```
                        certificate.GetArn() << std::setw(14) <<

 certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                        std::setw(14) <<

 certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                        std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的ListServer憑證。

CLI

AWS CLI

列出您 AWS 帳戶中的伺服器憑證

下列`list-server-certificates`命令列出所有儲存並可在您的 AWS 帳戶中使用的伺服器憑證。

```
aws iam list-server-certificates
```

輸出：

```
{
    "ServerCertificateMetadataList": [
        {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
```

```
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        {
            "Path": "/cloudfront/",
            "ServerCertificateName": "MyTestCert",
            "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyTestCert",
            "UploadDate": "2015-04-21T18:14:16+00:00",
            "Expiration": "2018-01-14T17:52:36+00:00"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的在 IAM 中管理伺服器憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListServer憑證。

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。

列出憑證。

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
```

```
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
    } else {
      break;
    }
  }
}
```

- 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ListServer憑證。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Success", data);
  }
});
```

- 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ListServer憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取已上傳至目前伺服器憑證的清單 AWS 帳戶。

```
Get-IAMServerCertificateList
```

輸出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListServer憑證。

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。

列出、更新和刪除伺服器憑證。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
```

```
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的ListServer憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListSigningCertificates`配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

列出 IAM 使用者的簽署憑證

下列`list-signing-certificates`命令列出名為 IAM 使用者的簽署憑證Bob。

```
aws iam list-signing-certificates \
```

```
    --user-name Bob
```

輸出：

```
{
    "Certificates": [
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-
body>-----END CERTIFICATE-----",
            "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
            "UploadDate": "2013-06-06T21:40:08Z"
        }
    ]
}
```

如需詳細資訊，請參閱 Amazon EC2 使用者指南中的管理簽署憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListSigning憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取與具名使用者相關聯之簽署憑證的詳細資料**Bob**。

```
Get-IAMSigningCertificate -UserName Bob
```

輸出：

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
```

```
 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListSigning憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListUserPolicies**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListUserPolicies`。

CLI

AWS CLI

列出 IAM 使用者的政策

下列 `list-user-policies` 命令會列出連接至名為 Bob 之 IAM 使用者的政策。

```
aws iam list-user-policies \
    --user-name Bob
```

輸出：

```
{
    "PolicyNames": [
        "ExamplePolicy",
        "TestPolicy"
    ]
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的在 AWS 帳戶中建立AWS IAM 使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListUser政策。

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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
```

```
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取內嵌在名為 IAM 使用者的內嵌政策名稱清單**David**。

```
Get-IAMUserPolicyList -UserName David
```

輸出：

```
Davids_IAM_Admin_Policy
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListUser原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListUserTags**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListUserTags`。

CLI

AWS CLI

列出附加至使用者的標籤

下列`list-user-tags`命令會擷取與指定 IAM 使用者相關聯的標籤清單。

```
aws iam list-user-tags \
    --user-name alice
```

輸出：

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

如需詳細資訊，請參閱 IAM 使用者指南中的標記AWS IAM 資源。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ListUser標籤。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取與使用者相關聯的標籤。

```
Get-IAMUserTagList -UserName joe
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ListUser標籤。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`ListUsers`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`ListUsers`。

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：

- 建立唯讀和讀寫的使用者

.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。

```
    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListUsers中的。

Bash

AWS CLI 與 Bash 腳本

> **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。

```
########################################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
########################################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

########################################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
########################################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
    echo "Lists the AWS Identity and Access Management (IAM) user in the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListUsers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。

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
```

```
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考ListUsers中的。

CLI

AWS CLI

列出 IAM 使用者

下列 `list-users` 命令會列出目前帳戶中的 IAM 使用者。

```
aws iam list-users
```

輸出：

```
{
    "Users": [
        {
            "UserName": "Adele",
            "Path": "/",
            "CreateDate": "2013-03-07T05:14:48Z",
            "UserId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Adele"
        },
        {
            "UserName": "Bob",
            "Path": "/",
            "CreateDate": "2012-09-21T23:03:13Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        }
    ]
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的列出 IAM 使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListUsers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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Go API 參考ListUsers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.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;

/**
 * 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 ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
```

```
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request =
 ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException 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ListUsers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。

列出使用者。

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);
  response.Users?.forEach(({ UserName, CreateDate }) => {
    console.log(`${UserName} created on: ${CreateDate}`);
  });
  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ListUsers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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
  }
});
```

- 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ListUsers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 listAllUsers() {
    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
```

```
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListUsers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。

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for PHP API 參考ListUsers中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取目前的使用者集合 AWS 帳戶。

```
Get-IAMUserList
```

輸出：

```
Arn              : arn:aws:iam::123456789012:user/Administrator
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE1
UserName         : Administrator

Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/6/2015 12:54:42 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : L3EWNONDOM3YUEXAMPLE2
UserName         : bab

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListUsers式參考中的。

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。

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListUsers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 設定和執行程式碼範例儲存庫。

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing users: #{e.message}")
    []
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考ListUsers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。

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK ListUsers中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func listUsers() async throws -> [MyUserRecord] {
    var userList: [MyUserRecord] = []
    var marker: String? = nil
    var isTruncated: Bool

    repeat {
        let input = ListUsersInput(marker: marker)
        let output = try await client.listUsers(input: input)

        guard let users = output.users else {
            return userList
        }

        for user in users {
            if let id = user.userId, let name = user.userName {
                userList.append(MyUserRecord(id: id, name: name))
            }
        }
        marker = output.marker
        isTruncated = output.isTruncated
    } while isTruncated == true
    return userList
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListUsers中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ListVirtualMfaDevices**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要列出虛擬 MFA 裝置

下列`list-virtual-mfa-devices`命令列出已針對目前帳戶設定的虛擬 MFA 裝置。

```
aws iam list-virtual-mfa-devices
```

輸出：

```
{
    "VirtualMFADevices": [
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
        },
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
        }
    ]
}
```

如需詳細資訊，請參閱 AWS IAM 使用者指南中的啟用虛擬多重要素身份驗證 (MFA) 裝置。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考ListVirtualMfaDevices中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會擷取指派給 AWS 帳戶中使用者的虛擬 MFA 裝置集合。每個**User**屬性都是一個物件，其中包含指派裝置的 IAM 使用者詳細資訊。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

輸出：

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程ListVirtualMfaDevices式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutGroupPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutGroupPolicy。

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：

- 建立群組並新增使用者。

.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。

```
    /// <summary>
```

```
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的PutGroup政策。

CLI

AWS CLI

將政策新增至群組

下列 put-group-policy 命令會將政策新增至名為 Admins 的 IAM 群組。

```
aws iam put-group-policy \
    --group-name Admins \
    --policy-document file://AdminPolicy.json \
    --policy-name AdminRoot
```

此命令不會產生輸出。

此原則會定義為 AdminPolicy.json 檔案中的 JSON 文件。(檔案名稱和副檔名沒有意義。)

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的PutGroup政策。

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立名為的內嵌政策，**AppTesterPolicy**並將其內嵌在 IAM 群組**AppTesters**中。如果已存在具有相同名稱的內嵌政策，則會覆寫該原則。JSON 策略內容來自文件**apptesterpolicy.json**。請注意，您必須使用**-Raw**參數才能成功處理 JSON 檔案的內容。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的PutGroup原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**PutRolePermissionsBoundary**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

範例 1：將基於自訂政策的許可界限套用至 IAM 角色

下列put-role-permissions-boundary範例會套用名intern-boundary為指定 IAM 角色權限界限的自訂政策。

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --role-name lambda-application-role
```

此命令不會產生輸出。

範例 2：將以 AWS 受管政策為基礎的許可界限套用至 IAM 角色

下列put-role-permissions-boundary範例會將受 AWS 管PowerUserAccess政策套用為指定 IAM 角色的許可界限。

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --role-name x-account-admin
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutRolePermissionsBoundary中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例顯示如何設定 IAM 角色的權限界限。您可以將 AWS 受管理的原則或自訂原則設定為權限界限。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutRolePermissionsBoundary式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutRolePolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutRolePolicy。

.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。

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考中的PutRole政策。

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。

```
bool AwsDoc::IAM::putRolePolicy(
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的PutRole政策。

CLI

AWS CLI

將許可政策連接至 IAM 角色

下列 put-role-policy 命令會將許可政策連接到名為 Test-Role 的角色。

```
aws iam put-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

此命令不會產生輸出。

此原則會定義為 AdminPolicy.json 檔案中的 JSON 文件。(檔案名稱和副檔名沒有意義。)

若要將信任政策連接至角色，請使用 update-assume-role-policy 命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的PutRole政策。

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。

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
      Action: [
```

```
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::some-test-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
});

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的PutRole政策。

PowerShell

用於的工具 PowerShell

範例 1：此範例會建立名為的內嵌政策，**FedTesterRolePolicy**並將其嵌入 IAM 角色**FedTesterRole**中。如果已存在具有相同名稱的內嵌政策，則會覆寫該原則。JSON 政策內容來自檔案**FedTesterPolicy.json**。請注意，您必須使用**-Raw**參數才能成功處理 JSON 檔案的內容。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的PutRole原則。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**PutUserPermissionsBoundary**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

範例 1：將基於自訂政策的許可界限套用至 IAM 使用者

下列put-user-permissions-boundary範例會套用名intern-boundary為指定 IAM 使用者權限界限的自訂政策。

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --user-name intern
```

此命令不會產生輸出。

範例 2：將基於 AWS 受管政策的許可界限套用至 IAM 使用者

下列put-user-permissions-boundary範例會套用名PowerUserAccess為指定 IAM 使用者權限界限的 AWS 受管政策。

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --user-name developer
```

此命令不會產生輸出。

如需詳細資訊，請參閱《IAM 使用者指南》AWS 中的新增和移除 IAM 身分許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考PutUserPermissionsBoundary中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例顯示如何設定使用者的權限界限。您可以將 AWS 受管理的原則或自訂原則設定為權限界限。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程PutUserPermissionsBoundary式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭PutUserPolicy配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用PutUserPolicy。

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：

- 建立使用者並擔任角色

CLI

AWS CLI

將政策連接至 IAM 使用者

下列 put-user-policy 命令會將政策連接至名為 Bob 的 IAM 使用者。

```
aws iam put-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

此命令不會產生輸出。

此原則會定義為 AdminPolicy.json 檔案中的 JSON 文件。(檔案名稱和副檔名沒有意義。)

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的新增和移除 IAM 身分許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的PutUser政策。

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。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
```

```
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
 actions []string,
  roleArn string) error {
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
 &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會建立名為的內嵌政策，**EC2AccessPolicy**並將其內嵌在 IAM 使用者**Bob**中。如果已存在具有相同名稱的內嵌政策，則會覆寫該原則。JSON 政策內容來自檔案**EC2AccessPolicy.json**。請注意，您必須使用**-Raw**參數才能成功處理 JSON 檔案的內容。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的PutUser原則。

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。

```
  # Creates an inline policy for a specified user.
  # @param username [String] The name of the IAM user.
  # @param policy_name [String] The name of the policy to create.
  # @param policy_document [String] The JSON policy document.
  # @return [Boolean]
  def create_user_policy(username, policy_name, policy_document)
    @iam_client.put_user_policy({
      user_name: username,
      policy_name: policy_name,
      policy_document: policy_document
    })
    @logger.info("Policy #{policy_name} created for user #{username}.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
```

```
    @logger.error("\t#{e.code}: #{e.message}")
    false
  end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的PutUser政策。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
   func putUserPolicy(policyDocument: String, policyName: String, user:
IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            throw error
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的PutUser政策以取得 Swift 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭RemoveClientIdFromOpenIdConnectProvider配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要從為指定 IAM OpenID Connect 提供者註冊的用戶端 ID 清單中移除指定的用戶端 ID

此範例會My-TestApp-3從與 ARN 的 IAM OIDC 提供者相關聯的用戶端 ID 清單中移除用戶端 ID。arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com

```
aws iam remove-client-id-from-open-id-connect-provider
    --client-id My-TestApp-3 \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的建立 OpenID Connect (OIDC) 身分識別提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考RemoveClientIdFromOpenIdConnectProvider中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例會**My-TestApp-3**從與 ARN 為 IAM OIDC 提供者相關聯的用戶端 ID 清單中移除用戶端 ID。**arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com**

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程[RemoveClientIdFromOpenIdConnectProvider](#)式參考中的。

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 IAM](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**RemoveRoleFromInstanceProfile**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要從執行個體設定檔移除角色

下列`remove-role-from-instance-profile`命令會從名為的執行個體設定檔`Test-Role`中移除名為的角色`ExampleInstanceProfile`。

```
aws iam remove-role-from-instance-profile \
    --instance-profile-name ExampleInstanceProfile \
    --role-name Test-Role
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的[使用執行個體設定檔](#)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的[RemoveRoleFromInstance設定檔](#)。

PowerShell

適用的工具 PowerShell

範例 1：此範例會刪除名為的 EC2 執行個體設定檔**MyNewRole**中指定的角色**MyNewRole**。在 IAM 主控台中建立的執行個體設定檔永遠與角色具有相同的名稱，如本範例所示。如果您在 API 或 CLI 中創建它們，則它們可以具有不同的名稱。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的[RemoveRoleFromInstance設定檔](#)。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭RemoveUserFromGroup配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用RemoveUserFromGroup。

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：

- 建立群組並新增使用者。

.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。

```
    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
```

```
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考RemoveUserFromGroup中的。

CLI

AWS CLI

從 IAM 群組移除使用者

下列 `remove-user-from-group` 命令會將名為 Bob 的使用者從名為 Admins 的 IAM 群組中移除。

```
aws iam remove-user-from-group \
    --user-name Bob \
    --group-name Admins
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的在 IAM 使用者群組中新增和移除使用者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考RemoveUserFromGroup中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會從群組**Bob**中移除 IAM 使用者**Testers**。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

範例 2：此範例會尋找 IAM 使用者**Theresa**為其成員的任何群組，然後**Theresa**從這些群組中移除。

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

範例 3：此範例顯示**Bob**從**Testers**群組移除 IAM 使用者的另一種方法。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程RemoveUserFromGroup式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**ResyncMfaDevice**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ResyncMfaDevice。

CLI

AWS CLI

若要同步化 MFA 裝置

下列resync-mfa-device範例會同步化與 IAM 使用者相關聯Bob且其 ARN 所在的 MFA 裝置arn:aws:iam::123456789012:mfa/BobsMFADevice與提供兩個驗證碼的驗證器程式。

```
aws iam resync-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 987654
```

此命令不會產生輸出。

如需詳細資訊，請參閱《IAM 使用者指南》中的在 AWS中使用多重要素驗證 (MFA)。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的ResyncMfa裝置。

PowerShell

適用的工具 PowerShell

範例 1：此範例會同步化與 IAM 使用者相關聯**Bob**且其 ARN 所在的 MFA 裝置**arn:aws:iam::123456789012:mfa/bob**與提供兩個驗證碼的驗證器程式。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

範例 2：此範例會將與 IAM 使用者**Theresa**關聯的 IAM MFA 裝置與具有序號**ABCD12345678**且提供兩個驗證碼的實體裝置同步處理。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的ResyncMfa裝置。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**SetDefaultPolicyVersion**配 AWS 開發套件或 CLI 使用

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

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：

- 管理政策
- 復原政策版本

CLI

AWS CLI

將指定策略的指定版本設定為策略的預設版本。

此範例會將 ARN arn:aws:iam::123456789012:policy/MyPolicy 為預設作用中v2版本的原則版本設定。

```
aws iam set-default-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的 IAM 中的政策和許可。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考SetDefaultPolicyVersion中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將 ARN 為預設**arn:aws:iam::123456789012:policy/MyPolicy**作用中**v2**版本的原則版本設定。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程SetDefaultPolicyVersion式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**TagRole**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用TagRole。

CLI

AWS CLI

若要將標籤新增至角色

下列tag-role命令會將具有部門名稱的標籤新增至指定的角色。

```
aws iam tag-role --role-name my-role \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

此命令不會產生輸出。

如需詳細資訊，請參閱 IAM 使用者指南中的標記AWS IAM 資源。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考TagRole中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例將標籤新增至「識別管理服務」中的角色

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程TagRole式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭TagUser配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用TagUser。

CLI

AWS CLI

若要將標籤新增至使用者

下列tag-user命令會將含有關聯部門的標籤新增至指定的使用者。

```
aws iam tag-user \
    --user-name alice \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

此命令不會產生輸出。

如需詳細資訊，請參閱 IAM 使用者指南中的標記AWS IAM 資源。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考TagUser中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例將標籤新增至「識別管理服務」中的使用者

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程TagUser式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UntagRole配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UntagRole。

CLI

AWS CLI

若要從角色中移除標籤

下列untag-role命令會從指定的角色中移除具有索引鍵名稱為「Develop」的任何標籤。

```
aws iam untag-role \
    --role-name my-role \
    --tag-keys Department
```

此命令不會產生輸出。

如需詳細資訊，請參閱 IAM 使用者指南中的標記AWS IAM 資源。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UntagRole中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會從名為「MyRoleName」的角色中移除標籤，並使用標籤索引鍵為「abac」。若要移除多個標籤，請提供以逗號分隔的標籤關鍵字清單。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UntagRole式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UntagUser配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UntagUser。

CLI

AWS CLI

若要從使用者移除標籤

下列`untag-user`命令會從指定的使用者中移除任何含有索引鍵名稱為「Develop」的標籤。

```
aws iam untag-user \
    --user-name alice \
    --tag-keys Department
```

此命令不會產生輸出。

如需詳細資訊，請參閱 IAM 使用者指南中的標記AWS IAM 資源。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UntagUser中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會從名為「joe」的使用者中移除標籤，標籤鍵為「abac」和「xyzw」的標籤。若要移除多個標籤，請提供以逗號分隔的標籤關鍵字清單。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UntagUser式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateAccessKey**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`UpdateAccessKey`。

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：

- 管理存取金鑰

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。

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  Aws::IAM::Model::StatusType status,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  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UpdateAccess金鑰。

CLI

AWS CLI

啟用或停用 IAM 使用者的存取金鑰

下列 update-access-key 命令會為名為 Bob 的 IAM 使用者停用指定的存取金鑰 (存取金鑰 ID 與私密存取金鑰)。

```
aws iam update-access-key \
    --access-key-id AKIAIOSFODNN7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

此命令不會產生輸出。

停用金鑰表示它無法用於以程式設計方式存取。 AWS但是，金鑰仍然可用，並且可以重新啟用。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。

- 如需 API 詳細資訊，請參閱UpdateAccess輸入AWS CLI 命令參考。

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.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * 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 UpdateAccessKey {

    private static StatusType statusType;

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

                Usage:
                    <username> <accessId> <status>\s

                Where:
                    username - The name of the user whose key you want to update.
\s
                    accessId - The access key ID of the secret access key you
 want to update.\s
                    status - The status you want to assign to the secret access
 key.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }
```

```
    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
            System.out.printf("Successfully updated the status of access key %s
 to" +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException 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UpdateAccess金鑰。

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。

更新存取金鑰。

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- 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UpdateAccess金鑰。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });
```

```
// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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UpdateAccess金鑰。

PowerShell

用於的工具 PowerShell

範例 1：此範例會將名AKIAIOSFODNN7EXAMPLE為的 IAM 使用者的存取金鑰狀態變更Bob為Inactive。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UpdateAccess金鑰。

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。

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise
```

- 如需 API 詳細資訊，請參閱AWS 開發套件中的UpdateAccess金鑰 (Boto3) 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UpdateAccountPasswordPolicy配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要設定或變更目前的帳號密碼策略

下列update-account-password-policy命令會將密碼原則設定為最少要求八個字元的長度，並且需要密碼中的一或多個數字。

```
aws iam update-account-password-policy \
    --minimum-password-length 8 \
    --require-numbers
```

此命令不會產生輸出。

帳戶密碼政策的變更會影響帳戶中為 IAM 使用者建立的任何新密碼。密碼策略變更不會影響現有密碼。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的設定 IAM 使用者的帳戶密碼政策。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateAccountPasswordPolicy中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會使用指定的設定更新帳戶的密碼策略。請注意，未包含在指令中的任何參數都不會保持未修改。相反地，它們會重設為預設值。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程[UpdateAccountPasswordPolicy](#)式參考中的。

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 IAM](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateAssumeRolePolicy**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

更新 IAM 角色的信任政策

下列`update-assume-role-policy`命令會更新名為之角色的信任原則`Test-Role`。

```
aws iam update-assume-role-policy \
    --role-name Test-Role \
    --policy-document file://Test-Role-Trust-Policy.json
```

此命令不會產生輸出。

在 Test-Role-Trust-Policy.json 檔案中，將信任政策定義為 JSON 文件。(檔案名稱和副檔名沒有意義。) 信任政策必須指定主體。

若要更新角色的權限原則，請使用`put-role-policy`命令。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的[建立 IAM 角色](#)。

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

PowerShell

適用的工具 PowerShell

範例 1：此範例會更新以新信任政策命名**ClientRole**的 IAM 角色，其內容來自檔案**ClientRolePolicy.json**。請注意，您必須使用 **-Raw** switch 參數才能成功處理 JSON 檔案的內容。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UpdateAssumeRolePolicy式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateGroup**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateGroup。

CLI

AWS CLI

若要重新命名 IAM 群組

下列update-group命令會將 IAM 群組的名稱變更Test為Test-1。

```
aws iam update-group \
    --group-name Test \
    --new-group-name Test-1
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的重新命名 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateGroup中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例將 IAM 群組重新命名**Testers**為**AppTesters**。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

範例 2：此範例會將 IAM 群組的路徑變更**AppTesters**為**/Org1/Org2/**。這會將**arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**群組的 ARN 變更為。

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UpdateGroup式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UpdateLoginProfile配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateLoginProfile。

CLI

AWS CLI

更新 IAM 使用者的密碼

下列update-login-profile命令會為名為的 IAM 使用者建立新密碼Bob。

```
aws iam update-login-profile \
    --user-name Bob \
    --password <password>
```

此命令不會產生輸出。

若要設定帳號的密碼策略，請使用update-account-password-policy指令。如果新密碼違反了帳號密碼策略，則命令會傳回PasswordPolicyViolation錯誤。

如果帳戶密碼政策允許他們，IAM 使用者可以使用change-password命令變更自己的密碼。

將密碼存儲在安全的地方。如果密碼丟失，則無法恢復密碼，您必須使用該create-login-profile命令創建一個新密碼。

如需詳細資訊，請參閱 IAM 使用者指南中的管理AWS IAM 使用者的密碼。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UpdateLogin設定檔。

PowerShell

適用的工具 PowerShell

範例 1：此範例為 IAM 使用者設定新的臨時密碼**Bob**，並要求使用者在下次登入時變更密碼。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UpdateLogin設定檔。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateOpenIdConnectProviderThumbprint**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

使用新清單取代現有的伺服器憑證指紋清單

此範例會針對其 ARN arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com 將使用新指紋的 OIDC 提供者更新憑證指紋清單。

```
aws iam update-open-id-connect-provider-thumbprint \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com \
    --thumbprint-list 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

此命令不會產生輸出。

如需詳細資訊，請參閱 AWS IAM 使用者指南中的建立 OpenID Connect (OIDC) 身分識別提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateOpenIdConnectProviderThumbprint中的。

PowerShell

適用的工具 PowerShell

範例 1：此範例會針對其 ARN **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 將使用新指紋的 OIDC 提供者更新憑證指紋清單。當與提供者關聯的憑證變更時，OIDC 提供者會共用新值。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UpdateOpenIdConnectProviderThumbprint式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateRole**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateRole。

CLI

AWS CLI

變更 IAM 角色的說明或工作階段持續時間

下列update-role命令會將 IAM 角色的描述變更為，Main production role並production-role將工作階段持續時間上限設定為 12 小時。

```
aws iam update-role \
    --role-name production-role \
    --description 'Main production role' \
    --max-session-duration 43200
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateRole中的。

PowerShell

用於的工具 PowerShell

範例 1：此範例會更新角色說明和工作階段持續時間上限值 (以秒為單位)，以便請求角色的工作階段。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UpdateRole式參考中的。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UpdateRoleDescription配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateRoleDescription。

CLI

AWS CLI

若要變更 IAM 角色的說明

下列update-role命令會將 IAM 角色的描述變更production-role為Main production role。

```
aws iam update-role-description \
    --role-name production-role \
    --description 'Main production role'
```

輸出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "production-role",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/production-role",
        "CreateDate": "2017-12-06T17:16:37+00:00",
        "AssumeRolePolicyDocument": {
```

```
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole",
                    "Condition": {}
                }
            ]
        },
        "Description": "Main production role"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的修改角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UpdateRole說明。

PowerShell

適用的工具 PowerShell

範例 1：此範例會更新您帳戶中 IAM 角色的說明。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UpdateRole說明。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`UpdateSamlProvider`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`UpdateSamlProvider`。

CLI

AWS CLI

更新現有 SAML 提供者的中繼資料文件

此範例會使arn:aws:iam::123456789012:saml-provider/SAMLADFS用檔案中的新 SAML 中繼資料文件，更新其 ARN 所在 IAM 中的 SAML 提供者。SAMLMetaData.xml

```
aws iam update-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

輸出：

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的建立 IAM SAML 身分提供者。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UpdateSaml提供者。

PowerShell

用於的工具 PowerShell

範例 1：此範例會使**arn:aws:iam::123456789012:saml-provider/SAMLADFS**用檔案中的新 SAML 中繼資料文件更新 IAM 中的 SAML 提供者。**SAMLMetaData.xml**請注意，您必須使用 **-Raw** switch 參數才能成功處理 JSON 檔案的內容。

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UpdateSaml提供者。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateServerCertificate**配 AWS 開發套件或 CLI 使用

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

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。

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
 &currentCertificateName,
                                           const Aws::String &newCertificateName,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName <<
 ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }
```

```
    return result;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考中的UpdateServer憑證。

CLI

AWS CLI

變更 AWS 帳戶中伺服器憑證的路徑或名稱

下列 update-server-certificate 命令會將憑證名稱從 myServerCertificate 變更為 myUpdatedServerCertificate。它還將路徑更改為，以/cloudfront/便可以通過 Amazon CloudFront 服務訪問它。此命令不會產生輸出。您可以透過執行 list-server-certificates 命令來查看更新的結果。

```
aws-iam update-server-certificate \
    --server-certificate-name myServerCertificate \
    --new-server-certificate-name myUpdatedServerCertificate \
    --new-path /cloudfront/
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的在 IAM 中管理伺服器憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UpdateServer憑證。

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。

更新伺服器憑證。

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} currentName
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- 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UpdateServer憑證。

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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};
```

```
iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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UpdateServer憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例將名為的憑證重新命名`MyServerCertificate`為`MyRenamedServerCertificate`。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

範例 2：此範例會將名為的憑證移`MyServerCertificate`至 /Org1/Org2/路徑。這會將`arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate`資源的 ARN 變更為。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UpdateServer憑證。

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。

列出、更新和刪除伺服器憑證。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "ServerCertificateManager"
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key,
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info("No server certificates found.")
      return
```

```
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考中的UpdateServer憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateSigningCertificate**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

啟用或停用 IAM 使用者的簽署憑證

下列update-signing-certificate命令會停用名Bob為 IAM 使用者的指定簽署憑證。

```
aws iam update-signing-certificate \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

若要取得簽署憑證的 ID，請使用list-signing-certificates指令。

如需詳細資訊，請參閱 Amazon EC2 使用者指南中的管理簽署憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UpdateSigning憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會更新與名為的 IAM 使用者**Bob**及其憑證 ID si 相關聯的憑證，以**Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU**將其標記為非作用中。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UpdateSigning憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UpdateUser**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateUser。

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：

- 建立唯讀和讀寫的使用者

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。

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName
 <<
                  ":" << 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UpdateUser中的。

CLI

AWS CLI

變更 IAM 使用者的名稱

下列 update-user 命令會將 IAM 使用者名稱從 Bob 變更為 Robert。

```
aws iam update-user \
    --user-name Bob \
    --new-user-name Robert
```

此命令不會產生輸出。

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的重新命名 IAM 使用者群組。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考UpdateUser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.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * 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 UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

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

        String curName = args[0];
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String
 newName) {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
 newName);
```

```
        } catch (IamException 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UpdateUser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。

更新使用者。

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });

  return client.send(command);
};
```

- 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UpdateUser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。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 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UpdateUser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 updateIAMUser(
    curName: String?,
    newName: String?
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

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

PowerShell

適用的工具 PowerShell

範例 1：此範例將 IAM 使用者重新命名**Bob**為**Robert**。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

範例 2：此範例會將 IAM 使用者的路徑變更**Bob**為**/Org1/Org2/**，這會有效地將使用者的 ARN 變更為。**arn:aws:iam::123456789012:user/Org1/Org2/bob**

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程UpdateUser式參考中的。

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。

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件UpdateUser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 設定和執行程式碼範例儲存庫。

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to
'#{new_name}': #{e.message}")
  false
end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考UpdateUser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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UploadServerCertificate**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

將伺服器憑證上傳至您的 AWS 帳戶

下列上傳伺服器憑證命令會將伺服器憑證上傳至您的帳戶。 AWS 在此範例中，憑證位於檔案 `public_key_cert_file.pem` 中，相關聯的私密金鑰位於檔案 `my_private_key.pem`

中，而憑證授權機構 (CA) 提供的憑證鏈結位於 my_certificate_chain_file.pem 檔案中。文件上傳完成後，它可以在我的名稱下使用ServerCertificate。以 file:// 開頭的參數會告訴命令讀取檔案的內容，並將其用作參數值 (而不是檔案名稱本身)。

```
aws iam upload-server-certificate \
    --server-certificate-name myServerCertificate \
    --certificate-body file://public_key_cert_file.pem \
    --private-key file://my_private_key.pem \
    --certificate-chain file://my_certificate_chain_file.pem
```

輸出：

```
{
    "ServerCertificateMetadata": {
        "Path": "/",
        "ServerCertificateName": "myServerCertificate",
        "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
        "Arn": "arn:aws:iam::1234567989012:server-certificate/
myServerCertificate",
        "UploadDate": "2019-04-22T21:13:44+00:00",
        "Expiration": "2019-10-15T22:23:16+00:00"
    }
}
```

如需詳細資訊，請參閱《使用 IAM 指南》中的「建立、上傳和刪除伺服器憑證」。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UploadServer憑證。

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。

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "fs";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
```

```
import * as path from "path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command,
 or the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });
```

```
  return client.send(command);
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考中的UploadServer憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會將新的伺服器憑證上傳至 IAM 帳戶。包含憑證主體、私密金鑰和 (選擇性) 憑證鏈結的檔案必須全部採用 PEM 編碼。請注意，參數需要檔案的實際內容，而不是檔案名稱。您必須使用 **-Raw** switch 參數才能成功處理檔案內容。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

輸出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UploadServer憑證。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**UploadSigningCertificate**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

上傳 IAM 使用者的簽署憑證

下列upload-signing-certificate命令會為名為的 IAM 使用者上傳簽署憑證Bob。

```
aws iam upload-signing-certificate \
    --user-name Bob \
    --certificate-body file://certificate.pem
```

輸出：

```
{
    "Certificate": {
        "UserName": "Bob",
        "Status": "Active",
        "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
 CERTIFICATE-----",
        "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
        "UploadDate": "2013-06-06T21:40:08.121Z"
    }
}
```

憑證位於 PEM 格式的名為憑證 .pem 的檔案中。

如需詳細資訊，請參閱使用 IAM 指南中的建立和上傳使用者簽署憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的UploadSigning憑證。

PowerShell

適用的工具 PowerShell

範例 1：此範例會上傳新的 X.509 簽署憑證，並將其與名為**Bob**的 IAM 使用者產生關聯。包含憑證主體的檔案是 PEM 編碼的。**CertificateBody**參數需要憑證檔案的實際內容，而不是檔案名稱。您必須使用 **-Raw** switch 參數才能成功處理檔案。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

輸出：

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
```

```
 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                  YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                  21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                  rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                  nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的UploadSigning憑證。

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 IAM。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 軟體開發套件的 IAM 案例

下列程式碼範例說明如何使用 AWS SDK 在 IAM 中實作常見案例。這些案例會向您展示如何呼叫 IAM 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 SDK 建置及管理彈性服務
- 建立 IAM 群組，並使用 AWS SDK 將使用者新增至群組
- 使用 AWS SDK 建立 IAM AWS STS 使用者並擔任角色
- 使用 SDK 建立唯讀和讀寫 IAM 使用者 AWS
- 使用 AWS SDK 管理 IAM 存取金鑰
- 使用 AWS 開發套件管理 IAM 政策
- 使用 AWS SDK 管理 IAM 角色
- 使用 AWS SDK 管理您的 IAM 帳戶
- 使用 AWS SDK 復原 IAM 政策版本
- 使用 SDK 使用 IAM 政策產生器 AWS API

使用 AWS SDK 建置及管理彈性服務

下列程式碼範例示範如何建立負載平衡的 Web 服務，以傳回書籍、影片和歌曲建議。此範例顯示服務如何回應失故障，以及如何在發生故障時重組服務以提高復原能力。

- 使用 Amazon EC2 Auto Scaling 群組根據啟動範本建立 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體，並將執行個體數量保持在指定範圍內。
- 使用 Elastic Load Balancing 處理和分發 HTTP 請求。
- 監控 Auto Scaling 群組中執行個體的運作狀態，並且只將請求轉送給運作良好的執行個體。
- 在每個 EC2 執行個體上執行一個 Python Web 伺服器來處理 HTTP 請求。Web 伺服器會回應建議和運作狀態檢查。
- 使用 Amazon DynamoDB 資料表模擬建議服務。
- 透過更新 AWS Systems Manager 參數來控制 Web 伺服器對要求和健康狀態檢查的回應。

.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。

```
   static async Task Main(string[] args)
   {
       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally, load local settings.
           .Build();


       // Set up dependency injection for the AWS services.
       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<IAmazonIdentityManagementService>()
                   .AddAWSService<IAmazonDynamoDB>()
                   .AddAWSService<IAmazonElasticLoadBalancingV2>()
                   .AddAWSService<IAmazonSimpleSystemsManagement>()
                   .AddAWSService<IAmazonAutoScaling>()
                   .AddAWSService<IAmazonEC2>()
                   .AddTransient<AutoScalerWrapper>()
                   .AddTransient<ElasticLoadBalancerWrapper>()
                   .AddTransient<SmParameterWrapper>()
                   .AddTransient<Recommendations>()
                   .AddSingleton<IConfiguration>(_configuration)
           )
           .Build();

       ServicesSetup(host);
       ResourcesSetup();

       try
       {
           Console.WriteLine(new string('-', 80));
           Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
           Console.WriteLine(new string('-', 80));
```

```
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
```

```
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
```

```
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
```

```
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
_elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
protocol, port, defaultVpc.VpcId);

        await
_elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
subnetIds, targetGroup);
        await
_autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
```

```
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
            var sshPortIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
 ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
 your default VPC must\n"
                    + "allows access from this computer. You can either add it
 automatically from this\n"
                    + "example or add it yourself using the AWS Management
 Console.\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
 inbound traffic from your computer's IP address?"))
                {
                    await
 _autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
 ipString);
                }
            }

            if (!sshPortIsOpen)
```

```
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
```

```
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
                          "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");
```

```
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
```

```
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);
```

```
        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
```

```
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
_recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

建立包裝 Auto Scaling 和 Amazon EC2 動作的類別。

```
/// <summary>
```

```
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration)
    {
        _amazonAutoScaling = amazonAutoScaling;
```

```
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance.The role has attached policies that specify the AWS permissions
granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                    "\"Version\": \"2012-10-17\"," +
```

```
                                "\"Statement\": [{" +
                                    "\"Effect\": \"Allow\"," +
                                    "\"Principal\": {" +
                                    "\"Service\": [" +
                                        "\"ec2.amazonaws.com\"" +
                                    "]" +
                                    "}," +
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
```

```
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
```

```
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }
```

```
    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
    /// The launch template specifies a Bash script in its user data field that
runs after
    /// the instance is started. This script installs the Python packages and
starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
    /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
    {
        await CreateKeyPair(_keyPairName);
        await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName, _instanceProfileName, instancePolicyPath);

        var startServerText = await File.ReadAllTextAsync(startupScriptPath);
        var plainTextBytes = System.Text.Encoding.UTF8.GetBytes(startServerText);

        var amiLatest = await _amazonSsm.GetParameterAsync(
```

```
            new GetParameterRequest() { Name = _amiParam });
        var amiId = amiLatest.Parameter.Value;
        var launchTemplateResponse = await _amazonEc2.CreateLaunchTemplateAsync(
            new CreateLaunchTemplateRequest()
            {
                LaunchTemplateName = _launchTemplateName,
                LaunchTemplateData = new RequestLaunchTemplateData()
                {
                    InstanceType = _instanceType,
                    ImageId = amiId,
                    IamInstanceProfile =
                        new

 LaunchTemplateIamInstanceProfileSpecificationRequest()
                        {
                            Name = _instanceProfileName
                        },
                    KeyName = _keyPairName,
                    UserData = System.Convert.ToBase64String(plainTextBytes)
                }
            });
        return launchTemplateResponse.LaunchTemplate;

    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
 Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
```

```
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("is-default", new List<string>() { "true" })
                }
```

```
            });
        return vpcResponse.Vpcs[0];
    }

    /// <summary>
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("vpc-id", new List<string>() { vpcId}),
                    new ("availability-zone", availabilityZones),
                    new ("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
            subnets.Add(subnet);
        }

        return subnets;
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
```

```
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonClientException)
        {
            Console.WriteLine($"Unable to delete template {templateName}.");
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
```

```
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
```

```
   public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
   {
       var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
           new DescribeIamInstanceProfileAssociationsRequest()
           {
               Filters = new List<Amazon.EC2.Model.Filter>()
               {
                   new ("instance-id", new List<string>() { instanceId })
               },
           });
       return response.IamInstanceProfileAssociations[0];
   }

   /// <summary>
   /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
   /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
   /// used to restart the Python web server.
   /// </summary>
   /// <param name="instanceId">The Id of the instance to update.</param>
   /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
   /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
   /// <returns>Async task.</returns>
   public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
   {
       await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
               new ReplaceIamInstanceProfileAssociationRequest()
               {
                   AssociationId = associationId,
                   IamInstanceProfile = new IamInstanceProfileSpecification()
                   {
                       Name = credsProfileName
                   }
               });
       // Allow time before resetting.
       Thread.Sleep(25000);
       var instanceReady = false;
       var retries = 5;
```

```
        while (retries-- > 0 && !instanceReady)
        {
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(10000);

            var instancesPaginator =
_amazonSsm.Paginators.DescribeInstanceInformation(
                new DescribeInstanceInformationRequest());
            // Get the entire list using the paginator.
            await foreach (var instance in
instancesPaginator.InstanceInformationList)
            {
                instanceReady = instance.InstanceId == instanceId;
                if (instanceReady)
                {
                    break;
                }
            }
        }
        Console.WriteLine($"Sending restart command to instance {instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                        {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
                }
            });
        Console.WriteLine($"Restarted the web server on instance {instanceId}");
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
```

```
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
```

```
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
```

```
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
```

```
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
```

```
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }
}
```

建立包裝 Elastic Load Balancing 動作的類別。

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
 loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
```

```
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
    public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
```

```
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
 exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
 ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
 _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
            new CreateTargetGroupRequest()
            {
                Name = groupName,
                Protocol = protocol,
                Port = port,
                HealthCheckPath = "/healthcheck",
                HealthCheckIntervalSeconds = 10,
                HealthCheckTimeoutSeconds = 5,
                HealthyThresholdCount = 2,
                UnhealthyThresholdCount = 2,
                VpcId = vpcId
```

```
            });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
    /// <param name="subnetIds">Subnets for the load balancer.</param>
    /// <param name="targetGroup">Target group for forwarded requests.</param>
    /// <returns>The new LoadBalancer object.</returns>
    public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
    {
        var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
            new CreateLoadBalancerRequest()
            {
                Name = name,
                Subnets = subnetIds
            });
        var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;
```

```
                loadBalancerReady = loadBalancerState ==
 LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
```

```
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
```

```
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
                        });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
                Console.WriteLine($"Deleted load balancing target group
{groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
```

```
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

建立使用 DynamoDB 模擬建議服務的類別。

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
 configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
```

```
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
```

```
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
 _amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
```

```
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

建立包裝 Systems Manager 動作的類別。

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
```

```
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
```

```
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - AttachLoadBalancerTarget群組
  - CreateAutoScalingGroup
  - CreateInstance設定檔
  - CreateLaunch模板
  - CreateListener
  - CreateLoad平衡器
  - CreateTarget集團
  - DeleteAutoScalingGroup
  - DeleteInstance設定檔
  - DeleteLaunch模板
  - DeleteLoad平衡器
  - DeleteTarget集團
  - DescribeAutoScalingGroups
  - DescribeAvailability區域
  - DescribeIamInstanceProfile協會
  - DescribeInstances
  - DescribeLoad平衡器
  - DescribeSubnets
  - DescribeTarget群組
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfile協會
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

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。

在命令提示中執行互動式案例。

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   public static void main(String[] args) throws IOException,
InterruptedException {
       Scanner in = new Scanner(System.in);
       Database database = new Database();
       AutoScaler autoScaler = new AutoScaler();
       LoadBalancer loadBalancer = new LoadBalancer();

       System.out.println(DASHES);
       System.out.println("Welcome to the demonstration of How to Build and
Manage a Resilient Service!");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("A - SETUP THE RESOURCES");
       System.out.println("Press Enter when you're ready to start deploying
resources.");
       in.nextLine();
       deploy(loadBalancer);
       System.out.println(DASHES);
       System.out.println(DASHES);
       System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
       System.out.println("Press Enter when you're ready.");
       in.nextLine();
       demo(loadBalancer);
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("C - DELETE THE RESOURCES");
       System.out.println("""
               This concludes the demo of how to build and manage a resilient
service.
               To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
               that were created for this demo.
               """);

       System.out.println("\n Do you want to delete the resources (y/n)? ");
       String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

       if (userInput.equals("y")) {
```

```
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
```

```
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
```

```
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
```

```
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
```

```
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
       paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

       System.out.println(
               """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                       """);
       demoChoices(loadBalancer);

       System.out.println(
               """
                       Instead of failing when the recommendation service fails,
the web service can return a static response.
                       While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                       """);
       paramHelper.put(paramHelper.failureResponse, "static");

       System.out.println("""
               Now, sending a GET request to the load balancer endpoint returns
a static response.
```

```
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);
```

```
        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
```

```
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
```

```
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode =
response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
balancer targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
```

```
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
 health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
 between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

建立包裝 Auto Scaling 和 Amazon EC2 動作的類別。

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;
```

```
    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
```

```
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
```

```
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
```

```
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
```

```
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
```

```
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();
```

```
            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " +
secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " +
ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
```

```
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

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

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
```

```
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
               .builder()
               .build();

       DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
       List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
               .collect(Collectors.toList());

       String availabilityZones = String.join(",", availabilityZoneNames);
       LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
               .launchTemplateName(templateName)
               .version("$Default")
               .build();

       String[] zones = availabilityZones.split(",");
       CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
               .launchTemplate(specification)
               .availabilityZones(zones)
               .maxSize(groupSize)
               .minSize(groupSize)
               .autoScalingGroupName(autoScalingGroupName)
               .build();

       try {
           getAutoScalingClient().createAutoScalingGroup(groupRequest);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
       return zones;
   }
```

```
    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
```

```
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
 role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }
```

```
        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
 profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

建立包裝 Elastic Load Balancing 動作的類別。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
```

```
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

.targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
```

```
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
```

```
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
```

```
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
```

```
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
 become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
 loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
 CreateListenerRequest.builder()

 .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .defaultActions(action)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

建立使用 DynamoDB 模擬建議服務的類別。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
 e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
 such
```

```
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
                                    .keyType(KeyType.HASH)
                                    .build(),
                            KeySchemaElement.builder()
                                    .attributeName("ItemId")
                                    .keyType(KeyType.RANGE)
                                    .build())
                    .provisionedThroughput(
                            ProvisionedThroughput.builder()
                                    .readCapacityUnits(5L)
                                    .writeCapacityUnits(5L)
                                    .build())
                    .build();

            getDynamoDbClient().createTable(createTableRequest);
            System.out.println("Creating table " + tableName + "...");

            // Wait until the Amazon DynamoDB table is created.
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                    .tableName(tableName)
```

```
                    .build();

            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Table " + tableName + " created.");

            // Add records to the table.
            populateTable(fileName, tableName);
        }
    }

    public void deleteTable(String tableName) {
        getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
        System.out.println("Table " + tableName + " deleted.");
    }

    // Populates the table with data located in a JSON file using the DynamoDB
    // enhanced client.
    public void populateTable(String fileName, String tableName) throws
IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(getDynamoDbClient())
                .build();
        ObjectMapper objectMapper = new ObjectMapper();
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);
```

```
            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

建立包裝 Systems Manager 動作的類別。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- 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。

- AttachLoadBalancerTarget群組
- CreateAutoScalingGroup
- CreateInstance設定檔
- CreateLaunch模板
- CreateListener
- CreateLoad平衡器
- CreateTarget集團
- DeleteAutoScalingGroup
- DeleteInstance設定檔
- DeleteLaunch模板
- DeleteLoad平衡器
- DeleteTarget集團
- DescribeAutoScalingGroups
- DescribeAvailability區域
- DescribeIamInstanceProfile協會
- DescribeInstances
- DescribeLoad平衡器
- DescribeSubnets
- DescribeTarget群組
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfile協會
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

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。

在命令提示中執行互動式案例。

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0


import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
```

```
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios);
}
```

建立步驟以部署所有資源。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
```

```
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
```

```
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
```

```
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
```

```
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
```

```
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
```

```
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
```

```
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
      // snippet-end:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
```

```
          Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
      }),
    ),
  );
}),
new ScenarioOutput(
  "createdAutoScalingGroup",
  /**
   * @param {{ availabilityZoneNames: string[] }} state
   */
  (state) =>
    MESSAGES.createdAutoScalingGroup
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
      .replace(
        "${AVAILABILITY_ZONE_NAMES}",
        state.availabilityZoneNames.join(", "),
      ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
  type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeVpcs]
  const client = new EC2Client({});
  const { Vpcs } = await client.send(
    new DescribeVpcsCommand({
      Filters: [{ Name: "is-default", Values: ["true"] }],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeVpcs]
  state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
  MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeSubnets]
  const client = new EC2Client({});
```

```
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DescribeSubnets]
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateTargetGroup]
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateTargetGroup]
```

```
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
```

```
  new ScenarioAction("createListener", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateListener]
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateListener]
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.AttachTargetGroup]
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.AttachTargetGroup]
  }),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
```

```
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
```

```
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      } else {
        return MESSAGES.noIpRules;
      }
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      } else {
        return MESSAGES.noIpRules;
      }
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
```

```
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    } else {
      return false;
    }
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
];
```

建立步驟以執行示範。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { readFileSync } from "node:fs";
import { join } from "node:path";
```

```
import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";
```

```
const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetGroups]
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetGroups]

  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  state.targetHealthDescriptions = TargetHealthDescriptions;
```

```
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
```

```
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
```

```
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
```

```
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      // snippet-start:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      // snippet-end:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      // snippet-start:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );
      // snippet-end:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
```

```
      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
```

```
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
```

```
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
```

```
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
  // snippet-start:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

建立步驟以銷毀所有資源。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
```

```
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
```

```
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    } else {
      return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
    }
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    } else {
      return MESSAGES.deletedKeyPair.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
```

```
    }
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          }),
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  }),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    } else {
      return MESSAGES.detachedPolicyFromRole
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  }),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`,
      );
    } else {
```

```
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  }),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    } else {
      return MESSAGES.deletedPolicy.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
  }),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.removeRoleFromInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    } else {
      return MESSAGES.removedRoleFromInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
```

```
    }
  }),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    } else {
      return MESSAGES.deletedInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
```

```
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    } else {
      return MESSAGES.deletedInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    } else {
      return MESSAGES.deletedAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
```

```
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    } else {
      return MESSAGES.deletedLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
```

```
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    } else {
      return MESSAGES.deletedLoadBalancer.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
    // snippet-end:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    } else {
      return MESSAGES.deletedLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
```

```
      );
    }
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    } else {
      return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
```

```
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    } else {
      return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    } else {
      return MESSAGES.detachedSsmOnlyAWSRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
```

```
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    }
  }),
```

```
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyRole.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
```

```
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    } else {
      console.log(err.name);
      throw err;
    }
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
```

```
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的下列主題。
  - AttachLoadBalancerTarget群組
  - CreateAutoScalingGroup
  - CreateInstance設定檔
  - CreateLaunch模板
  - CreateListener
  - CreateLoad平衡器
  - CreateTarget集團
  - DeleteAutoScalingGroup
  - DeleteInstance設定檔
  - DeleteLaunch模板
  - DeleteLoad平衡器
  - DeleteTarget集團
  - DescribeAutoScalingGroups
  - DescribeAvailability區域
  - DescribeIamInstanceProfile協會
  - DescribeInstances
  - DescribeLoad平衡器
  - DescribeSubnets
  - DescribeTarget群組
  - DescribeTargetHealth

- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfile協會
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

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。

在命令提示中執行互動式案例。

```
class Runner:
    def __init__(
        self, resource_path, recommendation, autoscaler, loadbalancer,
 param_helper
    ):
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

    def deploy(self):
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        print(
            "\nFor this demo, we'll use the AWS SDK for Python (Boto3) to create
 several AWS resources\n"
```

```
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n"
            "against various kinds of failures.\n\n"
            "Some of the resources create by this demo are:\n"
        )
        print(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations."
        )
        print(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server."
        )
        print(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones."
        )
        print(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests."
        )
        print("-" * 88)
        q.ask("Press Enter when you're ready to start deploying resources.")

        print(
            f"Creating and populating a DynamoDB table named
'{self.recommendation.table_name}'."
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)
        print("-" * 88)

        print(
            f"Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.\n"
            f"This script starts a Python web server defined in the `server.py`
script. The web server\n"
            f"listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            f"For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            f"run a web server, such as Apache, with least-privileged
credentials.\n"
        )
```

```
        print(
            f"The template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            f"permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            f"that control the flow of the demo.\n"
        )
        self.autoscaler.create_template(startup_script, instance_policy)
        print("-" * 88)

        print(
            f"Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            f"Availability Zone."
        )
        zones = self.autoscaler.create_group(3)
        print("-" * 88)
        print(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            "HTTP requests. You can see these instances in the console or
continue with the demo."
        )
        print("-" * 88)
        q.ask("Press Enter when you're ready to continue.")

        print(f"Creating variables that control the flow of the demo.\n")
        self.param_helper.reset()

        print(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            "defines how the load balancer connects to instances. The load
balancer provides a\n"
            "single endpoint where clients connect and dispatches requests to
instances in the group.\n"
        )
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            [subnet["SubnetId"] for subnet in subnets], target_group
```

```
        )
        self.autoscaler.attach_load_balancer_target_group(target_group)
        print(f"Verifying access to the load balancer endpoint...")
        lb_success = self.loadbalancer.verify_load_balancer_endpoint()
        if not lb_success:
            print(
                "Couldn't connect to the load balancer, verifying that the port
 is open..."
            )
            current_ip_address = requests.get(
                "http://checkip.amazonaws.com"
            ).text.strip()
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                print(
                    "For this example to work, the default security group for
 your default VPC must\n"
                    "allows access from this computer. You can either add it
 automatically from this\n"
                    "example or add it yourself using the AWS Management Console.
\n"
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
 from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
```

```
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success = self.loadbalancer.verify_load_balancer_endpoint()
        if lb_success:
            print("Your load balancer is ready. You can access it by browsing to:
\n")
            print(f"\thttp://{self.loadbalancer.endpoint()}\n")
        else:
            print(
                "Couldn't get a successful response from the load balancer
 endpoint. Troubleshoot by\n"
                "manually verifying that your VPC and security group are
 configured correctly and that\n"
                "you can successfully make a GET request to the load balancer
 endpoint:\n"
            )
            print(f"\thttp://{self.loadbalancer.endpoint()}\n")
        print("-" * 88)
        q.ask("Press Enter when you're ready to continue with the demo.")

    def demo_choices(self):
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            print("-" * 88)
            print(
                "\nSee the current state of the service by selecting one of the
 following choices:\n"
            )
            choice = q.choose("\nWhich action would you like to take? ", actions)
            print("-" * 88)
            if choice == 0:
                print("Request:\n")
                print(f"GET http://{self.loadbalancer.endpoint()}")
                response = requests.get(f"http://{self.loadbalancer.endpoint()}")
                print("\nResponse:\n")
                print(f"{response.status_code}")
                if response.headers.get("content-type") == "application/json":
```

```
                    pp(response.json())
            elif choice == 1:
                print("\nChecking the health of load balancer targets:\n")
                health = self.loadbalancer.check_target_health()
                for target in health:
                    state = target["TargetHealth"]["State"]
                    print(
                        f"\tTarget {target['Target']['Id']} on port
{target['Target']['Port']} is {state}"
                    )
                    if state != "healthy":
                        print(
                            f"\t\t{target['TargetHealth']['Reason']}:
{target['TargetHealth']['Description']}\n"
                        )
                print(
                    f"\nNote that it can take a minute or two for the health
check to update\n"
                    f"after changes are made.\n"
                )
            elif choice == 2:
                print("\nOkay, let's move on.")
                print("-" * 88)

    def demo(self):
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        print("\nResetting parameters to starting values for demo.\n")
        self.param_helper.reset()

        print(
            "\nThis part of the demonstration shows how to toggle different parts
of the system\n"
            "to create situations where the web service fails, and shows how
using a resilient\n"
            "architecture can keep the web service running in spite of these
failures."
        )
        print("-" * 88)

        print(
            "At the start, the load balancer endpoint returns recommendations and
reports that all targets are healthy."
        )
```

```
        self.demo_choices()

        print(
            f"The web service running on the EC2 instances gets recommendations
by querying a DynamoDB table.\n"
            f"The table name is contained in a Systems Manager parameter named
'{self.param_helper.table}'.\n"
            f"To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\nNow, sending a GET request to the load balancer endpoint returns a
failure code. But, the service reports as\n"
            "healthy to the load balancer because shallow health checks don't
check for failure of the recommendation service."
        )
        self.demo_choices()

        print(
            f"Instead of failing when the recommendation service fails, the web
service can return a static response.\n"
            f"While this is not a perfect solution, it presents the customer with
a somewhat better experience than failure.\n"
        )
        self.param_helper.put(self.param_helper.failure_response, "static")
        print(
            f"\nNow, sending a GET request to the load balancer endpoint returns
a static response.\n"
            f"The service still reports as healthy because health checks are
still shallow.\n"
        )
        self.demo_choices()

        print("Let's reinstate the recommendation service.\n")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)
        print(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n"
            "access the DynamoDB recommendation table.\n"
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
```

```
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        print(
            f"\nReplacing the profile for instance {bad_instance_id} with a
profile that contains\n"
            f"bad credentials...\n"
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        print(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n"
            "depending on which instance is selected by the load balancer.\n"
        )
        self.demo_choices()

        print(
            "\nLet's implement a deep health check. For this demo, a deep health
check tests whether\n"
            "the web service can access the DynamoDB table that it depends on for
recommendations. Note that\n"
            "the deep health check is only for ELB routing and not for Auto
Scaling instance health.\n"
            "This kind of deep health check is not recommended for Auto Scaling
instance health, because it\n"
            "risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.\n"
        )
        print(
            "By implementing deep health checks, the load balancer can detect
when one of the instances is failing\n"
            "and take that instance out of rotation.\n"
        )
        self.param_helper.put(self.param_helper.health_check, "deep")
        print(
```

```
            f"\nNow, checking target health indicates that the instance with bad
credentials ({bad_instance_id})\n"
            f"is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy \n"
            f"instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because\n"
            "the load balancer takes unhealthy instances out of its rotation.\n"
        )
        self.demo_choices()

        print(
            "\nBecause the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy\n"
            "instance is to terminate it and let the auto scaler start a new
instance to replace it.\n"
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        print(
            "\nEven while the instance is terminating and the new instance is
starting, sending a GET\n"
            "request to the web service continues to get a successful
recommendation response because\n"
            "the load balancer routes requests to the healthy instances. After
the replacement instance\n"
            "starts and reports as healthy, it is included in the load balancing
rotation.\n"
            "\nNote that terminating and replacing an instance typically takes
several minutes, during which time you\n"
            "can see the changing health check status until the new instance is
running and healthy.\n"
        )
        self.demo_choices()

        print(
            "\nIf the recommendation service fails now, deep health checks mean
all instances report as unhealthy.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\nWhen all instances are unhealthy, the load balancer continues to
route requests even to\n"
            "unhealthy instances, allowing them to fail open and return a static
response rather than fail\n"
            "closed and report failure to the customer."
```

```
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self):
        print(
            "This concludes the demo of how to build and manage a resilient
 service.\n"
            "To keep things tidy and to avoid unwanted charges on your account,
 we can clean up all AWS resources\n"
            "that were created for this demo."
        )
        if q.ask("Do you want to clean up all demo resources? (y/n) ",
 q.is_yesno):
            self.loadbalancer.delete_load_balancer()
            self.loadbalancer.delete_target_group()
            self.autoscaler.delete_group()
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            self.recommendation.destroy()
        else:
            print(
                "Okay, we'll leave the resources intact.\n"
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            )


def main():
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
```

```
        default="../../../workflows/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
 policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    print("-" * 88)
    print(
        "Welcome to the demonstration of How to Build and Manage a Resilient
 Service!"
    )
    print("-" * 88)

    prefix = "doc-example-resilience"
    recommendation = RecommendationService.from_client(
        "doc-example-recommendation-service"
    )
    autoscaler = AutoScaler.from_client(prefix)
    loadbalancer = LoadBalancer.from_client(prefix)
    param_helper = ParameterHelper.from_client(recommendation.table_name)
    runner = Runner(
        args.resource_path, recommendation, autoscaler, loadbalancer,
 param_helper
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
 "destroy"]
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

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


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

建立包裝 Auto Scaling 和 Amazon EC2 動作的類別。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is
                          created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
```

```
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from Boto3 clients.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        """
        as_client = boto3.client("autoscaling")
        ec2_client = boto3.client("ec2")
        ssm_client = boto3.client("ssm")
        iam_client = boto3.client("iam")
        return cls(
            resource_prefix,
            "t3.micro",
            "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
            as_client,
            ec2_client,
            ssm_client,
            iam_client,
        )

    def create_instance_profile(
        self, policy_file, policy_name, role_name, profile_name,
aws_managed_policies=()
    ):
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
```

```
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        with open(policy_file) as file:
            instance_policy_doc = file.read()

        policy_arn = None
        try:
            pol_response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=instance_policy_doc
            )
            policy_arn = pol_response["Policy"]["Arn"]
            log.info("Created policy with ARN %s.", policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Policy %s already exists, nothing to do.", policy_name)
                list_pol_response = self.iam_client.list_policies(Scope="Local")
                for pol in list_pol_response["Policies"]:
                    if pol["PolicyName"] == policy_name:
                        policy_arn = pol["Arn"]
                        break
            if policy_arn is None:
                raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

        try:
            self.iam_client.create_role(
```

```
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info("Created role %s and attached policy %s.", role_name,
policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Role %s already exists, nothing to do.", role_name)
            else:
                raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
```

```
                    f"Couldn't create profile {profile_name} and attach it to
role\n"
                    f"{role_name}: {err}"
                )
        return profile_arn


    def get_instance_profile(self, instance_id):
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't get instance profile association for instance
{instance_id}: {err}"
            )
        else:
            return response["IamInstanceProfileAssociations"][0]


    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
        """
        Replaces the profile associated with a running instance. After the
profile is
        replaced, the instance is rebooted to ensure that it uses the new
profile. When
        the instance is ready, Systems Manager is used to restart the Python web
server.

        :param instance_id: The ID of the instance to update.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                          the specified instance.
```

```
        :param profile_association_id: The ID of the existing profile association
for the
                                       instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)
            inst_ready = False
            tries = 0
            while not inst_ready:
                if tries % 6 == 0:
                    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                    log.info(
                        "Rebooting instance %s and waiting for it to to be
ready.",
                        instance_id,
                    )
                tries += 1
                time.sleep(10)
                response = self.ssm_client.describe_instance_information()
                for info in response["InstanceInformationList"]:
                    if info["InstanceId"] == instance_id:
                        inst_ready = True
            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
            )
```

```
    def delete_instance_profile(self, profile_name, role_name):
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete instance profile {profile_name} or detach "
                    f"policies and delete role {role_name}: {err}"
                )


    def create_key_pair(self, key_pair_name):
        """
```

```
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        :return: The newly created key pair.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't create key pair {key_pair_name}:
{err}")


    def delete_key_pair(self):
        """
        Deletes a key pair.

        :param key_pair_name: The name of the key pair to delete.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't delete key pair {self.key_pair_name}: {err}"
            )
        except FileNotFoundError:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
        except PermissionError:
            log.info(
                "Inadequate permissions to delete key pair %s.",
self.key_pair_name
            )
        except Exception as err:
            raise AutoScalerError(
                f"Couldn't delete key pair {self.key_pair_name}: {err}"
            )
```

```
    def create_template(self, server_startup_script_file, instance_policy_file):
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
```

```
            template = lt_response["LaunchTemplate"]
            log.info(
                "Created launch template %s for AMI %s on %s.",
                self.launch_template_name,
                ami_id,
                self.inst_type,
            )
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.AlreadyExistsException"
            ):
                log.info(
                    "Launch template %s already exists, nothing to do.",
                    self.launch_template_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create launch template
{self.launch_template_name}: {err}."
                )
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
```

```
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete launch template
{self.launch_template_name}: {err}."
                )


    def get_availability_zones(self):
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get availability zones: {err}.")
        else:
            return zones


    def create_group(self, group_size):
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
        :return: The list of Availability Zones specified for the group.
        """
        zones = []
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
```

```
                MaxSize=group_size,
            )
            log.info(
                "Created EC2 Auto Scaling group %s with availability zones %s.",
                self.launch_template_name,
                zones,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExists":
                log.info(
                    "EC2 Auto Scaling group %s already exists, nothing to do.",
                    self.group_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create EC2 Auto Scaling group {self.group_name}:
{err}"
                )
        return zones


    def get_instances(self):
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: Data about the instances.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't get instances for Auto Scaling group
{self.group_name}: {err}"
            )
        else:
            return instance_ids
```

```
    def terminate_instance(self, instance_id):
        """
        Terminates and instances in an EC2 Auto Scaling group. After an instance
is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id, ShouldDecrementDesiredCapacity=False
            )
            log.info("Terminated instance %s.", instance_id)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't terminate instance {instance_id}:
{err}")

    def attach_load_balancer_target_group(self, lb_target_group):
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forward requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't attach load balancer target group
{lb_target_group['TargetGroupName']}\n"
                f"to auto scaling group {self.group_name}"
            )
```

```
   def _try_terminate_instance(self, inst_id):
       stopping = False
       log.info(f"Stopping {inst_id}.")
       while not stopping:
           try:
               self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                   InstanceId=inst_id, ShouldDecrementDesiredCapacity=True
               )
               stopping = True
           except ClientError as err:
               if err.response["Error"]["Code"] == "ScalingActivityInProgress":
                   log.info("Scaling activity in progress for %s. Waiting...",
inst_id)
                   time.sleep(10)
               else:
                   raise AutoScalerError(f"Couldn't stop instance {inst_id}:
{err}.")

   def _try_delete_group(self):
       """
       Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
       the function waits and retries until the group is successfully deleted.
       """
       stopped = False
       while not stopped:
           try:
               self.autoscaling_client.delete_auto_scaling_group(
                   AutoScalingGroupName=self.group_name
               )
               stopped = True
               log.info("Deleted EC2 Auto Scaling group %s.", self.group_name)
           except ClientError as err:
               if (
                   err.response["Error"]["Code"] == "ResourceInUse"
                   or err.response["Error"]["Code"] ==
"ScalingActivityInProgress"
               ):
                   log.info(
                       "Some instances are still running. Waiting for them to
stop..."
                   )
```

```
                    time.sleep(10)
                else:
                    raise AutoScalerError(
                        f"Couldn't delete group {self.group_name}: {err}."
                    )

    def delete_group(self):
        """
        Terminates all instances in the group, deletes the EC2 Auto Scaling
 group.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=self.group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self._try_terminate_instance(inst_id)
                self._try_delete_group()
            else:
                log.info("No groups found named %s, nothing to do.",
 self.group_name)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't delete group {self.group_name}:
 {err}.")


    def get_default_vpc(self):
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
```

```
            raise AutoScalerError(f"Couldn't get default VPC: {err}")
        else:
            return response["Vpcs"][0]


    def verify_inbound_port(self, vpc, port, ip_address):
        """
        Verify the default security group of the specified VPC allows ingress
from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
you
        must instead specify a prefix list ID. You can also temporarily open the
port to
        any IP address while running this example. If you do, be sure to remove
public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specific VPC, and a value that
indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info("Found default security group %s.", sec_group["GroupId"])
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info("Found inbound rule: %s", ip_perm)
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
```

```
                    if not port_is_open:
                        log.info(
                            "The inbound rule does not appear to be open to
either this computer's IP\n"
                            "address of %s, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.",
                            ip_address,
                        )
                    else:
                        break
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't verify inbound rule for port {port} for VPC
{vpc['VpcId']}: {err}"
            )
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id, port, ip_address):
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.

        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
```

```
            raise AutoScalerError(
                f"Couldn't authorize ingress to {sec_group_id} on port {port}
 from {ip_address}: {err}"
            )


    def get_subnets(self, vpc_id, zones):
        """
        Gets the default subnets in a VPC for a specified list of Availability
 Zones.

        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        try:
            response = self.ec2_client.describe_subnets(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )
            subnets = response["Subnets"]
            log.info("Found %s subnets for the specified zones.", len(subnets))
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get subnets: {err}")
        else:
            return subnets
```

建立包裝 Elastic Load Balancing 動作的類別。

```
class LoadBalancer:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, target_group_name, load_balancer_name, elb_client):
        """
        :param target_group_name: The name of the target group associated with
 the load balancer.
```

```
        :param load_balancer_name: The name of the load balancer.
        :param elb_client: A Boto3 Elastic Load Balancing client.
        """
        self.target_group_name = target_group_name
        self.load_balancer_name = load_balancer_name
        self.elb_client = elb_client
        self._endpoint = None


    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from a Boto3 client.

        :param resource_prefix: The prefix to give to AWS resources created by
this class.
        """
        elb_client = boto3.client("elbv2")
        return cls(f"{resource_prefix}-tg", f"{resource_prefix}-lb", elb_client)

    def endpoint(self):
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        if self._endpoint is None:
            try:
                response = self.elb_client.describe_load_balancers(
                    Names=[self.load_balancer_name]
                )
                self._endpoint = response["LoadBalancers"][0]["DNSName"]
            except ClientError as err:
                raise LoadBalancerError(
                    f"Couldn't get the endpoint for load balancer
{self.load_balancer_name}: {err}"
                )
        return self._endpoint


    def create_target_group(self, protocol, port, vpc_id):
        """
        Creates an Elastic Load Balancing target group. The target group
specifies how
```

```
        the load balancer forward requests to instances in the group and how
instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
times and
        lower thresholds. In production, you might want to decrease the
sensitivity of
        your health checks to avoid unwanted failures.

        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=self.target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
            )
            target_group = response["TargetGroups"][0]
            log.info("Created load balancing target group %s.",
self.target_group_name)
        except ClientError as err:
            raise LoadBalancerError(
                f"Couldn't create load balancing target group
{self.target_group_name}: {err}"
            )
        else:
            return target_group


    def delete_target_group(self):
        """
        Deletes the target group.
        """
        done = False
```

```
        while not done:
            try:
                response = self.elb_client.describe_target_groups(
                    Names=[self.target_group_name]
                )
                tg_arn = response["TargetGroups"][0]["TargetGroupArn"]
                self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
                log.info(
                    "Deleted load balancing target group %s.",
self.target_group_name
                )
                done = True
            except ClientError as err:
                if err.response["Error"]["Code"] == "TargetGroupNotFound":
                    log.info(
                        "Load balancer target group %s not found, nothing to
do.",
                        self.target_group_name,
                    )
                    done = True
                elif err.response["Error"]["Code"] == "ResourceInUse":
                    log.info(
                        "Target group not yet released from load balancer,
waiting..."
                    )
                    time.sleep(10)
                else:
                    raise LoadBalancerError(
                        f"Couldn't delete load balancing target group
{self.target_group_name}: {err}"
                    )


    def create_load_balancer(self, subnet_ids, target_group):
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param subnet_ids: A list of subnets to associate with the load balancer.
        :param target_group: An existing target group that is added as a listener
to the
                             load balancer.
        :return: Data about the newly created load balancer.
```

```
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=self.load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info("Created load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info("Waiting for load balancer to be available...")
            waiter.wait(Names=[self.load_balancer_name])
            log.info("Load balancer is available!")
            self.elb_client.create_listener(
                LoadBalancerArn=load_balancer["LoadBalancerArn"],
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                "Created listener to forward traffic from load balancer %s to
target group %s.",
                self.load_balancer_name,
                target_group["TargetGroupName"],
            )
        except ClientError as err:
            raise LoadBalancerError(
                f"Failed to create load balancer {self.load_balancer_name}"
                f"and add a listener for target group
{target_group['TargetGroupName']}: {err}"
            )
        else:
            self._endpoint = load_balancer["DNSName"]
            return load_balancer


    def delete_load_balancer(self):
        """
        Deletes a load balancer.
        """
        try:
```

```
            response = self.elb_client.describe_load_balancers(
                Names=[self.load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[self.load_balancer_name])
        except ClientError as err:
            if err.response["Error"]["Code"] == "LoadBalancerNotFound":
                log.info(
                    "Load balancer %s does not exist, nothing to do.",
                    self.load_balancer_name,
                )
            else:
                raise LoadBalancerError(
                    f"Couldn't delete load balancer {self.load_balancer_name}:
{err}"
                )


    def verify_load_balancer_endpoint(self):
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.
        """
        success = False
        retries = 3
        while not success and retries > 0:
            try:
                lb_response = requests.get(f"http://{self.endpoint()}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    success = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
```

```
                )
                retries -= 1
                time.sleep(10)
        return success

    def check_target_health(self):
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[self.target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            raise LoadBalancerError(
                f"Couldn't check health of {self.target_group_name} targets:
 {err}"
            )
        else:
            return health_response["TargetHealthDescriptions"]
```

建立使用 DynamoDB 模擬建議服務的類別。

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name, dynamodb_client):
        """
        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
```

```
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    @classmethod
    def from_client(cls, table_name):
        """
        Creates this class from a Boto3 client.

        :param table_name: The name of the DynamoDB recommendations table.
        """
        ddb_client = boto3.client("dynamodb")
        return cls(table_name, ddb_client)

    def create(self):
        """
        Creates a DynamoDB table to use a recommendation service. The table has a
        hash key named 'MediaType' that defines the type of media recommended,
such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
        forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file):
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self):
        """
        Deletes the recommendations table.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
```

```
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

建立包裝 Systems Manager 動作的類別。

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table = "doc-example-resilient-architecture-table"
    failure_response = "doc-example-resilient-architecture-failure-response"
    health_check = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name, ssm_client):
        """
        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    @classmethod
    def from_client(cls, table_name):
        ssm_client = boto3.client("ssm")
        return cls(table_name, ssm_client)

    def reset(self):
        """
        Resets the Systems Manager parameters to starting values for the demo.
```

```
        These are the name of the DynamoDB recommendation table, no response when
a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name, value):
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting demo parameter %s to '%s'.", name, value)
        except ClientError as err:
            raise ParameterHelperError(
                f"Couldn't set parameter {name} to {value}: {err}"
            )
```

- 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。
  - AttachLoadBalancerTarget群組
  - CreateAutoScalingGroup
  - CreateInstance設定檔
  - CreateLaunch模板
  - CreateListener
  - CreateLoad平衡器
  - CreateTarget集團
  - DeleteAutoScalingGroup
  - DeleteInstance設定檔

- DeleteLaunch模板
- DeleteLoad平衡器
- DeleteTarget集團
- DescribeAutoScalingGroups
- DescribeAvailability區域
- DescribeIamInstanceProfile協會
- DescribeInstances
- DescribeLoad平衡器
- DescribeSubnets
- DescribeTarget群組
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfile協會
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

建立 IAM 群組，並使用 AWS SDK 將使用者新增至群組

以下程式碼範例顯示做法：

- 建立群組並為其授予完整的 Amazon S3 存取許可。
- 建立一個無權存取 Amazon S3 的新使用者。
- 將使用者新增至群組，並顯示他們現在擁有 Amazon S3 的許可，然後清理資源。

.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。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
 groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
 AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
```

```
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
 { GroupName = groupName });
        return response.Group;
    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
```

```
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
```

```
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
 { UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
```

```
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
 { UserName = userName });
        return response.User;
```

```
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
```

```
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
```

```
    }


    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };
```

```
        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
 policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
```

```
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}


using Microsoft.Extensions.Configuration;

namespace IAMGroups;

public class IAMGroups
{
```

```
    private static ILogger logger = null!;

    // Represents JSON code for AWS full access policy for Amazon Simple
    // Storage Service (Amazon S3).
    private const string S3FullAccessPolicyDocument = "{" +
        " \"Statement\" : [{" +
            " \"Action\" : [\"s3:*\"]," +
            " \"Effect\" : \"Allow\"," +
            " \"Resource\" : \"*\"" +
        "}]" +
    "}";


    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        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<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMGroups>();

        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var groupUserName = configuration["GroupUserName"];
        var groupName = configuration["GroupName"];
        var groupPolicyName = configuration["GroupPolicyName"];
        var groupBucketName = configuration["GroupBucketName"];
```

```
        var wrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayGroupsOverview();
        uiWrapper.PressEnter();

        // Create an IAM group.
        uiWrapper.DisplayTitle("Create IAM group");
        Console.WriteLine("Let's begin by creating a new IAM group.");
        var group = await wrapper.CreateGroupAsync(groupName);

        // Add an inline IAM policy to the group.
        uiWrapper.DisplayTitle("Add policy to group");
        Console.WriteLine("Add an inline policy to the group that allows members
to have full access to");
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) buckets.");

        await wrapper.PutGroupPolicyAsync(group.GroupName, groupPolicyName,
S3FullAccessPolicyDocument);

        uiWrapper.PressEnter();

        // Now create a new user.
        uiWrapper.DisplayTitle("Create an IAM user");
        Console.WriteLine("Now let's create a new IAM user.");
        var groupUser = await wrapper.CreateUserAsync(groupUserName);

        // Add the new user to the group.
        uiWrapper.DisplayTitle("Add the user to the group");
        Console.WriteLine("Adding the user to the group, which will give the user
the same permissions as the group.");
        await wrapper.AddUserToGroupAsync(groupUser.UserName, group.GroupName);

        Console.WriteLine($"User, {groupUser.UserName}, has been added to the
group, {group.GroupName}.");
        uiWrapper.PressEnter();

        Console.WriteLine("Now that we have created a user, and added the user to
the group, let's create an IAM access key.");

        // Create access and secret keys for the user.
        var accessKey = await wrapper.CreateAccessKeyAsync(groupUserName);
        Console.WriteLine("Key created.");
```

```
        uiWrapper.WaitABit(15, "Waiting for the access key to be ready for
use.");

        uiWrapper.DisplayTitle("List buckets");
        Console.WriteLine("To prove that the user has access to Amazon S3, list
the S3 buckets for the account.");

        var s3Client = new AmazonS3Client(accessKey.AccessKeyId,
accessKey.SecretAccessKey);
        var stsClient = new
AmazonSecurityTokenServiceClient(accessKey.AccessKeyId,
accessKey.SecretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client, stsClient);

        var buckets = await s3Wrapper.ListMyBucketsAsync();

        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName}\tcreated on:
{bucket.CreationDate}");
            });
        }

        // Show that the user also has write access to Amazon S3 by creating
        // a new bucket.
        uiWrapper.DisplayTitle("Create a bucket");
        Console.WriteLine("Since group members have full access to Amazon S3,
let's create a bucket.");
        var success = await s3Wrapper.PutBucketAsync(groupBucketName);

        if (success)
        {
            Console.WriteLine($"Successfully created the bucket:
{groupBucketName}.");
        }

        uiWrapper.PressEnter();

        Console.WriteLine("Let's list the user's S3 buckets again to show the new
bucket.");
```

```
        buckets = await s3Wrapper.ListMyBucketsAsync();

        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName}\tcreated on:
 {bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("First delete the bucket we created.");
        await s3Wrapper.DeleteBucketAsync(groupBucketName);

        Console.WriteLine($"Now remove the user, {groupUserName}, from the group,
 {groupName}.");
        await wrapper.RemoveUserFromGroupAsync(groupUserName, groupName);

        Console.WriteLine("Delete the user's access key.");
        await wrapper.DeleteAccessKeyAsync(accessKey.AccessKeyId, groupUserName);

        // Now we can safely delete the user.
        Console.WriteLine("Now we can delete the user.");
        await wrapper.DeleteUserAsync(groupUserName);

        uiWrapper.PressEnter();

        Console.WriteLine("Now we will delete the IAM policy attached to the
 group.");
        await wrapper.DeleteGroupPolicyAsync(groupName, groupPolicyName);

        Console.WriteLine("Now we delete the IAM group.");
        await wrapper.DeleteGroupAsync(groupName);

        uiWrapper.PressEnter();

        Console.WriteLine("The IAM groups demo has completed.");

        uiWrapper.PressEnter();
    }
}
```

```
namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };
```

```
        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
```

```
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
```

```
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
```

```
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。

- AddUserToGroup
- AttachRole政策
- CreateAccess關鍵
- CreateGroup
- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccess關鍵
- DeleteGroup
- DeleteGroup政策
- DeleteUser
- PutGroup政策
- RemoveUserFromGroup

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 建立 IAM AWS STS 使用者並擔任角色

下列程式碼範例示範如何建立使用者並擔任角色。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立沒有許可的使用者。
- 建立一個可授予許可的角色，以列出帳戶的 Amazon S3 儲存貯體。
- 新增政策，讓使用者擔任該角色。
- 使用暫時憑證，擔任角色並列出 Amazon S3 儲存貯體，然後清理資源。

.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。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Add an existing IAM user to an existing IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to add.</param>
    /// <param name="groupName">The name of the group to add the user to.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AddUserToGroupAsync(string userName, string
 groupName)
    {
        var response = await _IAMService.AddUserToGroupAsync(new
 AddUserToGroupRequest
        {
            GroupName = groupName,
            UserName = userName,
        });

        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
```

```
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM group.
    /// </summary>
    /// <param name="groupName">The name to give the IAM group.</param>
    /// <returns>The IAM group that was created.</returns>
    public async Task<Group> CreateGroupAsync(string groupName)
    {
        var response = await _IAMService.CreateGroupAsync(new CreateGroupRequest
 { GroupName = groupName });
        return response.Group;
    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
```

```
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupAsync(string groupName)
    {
        var response = await _IAMService.DeleteGroupAsync(new DeleteGroupRequest
{ GroupName = groupName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an IAM policy associated with an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group associated with the
    /// policy.</param>
    /// <param name="policyName">The name of the policy to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteGroupPolicyAsync(string groupName, string
policyName)
    {
        var request = new DeleteGroupPolicyRequest()
        {
            GroupName = groupName,
            PolicyName = policyName,
        };

        var response = await _IAMService.DeleteGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
```

```
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
 { UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
```

```
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

```
    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
 { UserName = userName });
        return response.User;
```

```
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }
```

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
```

```
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
```

```
    }


    /// <summary>
    /// Remove a user from an IAM group.
    /// </summary>
    /// <param name="userName">The username of the user to remove.</param>
    /// <param name="groupName">The name of the IAM group to remove the user
from.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> RemoveUserFromGroupAsync(string userName, string
groupName)
    {
        // Remove the user from the group.
        var removeUserRequest = new RemoveUserFromGroupRequest()
        {
            UserName = userName,
            GroupName = groupName,
        };

        var response = await
_IAMService.RemoveUserFromGroupAsync(removeUserRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM group.
    /// </summary>
    /// <param name="groupName">The name of the IAM group.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutGroupPolicyAsync(string groupName, string
policyName, string policyDocument)
    {
        var request = new PutGroupPolicyRequest
        {
            GroupName = groupName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };
```

```
        var response = await _IAMService.PutGroupPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
 policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
```

```
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
```

```
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        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<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
```

```
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");
```

```
        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);
```

```
        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();
```

```
        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
```

```
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
 { BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
```

```
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
{ BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
```

```
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
 creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
```

```
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
```

```
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

Bash

AWS CLI 與 Bash 腳本

> **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。

```
#############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
#############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
```

```
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
```

```
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
  policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created  IAM policy named $policy_name"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
```

```
 fi

 local assume_role_policy_document="{
               \"Version\": \"2012-10-17\",
               \"Statement\": [{
                   \"Effect\": \"Allow\",
                   \"Action\": \"sts:AssumeRole\",
                   \"Resource\": \"$role_arn\"}]}"

 local assume_role_policy_name
 assume_role_policy_name=$(generate_random_name "test-assume-role-")

 # shellcheck disable=SC2181
 local assume_role_policy_arn
 assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
 else
   errecho "The policy failed to create."
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
   return 1
 fi

 echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
 sleep 10
 echo_repeat "*" 88
 echo

 echo "Try to list buckets without the new user assuming the role."
 echo_repeat "*" 88
 echo

 # Set the environment variables for the created user.
 # bashsupport disable=BP2001
 export AWS_ACCESS_KEY_ID=$key_name
 # bashsupport disable=BP2001
 export AWS_SECRET_ACCESS_KEY=$key_secret

 local buckets
 buckets=$(s3_list_buckets)
```

```
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. This should not have
happened."
 else
   errecho "Because the role with permissions has not been assumed, listing
buckets failed."
 fi

 echo
 echo_repeat "*" 88
 echo "Now assume the role $iam_role_name and list the buckets."
 echo_repeat "*" 88
 echo

 local credentials

 credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
 # bashsupport disable=BP2001
 export AWS_SESSION_TOKEN=${credentials[2]}

 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
```

```
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
 succeeded because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
 "$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

此案例中使用的 IAM 函數。

```
########################################################################################
# function iam_user_exists
#
```

```
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
```

```
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
```

```
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_create_user_access_key"
  echo "Creates an AWS Identity and Access Management (IAM) key pair."
  echo "  -u user_name   The name of the IAM user."
  echo "  [-f file_name]   Optional file name for the access key output."
  echo ""
}

# Retrieve the calling parameters.
while getopts "u:f:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    f) file_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

response=$(aws iam create-access-key \
  --user-name "$user_name" \
  --output text)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-access-key operation failed.$response"
  return 1
```

```
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }
```

```
# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-role operation failed.\n$response"
  return 1
fi
```

```
  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
```

```
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
```

```
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam detach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
  return 1
```

```
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an WS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
```

```
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
```

```
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an WS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an WS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
```

```
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
  return 1
fi

iecho "delete-access-key response:$response"
iecho
```

```
  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an WS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
```

```
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- 如需 API 詳細資訊，請參閱《AWS CLI 命令參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccess關鍵
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUser政策
- DetachRole政策
- PutUser政策

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。

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;
```

```
    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
 std::endl;
        }

        user = outcome.GetResult().GetUser();
    }
```

```
    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
```

```
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
```

```
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
```

```
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
```

```
                        listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
```

```
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
 role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
 s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
 LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
 std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
```

```
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }

        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }
```

```
    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for C++ API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy

- CreateRole
- CreateUser
- DeleteAccess關鍵
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUser政策
- DetachRole政策
- PutUser政策

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。

```
// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper actions.PolicyWrapper
```

```
 roleWrapper actions.RoleWrapper
 userWrapper actions.UserWrapper
 questioner demotools.IQuestioner
 helper IScenarioHelper
 isTestRun bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
  helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:   sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run() {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
```

```
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser()
 accessKey := scenario.CreateAccessKey(user)
 role := scenario.CreateRoleAndPolicies(user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(accessKey)
 scenario.ListBucketsWithAssumedRole(noPermsConfig, role)
 scenario.Cleanup(user, role)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser() *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(userName)
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
```

```
func (scenario AssumeRoleScenario) CreateAccessKey(user *types.User)
 *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(*user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(user *types.User)
 *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err :=
 scenario.roleWrapper.CreateRole(scenario.helper.GetName(), *user.Arn)
 if err != nil {panic(err)}
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
 if err != nil {panic(err)}
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(*listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {panic(err)}
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(*user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {panic(err)}
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
```

```
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(accessKey
 *types.AccessKey) *aws.Config {
  log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
  scenario.questioner.Ask("Press Enter when you're ready.")
  noPermsConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
 ))
  if err != nil {panic(err)}

  // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

  s3Client := s3.NewFromConfig(noPermsConfig)
  _, err = s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
  if err != nil {
   // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
```

```
  } else {
   log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
 }
 log.Println(strings.Repeat("-", 88))
  return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//   the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
 list the
//    buckets.
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
 generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(noPermsConfig
 *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(context.TODO(),
 &sts.AssumeRoleInput{
  RoleArn:           role.Arn,
  RoleSessionName:   aws.String("AssumeRoleExampleSession"),
  DurationSeconds:   aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(context.TODO(),
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
```

```
 )
 if err != nil {panic(err)}

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(context.TODO(), &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(user *types.User, role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
   policies, err := scenario.roleWrapper.ListAttachedRolePolicies(*role.RoleName)
   if err != nil {panic(err)}
   for _, policy := range policies {
    err = scenario.roleWrapper.DetachRolePolicy(*role.RoleName,
 *policy.PolicyArn)
    if err != nil {panic(err)}
    err = scenario.policyWrapper.DeletePolicy(*policy.PolicyArn)
    if err != nil {panic(err)}
    log.Printf("Detached policy %v from role %v and deleted the policy.\n",
     *policy.PolicyName, *role.RoleName)
   }
   err = scenario.roleWrapper.DeleteRole(*role.RoleName)
   if err != nil {panic(err)}
   log.Printf("Deleted role %v.\n", *role.RoleName)

   userPols, err := scenario.userWrapper.ListUserPolicies(*user.UserName)
   if err != nil {panic(err)}
```

```
   for _, userPol := range userPols {
    err = scenario.userWrapper.DeleteUserPolicy(*user.UserName, userPol)
    if err != nil {panic(err)}
    log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
   }
   keys, err := scenario.userWrapper.ListAccessKeys(*user.UserName)
   if err != nil {panic(err)}
   for _, key := range keys {
    err = scenario.userWrapper.DeleteAccessKey(*user.UserName, *key.AccessKeyId)
    if err != nil {panic(err)}
    log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
   }
   err = scenario.userWrapper.DeleteUser(*user.UserName)
   if err != nil {panic(err)}
   log.Printf("Deleted user %v.\n", *user.UserName)
   log.Println(strings.Repeat("-", 88))
 }

}
```

定義包裝帳號動作的結構。

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy() (*types.PasswordPolicy,
 error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(context.TODO(),
```

```
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders() ([]types.SAMLProviderListEntry,
 error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(context.TODO(),
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

定義包裝政策動作的結構。

```


// 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"`
```

```
}



// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(maxPolicies int32) ([]types.Policy,
 error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(context.TODO(),
 &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(policyName string, actions []string,
  resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
```

```
   Action: actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(context.TODO(),
 &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(policyArn string) (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(context.TODO(), &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(policyArn string) error {
 _, err := wrapper.IamClient.DeletePolicy(context.TODO(), &iam.DeletePolicyInput{
```

```
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

定義包裝角色動作的結構。

```
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(maxRoles int32) ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(context.TODO(),
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
```

```
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(roleName string, trustedUserArn string)
 (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action: []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(context.TODO(),
 &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(roleName string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(context.TODO(),
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
```

```
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(serviceName string,
 description string) (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(context.TODO(),
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(context.TODO(),
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(policyArn string, roleName string)
 error {
```

```
 _, err := wrapper.IamClient.AttachRolePolicy(context.TODO(),
 &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(roleName string)
 ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(context.TODO(),
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(roleName string, policyArn string)
 error {
 _, err := wrapper.IamClient.DetachRolePolicy(context.TODO(),
 &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
```

```
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(roleName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(context.TODO(),
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(roleName string) error {
 _, err := wrapper.IamClient.DeleteRole(context.TODO(), &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

定義包裝使用者動作的結構。

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(maxUsers int32) ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(context.TODO(), &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(context.TODO(), &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
```

```
  user = result.User
 }
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(userName string) (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(context.TODO(),
 &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(userName string, policyName string,
 actions []string,
  roleArn string) error {
 policyDoc := PolicyDocument{
  Version:   "2012-10-17",
  Statement: []PolicyStatement{{
   Effect: "Allow",
   Action: actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
```

```
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(context.TODO(),
 &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(userName string) ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(context.TODO(),
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(userName string, policyName string)
 error {
 _, err := wrapper.IamClient.DeleteUserPolicy(context.TODO(),
 &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(userName string) error {
 _, err := wrapper.IamClient.DeleteUser(context.TODO(), &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(userName string)
 (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(context.TODO(),
 &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
```

```
func (wrapper UserWrapper) DeleteAccessKey(userName string, keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(context.TODO(),
 &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(context.TODO(),
 &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Go API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵

- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUser政策
- DetachRole政策
- PutUser政策

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。

建立可包裝 IAM 使用者動作的函數。

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */
```

```
public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
 <bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
 assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which
 objects are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        String policyName = args[1];
        String roleName = args[2];
```

```
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
```

```
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

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

```
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

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

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
```

```
            return response.role().arn();

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

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

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

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

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

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();
```

```
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteRole(IamClient iam, String roleName, String polArn)
{

       try {
           // First the policy needs to be detached.
           DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                   .policyArn(polArn)
                   .roleName(roleName)
                   .build();

           iam.detachRolePolicy(rolePolicyRequest);

           // Delete the policy.
           DeletePolicyRequest request = DeletePolicyRequest.builder()
                   .policyArn(polArn)
                   .build();

           iam.deletePolicy(request);
           System.out.println("*** Successfully deleted " + polArn);

           // Delete the role.
           DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           iam.deleteRole(roleRequest);
           System.out.println("*** Successfully deleted " + roleName);

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

   public static void deleteKey(IamClient iam, String username, String
accessKey) {
       try {
           DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                   .accessKeyId(accessKey)
                   .userName(username)
                   .build();

           iam.deleteAccessKey(request);
```

```
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

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

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException 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。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策

- PutUser政策

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。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "test_name";
const policyName = "test_policy";
const roleName = "test_role";

/**
```

```
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    } else {
      throw caught;
    }
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
```

```
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
(AWS STS).
  // It's not best practice to use access keys. For more information, see
https://aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;

  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
```

```
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
          RoleName: roleName,
        }),
      ),
  );

  if (!Role) {
    throw new Error("Role not created");
  }

  // Create a policy that allows the user to list S3 buckets.
  const { Policy: listBucketPolicy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Action: ["s3:ListAllMyBuckets"],
            Resource: "*",
          },
        ],
      }),
```

```
    PolicyName: policyName,
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
```

```
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 60 }, () =>
  listBuckets(s3Client),
);

// Clean up.
await iamClient.send(
  new DetachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeletePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
  }),
);

await iamClient.send(
  new DeleteRoleCommand({
    RoleName: Role.RoleName,
  }),
);

await iamClient.send(
  new DeleteAccessKeyCommand({
    UserName: userName,
    AccessKeyId,
  }),
);

await iamClient.send(
  new DeleteUserCommand({
    UserName: userName,
```

```
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- 如需 API 詳細資訊，請參閱《AWS SDK for JavaScript API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

建立可包裝 IAM 使用者動作的函數。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
```

```
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue: String =
        "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
```

```
            "}"

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
```

```
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String
) {
    val stsClient =
        StsClient {
```

```
            region = "us-east-1"
        }

    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials =
        StaticCredentialsProvider {
            accessKeyId = key
            secretAccessKey = secKey
            sessionToken = secToken
        }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 =
        S3Client {
            credentialsProvider = staticCredentials
            region = "us-east-1"
        }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}
```

```
suspend fun deleteRole(
    roleNameVal: String,
    polArn: String
) {
    val iam = IamClient { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}
```

```
@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

```
namespace Iam\Basics;

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

```
use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
```

```
$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}
```

```
$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- 如需 API 詳細資訊，請參閱《AWS SDK for PHP API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策
  - PutUser政策

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。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
```

```
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
```

```
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version": "2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
"
            f"the role."
```

```
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
```

```
    Uses the temporary credentials from the role to list the buckets that are
owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
```

```
        raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
```

```
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- 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。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUser政策
  - DetachRole政策

- PutUser政策

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。

建立一個可授予許可的 IAM 使用者和角色，以列出 Amazon S3 儲存貯體。使用者只有擔任該角色的權利。擔任角色後，請使用暫時性憑證列出該帳戶的儲存貯體。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts("Give AWS time to propagate resources...")
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
```

```
    @logger.info("Tried and failed to create demo user.")
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: "2012-10-17",
      Statement: [{
                    Effect: "Allow",
                    Principal: {'AWS': user.arn},
                    Action: "sts:AssumeRole"
                  }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
```

```
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: "2012-10-17",
      Statement: [{
                   Effect: "Allow",
                   Action: "s3:ListAllMyBuckets",
                   Resource: "arn:aws:s3:::*"
                 }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
```

```
 # @param user [Aws::IAM::User] The user that owns the policy.
 # @param role [Aws::IAM::Role] The role that can be assumed.
 # @return [Aws::IAM::UserPolicy] The newly created policy.
 def create_user_policy(policy_name, user, role)
   policy_document = {
     Version: "2012-10-17",
     Statement: [{
                  Effect: "Allow",
                  Action: "sts:AssumeRole",
                  Resource: role.arn
                }]
   }.to_json
   @iam_client.put_user_policy(
     user_name: user.user_name,
     policy_name: policy_name,
     policy_document: policy_document
   )
   puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated
into a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
```

```
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == "AccessDenied"
      puts("Attempt to list buckets with no permissions: AccessDenied.")
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: "create-use-assume-role-scenario"
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end
```

```
  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
 role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
 policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
 deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
```

```
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts("Try to list buckets with credentials for a user who has no permissions.")
  puts("Expect AccessDenied from this call.")
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key)))
  puts("Now, assume the role that grants permission.")
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key))
  puts("Here are your buckets:")
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts("Thanks for watching!")
  puts("-" * 88)
rescue Aws::Errors::ServiceError => e
  puts("Something went wrong with the demo.")
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Ruby API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccess關鍵
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUser政策
- DetachRole政策
- PutUser政策

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。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;
```

```
    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
```

```
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );
```

```
   let attach_role_policy_result =
       iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
           .await?;
   println!(
       "Attached the policy to the role: {:?}",
       attach_role_policy_result
   );

   let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
   let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
   iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
       .await?;
   println!("Created inline policy.");

   //First, fail to list the buckets with the user.
   let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
   let fail_config = aws_config::from_env()
       .credentials_provider(creds.clone())
       .load()
       .await;
   println!("Fail config: {:?}", fail_config);
   let fail_client: s3Client = s3Client::new(&fail_config);
   match fail_client.list_buckets().send().await {
       Ok(e) => {
           println!("This should not run. {:?}", e);
       }
       Err(e) => {
           println!("Successfully failed with error: {:?}", e)
       }
   }

   let sts_config = aws_config::from_env()
       .credentials_provider(creds.clone())
       .load()
       .await;
   let sts_client: stsClient = stsClient::new(&sts_config);
   sleep(Duration::from_secs(10)).await;
   let assumed_role = sts_client
       .assume_role()
```

```
        .role_arn(assume_role_role.arn())
        .role_session_name(&format!("{}{}", "iam_demo_assumerole_session_",
uuid))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
```

```
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- 如需 API 詳細資訊，請參閱《適用於 Rust 的AWS SDK API 參考》中的下列主題。
  - AttachRole政策
  - CreateAccess關鍵
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccess關鍵
  - DeletePolicy
  - DeleteRole
  - DeleteUser

- DeleteUser政策
- DetachRole政策
- PutUser政策

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 建立唯讀和讀寫 IAM 使用者 AWS

下列程式碼範例示範如何建立使用者並為它們連接政策。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立兩個 IAM 使用者。
- 對一位使用者連接政策，並將物件放在 Amazon S3 儲存貯體中。
- 為第二位使用者連接政策，以便從儲存貯體取得物件。
- 依據使用者憑證取得儲存貯體的不同許可。

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。

建立可包裝 IAM 使用者動作的函數。

```
import logging
import time

import boto3
```

```
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user



def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
```

```
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

建立可包裝 IAM 政策動作的函數。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.
```

```
    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::my-bucket/*' to allow actions on all
 objects
                         in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
```

```
        raise
```

建立可包裝 IAM 存取金鑰動作的函數。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
```

```
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

使用包裝函數建立具有不同政策的使用者，並使用其憑證存取 Amazon S3 儲存貯體。

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
 an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
 permitted
    to perform.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
```

```
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and
{user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN:
{read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to
{user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
```

```
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )

    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
```

```
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's
credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials.
"
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- 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。
  - AttachUser政策
  - CreateAccess鑰匙
  - CreatePolicy
  - CreateUser
  - DeleteAccess鑰匙
  - DeletePolicy
  - DeleteUser
  - DetachUser政策
  - ListUsers
  - UpdateUser

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 管理 IAM 存取金鑰

下列程式碼範例示範如何管理存取金鑰。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立並列出存取金鑰。
- 找出上次使用存取金鑰的時間和方式。
- 更新和刪除存取金鑰。

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。

建立可包裝 IAM 存取金鑰動作的函數。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.
```

```
    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

使用包裝函數執行目前使用者的存取金鑰動作。

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print(
            "The current user already has the maximum of 2 access keys. To run "
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
        )
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
 {new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access
 {last_use['ServiceName']} "
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
```

```
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- 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。
  - CreateAccess關鍵
  - DeleteAccess關鍵
  - GetAccessKeyLast二手
  - ListAccess按鍵
  - UpdateAccess關鍵

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 IAM。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 開發套件管理 IAM 政策

以下程式碼範例顯示做法：

- 建立並列出政策。
- 建立並取得政策版本。
- 將政策復原至先前的版本。
- 刪除政策。

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。

建立可包裝 IAM 政策動作的函數。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::my-bucket/*' to allow actions on all
 objects
                         in the bucket named 'my-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
```

```
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies



def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version": "2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
```

```
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.
```

```
    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
```

```
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

使用包裝函數來建立政策、更新版本，以及取得有關它們的資訊。

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::made-up-bucket-name"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
```

```
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- 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。
  - CreatePolicy
  - CreatePolicy版本
  - DeletePolicy
  - DeletePolicy版本
  - GetPolicy版本
  - ListPolicies
  - ListPolicy版本
  - SetDefaultPolicyVersion

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 管理 IAM 角色

以下程式碼範例顯示做法：

- 建立 IAM 角色。
- 連接和分離角色的政策。
- 刪除角色。

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。

建立可包裝 IAM 角色動作的函數。

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
```

```
    trust_policy = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role



def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise



def detach_policy(role_name, policy_arn):
```

```
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

使用包裝函數建立角色，然後連接和分離政策。

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
```

```
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
"batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- 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。
  - AttachRole政策
  - CreateRole
  - DeleteRole
  - DetachRole政策

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 管理您的 IAM 帳戶

以下程式碼範例顯示做法：

- 取得並更新帳戶別名。
- 產生使用者及憑證的報告。
- 取得帳戶用量摘要。
- 取得帳戶中所有使用者、群組、角色和政策的詳細資訊，包含這些項目彼此之間的關係。

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。

建立可包裝 IAM 帳戶動作的函數。

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
```

```
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise



def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
```

```
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]



def get_summary():
    """
    Gets a summary of account usage.
```

```
    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

呼叫包裝函數以變更帳戶別名並取得有關帳戶的報告。

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
```

```
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != "COMPLETE":
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [
        line.split(",")[:col_count] for line in
cred_report.decode("utf-8").split("\n")
```

```
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)?
 ")
    if see_pw_policy.lower() == "y":
        while True:
            if print_password_policy():
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
```

```
    list_saml_providers(10)

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

- 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。
  - CreateAccount別名
  - DeleteAccount別名
  - GenerateCredential報告
  - GetAccountAuthorizationDetails
  - GetAccount摘要
  - GetCredential報告
  - ListAccount別名

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 AWS SDK 復原 IAM 政策版本

以下程式碼範例顯示做法：

- 依日期順序取得政策版本清單。
- 查找預設政策版本。
- 將先前的政策版本設為預設值。
- 刪除舊的預設版本。

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。

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
```

```
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version
```

- 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。
  - DeletePolicy版本
  - ListPolicy版本
  - SetDefaultPolicyVersion

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK 使用 IAM 政策產生器 AWS API

以下程式碼範例顯示做法：

- 使用物件導向 API 建立 IAM 政策。
- 將 IAM 政策產生器 API 與 IAM 服務搭配使用。

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。

這些範例使用下列匯入。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

建立以時間為基礎的政策。

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
```

```
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_GREATER_THAN)

.key("aws:CurrentTime")

.value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

.operator(IamConditionOperator.DATE_LESS_THAN)

.key("aws:CurrentTime")

.value("2020-06-30T23:59:59Z")))
                                .build();

                // Use an IamPolicyWriter to write out the JSON string to a more
readable
                // format.
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
        }
```

建立具有多個條件的政策。

```
        public String multipleConditionsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")

.addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

.addAction("dynamodb:BatchWriteItem")

.addResource("arn:aws:dynamodb:*:*:table/table-name")
```

```
 .addConditions(IamConditionOperator.STRING_EQUALS

 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                    List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS

 .addSuffix("IfExists"))

 .key("dynamodb:Select")

 .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

在政策中使用主體。

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)

 .addResource("arn:aws:s3:::BUCKETNAME/*")

 .addResource("arn:aws:s3:::BUCKETNAME")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
```

```
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

允許跨帳戶 存取權。

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)

 .addPrincipal(IamPrincipalType.AWS, "111122223333")
                                                .addAction("s3:PutObject")
                                                .addResource("arn:aws:s3:::DOC-
EXAMPLE-BUCKET/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                .key("s3:x-amz-
acl")
                                                                .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

建置並上傳 IamPolicy。

```
        public String createAndUploadPolicyExample(IamClient iam, String
 accountID, String policyName) {
                // Build the policy.
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                .addStatement(IamStatement.builder()
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:PutItem")

 .addResource("arn:aws:dynamodb:us-east-1:" + accountID
```

```
                                                                    + ":table/
exampleTableName")
                                                .build())
                                .build();
                // Upload the policy.
                iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
                return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

下載並使用 IamPolicy。

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
 accountID, String policyName,
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
 policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
 r.policyArn(policyArn));

                String policyVersion =
 getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
 r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned
 from IAM.
                String decodedPolicy =
 URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
 that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
```

```
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));

                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- 如需詳細資訊，請參閱《AWS SDK for Java 2.x 開發人員指南》。
- 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。
  - CreatePolicy
  - GetPolicy
  - GetPolicy版本

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

AWS STS 使用 AWS SDK 的程式碼範例

下列程式碼範例顯示如何搭 AWS STS 配 AWS 軟體開發套件 (SDK) 使用。

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。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

程式碼範例

- AWS STS 使用 AWS SDK 的動作
  - 搭AssumeRole配 AWS 開發套件或 CLI 使用
  - 搭AssumeRoleWithWebIdentity配 AWS 開發套件或 CLI 使用
  - 搭DecodeAuthorizationMessage配 AWS 開發套件或 CLI 使用
  - 搭GetFederationToken配 AWS 開發套件或 CLI 使用
  - 搭GetSessionToken配 AWS 開發套件或 CLI 使用
- AWS STS 使用 AWS SDK 的案例
  - 假設需要使用 SDK 的 MFA 權杖的 AWS STS IAM AWS 角色
  - 使用 SDK AWS STS 為同盟使用者建構 URL AWS
  - AWS STS 使用 SDK 獲取需要 MFA 令牌的會話令 AWS 牌

AWS STS 使用 AWS SDK 的動作

下列程式碼範例示範如何使用 AWS SDK 執 AWS STS 行個別動作。這些摘錄會呼叫 AWS STS API，是來自必須在內容中執行的大型程式碼摘錄。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。如需完整清單，請參閱《AWS Security Token Service (AWS STS) API 參考》。

範例

- 搭AssumeRole配 AWS 開發套件或 CLI 使用
- 搭AssumeRoleWithWebIdentity配 AWS 開發套件或 CLI 使用
- 搭DecodeAuthorizationMessage配 AWS 開發套件或 CLI 使用
- 搭GetFederationToken配 AWS 開發套件或 CLI 使用
- 搭GetSessionToken配 AWS 開發套件或 CLI 使用

搭`AssumeRole`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`AssumeRole`。

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：

- 擔任需要 MFA 字符的 IAM 角色
- 為聯合身分使用者建構 URL

.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;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
```

```
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考AssumeRole中的。

Bash

AWS CLI 與 bash 腳本

> **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。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
```

```
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        ech o"Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

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

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
```

```
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for C++ API 參考AssumeRole中的。

CLI

AWS CLI

擔任角色

下列 assume-role 命令會為 IAM 角色 s3-access-example 擷取一組短期憑證。

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

輸出：

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
```

```
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

該命令的輸出包含存取金鑰、私密金鑰以及可用來向 AWS進行驗證的工作階段字符。

對於 AWS CLI 使用，您可以設置與角色關聯的具名配置文件。當您使用設定檔時， AWS CLI 會為您呼叫 assume-role 並管理認證。如需詳細資訊，請參閱 CLI 使用者指南中的 AWS CLI 中的AWS 使用 IAM 角色。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考AssumeRole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.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;
```

```
/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version": "2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, rn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考AssumeRole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。

建立用戶端。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

擔任 IAM 角色。

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
```

```
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考[AssumeRole](#)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 設定和執行程式碼範例儲存庫](#)。

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
```

```
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for JavaScript API 參考AssumeRole中的。

PowerShell

適用的工具 PowerShell

示例 1：返回一組臨時憑據（訪問密鑰，秘密密鑰和會話令牌），可用於一個小時來訪問請求用戶通常無法訪問的 AWS 資源。傳回的認證具有所承擔之角色的存取原則所允許的權限，以及所提供的原則 (您無法使用提供的原則來授與超過假定角色之存取原則所定義的權限)。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

範例 2：傳回一組臨時登入資料，有效期為一小時，這些登入資料具有與所假定角色之存取原則中所定義的相同權限。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

範例 3：傳回一組臨時認證，提供序號與用來執行指令程式之使用者認證相關聯的 MFA 產生的權杖。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

範例 4：傳回一組已擔任客戶帳戶中定義角色的臨時登入資料。對於第三方可以承擔的每個角色，客戶帳戶必須使用識別碼建立角色，識別碼必須在每次假設角色時傳入-ExternalId 參數。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程AssumeRole式參考中的。

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。

擔任需要 MFA 字符的 IAM 角色，並使用暫時性憑證列出該帳戶的 Amazon S3 儲存貯體。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.
```

```
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AssumeRole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 設定和執行程式碼範例儲存庫。

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: "create-use-assume-role-scenario"
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Ruby API 參考AssumeRole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 assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- 如需 API 的詳細資訊，請參閱 AWS SDK AssumeRole中的 Rust API 參考資料。

Swift

適用於 Swift 的 SDK

> **Note**
>
> 這是適用於預覽版本 SDK 的發行前版本文件。內容可能變動。

> **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。

```
public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
                async throws -> STSClientTypes.Credentials {
    let input = AssumeRoleInput(
        roleArn: role.arn,
        roleSessionName: sessionName
    )
    do {
        let output = try await stsClient.assumeRole(input: input)

        guard let credentials = output.credentials else {
            throw ServiceHandlerError.authError
        }

        return credentials
    } catch {
        throw error
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK AssumeRole中的斯威夫特 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**AssumeRoleWithWebIdentity**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要取得使用網頁身分驗證之角色的短期憑證 (OAuth 2."0)

下列 assume-role-with-web-identity 命令會為 IAM 角色 app1 擷取一組短期憑證。請求通過使用指定的 Web 身份提供商提供的 Web 身份令牌進行身份驗證。另外兩個原則會套用至工作階段，以進一步限制使用者可以執行的動作。傳回的認證會在產生後一小時過期。

```
aws sts assume-role-with-web-identity \
    --duration-seconds 3600 \
    --role-session-name "app1" \
    --provider-id "www.amazon.com" \
    --policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/
webidentitydemopolicy2" \
    --role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
    --web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCz
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1wl7WTI7jn-Pcb6M-
buCgHhFOzTQxod27L9CqnOLio7N3gZAGpsp6n1-
AJBOCJckcyXe2c6uD0srOJeZlKUm2eTDVMf8IehDVI0r1QOnTV6KzzAI3OY87Vd_cVMQ"
```

輸出：

```
{
    "SubjectFromWebIdentityToken": "amzn1.account.AF6RHO7KZU5XRVQJGXK6HB56KR2A"
    "Audience": "client.5498841531868486423.1548@apps.example.com",
    "AssumedRoleUser": {
        "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/
app1",
        "AssumedRoleId": "AROACLKWSDQRAOEXAMPLE:app1"
    }
    "Credentials": {
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
```

```
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    },
    "Provider": "www.amazon.com"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的請求臨時安全憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的AssumeRoleWithWeb識別。

PowerShell

適用的工具 PowerShell

範例 1：針對已透過 Amazon 身分提供者登入驗證的使用者，傳回一組臨時登入資料，有效期為一小時。認證會採用與角色 ARN 所識別之角色相關聯的存取原則。或者，您可以將 JSON 政策傳遞給-Policy 參數，以進一步調整存取權限 (您授與的權限不能超過與角色相關聯之權限的可用權限)。提供給-的值WebIdentityToken 是身分識別提供者傳回的唯一使用者識別碼。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的AssumeRoleWithWeb識別。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**DecodeAuthorizationMessage**配 AWS 開發套件或 CLI 使用

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

CLI

AWS CLI

若要解碼回應要求而傳回的編碼授權訊息

下列decode-authorization-message範例會針對回應 Amazon Web Services 請求而傳回的編碼訊息，解碼有關請求授權狀態的其他資訊。

```
aws sts decode-authorization-message \
    --encoded-message EXAMPLEWodyRNrtlQARDip-
eTA6i6DrlUhHhPQrLWB_lAbl5pAKxl9mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqeO7e2NMk5j_hmzAiChJN-8oy3
Ojau7BMjOTWwOtHPhV_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDgVh1gjqZJFUiQoubv78V1RbHNYnK
pOu3FZjwYStfvTb3GHs3-6rLribGO9jZOktkfE6vqxlFzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VVWPxioqgo
JWP5pwe_mAyqh0NLw-r1S56YC_90onj9A80sNrHlI-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYGtPp5NIeSpuf3fOHanFuBZgENxZQZ2dlH3xJGMTtYayzZrRXjiq_SfX9
FaoPIb8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067POFrZECLjjOtNPBOZCcuEKE
```

輸出：

```
{
    "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,
\"matchedStatements\":{\"items\":[{\"statementId\":\"VisualEditor0\",\"effect
\":\"DENY\",\"principals\":{\"items\":[{\"value\":\"AROA123456789EXAMPLE
\"}]},\"principalGroups\":{\"items\":[]},\"actions\":{\"items\":[{\"value
\":\"ec2:RunInstances\"}]},\"resources\":{\"items\":[{\"value\":\"*
\"}]},\"conditions\":{\"items\":[]}}]},\"failures\":{\"items\":[]},
\"context\":{\"principal\":{\"id\":\"AROA123456789EXAMPLE:Ana\",\"arn
\":\"arn:aws:sts::111122223333:assumed-role/Developer/Ana\"},\"action\":
\"RunInstances\",\"resource\":\"arn:aws:ec2:us-east-1:111122223333:instance/*
\",\"conditions\":{\"items\":[{\"key\":\"ec2:MetadataHttpPutResponseHopLimit\",
\"values\":{\"items\":[{\"value\":\"2\"}]}},{\"key\":\"ec2:InstanceMarketType
\",\"values\":{\"items\":[{\"value\":\"on-demand\"}]}},{\"key\":\"aws:Resource
\",\"values\":{\"items\":[{\"value\":\"instance/*\"}]}},{\"key\":\"aws:Account
\",\"values\":{\"items\":[{\"value\":\"111122223333\"}]}},{\"key\":
\"ec2:AvailabilityZone\",\"values\":{\"items\":[{\"value\":\"us-east-1f\"}]}},
{\"key\":\"ec2:ebsOptimized\",\"values\":{\"items\":[{\"value\":\"false\"}]}},
{\"key\":\"ec2:IsLaunchTemplateResource\",\"values\":{\"items\":[{\"value\":
\"false\"}]}},{\"key\":\"ec2:InstanceType\",\"values\":{\"items\":[{\"value
\":\"t2.micro\"}]}},{\"key\":\"ec2:RootDeviceType\",\"values\":{\"items\":
[{\"value\":\"ebs\"}]}},{\"key\":\"aws:Region\",\"values\":{\"items\":[{\"value
\":\"us-east-1\"}]}},{\"key\":\"ec2:MetadataHttpEndpoint\",\"values\":{\"items
\":[{\"value\":\"enabled\"}]}},{\"key\":\"aws:Service\",\"values\":{\"items
\":[{\"value\":\"ec2\"}]}},{\"key\":\"ec2:InstanceID\",\"values\":{\"items\":
[{\"value\":\"*\"}]}},{\"key\":\"ec2:MetadataHttpTokens\",\"values\":{\"items
\":[{\"value\":\"required\"}]}},{\"key\":\"aws:Type\",\"values\":{\"items
\":[{\"value\":\"instance\"}]}},{\"key\":\"ec2:Tenancy\",\"values\":{\"items
\":[{\"value\":\"default\"}]}},{\"key\":\"ec2:Region\",\"values\":{\"items
```

```
\":[{\"value\":\"us-east-1\"}]}},{\"key\":\"aws:ARN\",\"values\":{\"items\":
[{\"value\":\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}}]}}"
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的政策評估邏輯。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的DecodeAuthorization訊息。

PowerShell

適用的工具 PowerShell

範例 1：針對要求而傳回的所提供編碼訊息內容中包含的其他資訊進行解碼。其他資訊會經過編碼，因為授權狀態的詳細資訊可能會構成要求動作的使用者看不到的特權資訊。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指令程式參考中的DecodeAuthorization訊息。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭`GetFederationToken`配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用`GetFederationToken`。

CLI

AWS CLI

使用 IAM 使用者存取金鑰登入資料傳回一組臨時安全登入資料

下列`get-federation-token`範例會傳回使用者的一組暫時安全性登入資料 (包含存取金鑰 ID、秘密存取金鑰和安全性權杖)。您必須使用 IAM 使`GetFederationToken`用者的長期安全登入資料呼叫作業。

```
aws sts get-federation-token \
    --name Bob \
    --policy file://myfile.json \
    --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
```

```
    --duration-seconds 900
```

myfile.json 的內容：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe*",
            "Resource": "*"
        }
    ]
}
```

輸出：

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QQunWMTfKq0DCOP/////////
```

```
wEQAxoMNDUyOTI1MTcwNTA3Igxuy3AOpuuoLsk3MJwqgQPg8QOd9HuoClUxq26wnc/nm
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
        "Expiration": "2023-12-20T02:06:07+00:00"
    },
    "FederatedUser": {
        "FederatedUserId": "111122223333:Bob",
        "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
    },
    "PackedPolicySize": 36
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的請求臨時安全憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetFederation權杖。

PowerShell

適用的工具 PowerShell

範例 1：使用「Bob」作為聯合使用者的名稱，請求有效期為一小時的聯合權杖。此名稱可用來參考以資源為基礎的政策 (例如 Amazon S3 儲存貯體政策) 中的聯合身分使用者名稱。提供的 IAM 政策採用 JSON 格式，用於縮小 IAM 使用者可用的許可範圍。提供的政策不能授予比授予請求使用者更多的權限，同盟使用者的最終許可是根據傳遞的政策和 IAM 使用者政策的交集限制最嚴格的設定。

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指GetFederation令程式參考中的權杖。

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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭**GetSessionToken**配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetSessionToken。

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：

- 獲取需要 MFA 字符的工作階段字符

CLI

AWS CLI

為 IAM 身分取得一組短期憑證

下列 get-session-token 命令會為進行呼叫的 IAM 身分擷取一組短期憑證。產生的憑證可用於政策要求多重要素驗證 (MFA) 的請求。憑證會在產生後的 15 分鐘過期。

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

輸出：

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

如需詳細資訊，請參閱《AWS IAM 使用者指南》中的請求臨時安全憑證。

- 如需 API 詳細資訊，請參閱AWS CLI 命令參考中的GetSession權杖。

PowerShell

適用的工具 PowerShell

範例 1：傳回包含一段時間內有效之暫時認證的**Amazon.RuntimeAWSCredentials**執行個體。用來請求臨時登入資料的認證是從目前的 shell 預設值推斷出來的。若要指定其他認證，請使用-ProfileName 或-AccessKey /-SecretKey 參數。

```
Get-STSSessionToken
```

輸出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

範例 2：傳回包含有效期為一小時之臨時認證的**Amazon.RuntimeAWSCredentials**執行個體。用來提出要求的認證是從指定的設定檔取得。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

輸出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

範例 3：傳回包含有效期為一小時之臨時認證的**Amazon.RuntimeAWSCredentials**執行個體，該執行個體會使用與帳戶相關聯的 MFA 裝置識別號碼 (其認證在設定檔 'myprofilename' 中指定) 以及裝置提供的值。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

輸出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
```

```
EXAMPLEACCESSKEYID                    2/16/2015 9:12:28 PM
 examplesecretaccesskey...            SamPleTokeN.....
```

- 如需 API 詳細資訊，請參閱AWS Tools for PowerShell 指GetSession令程式參考中的權杖。

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。

透過傳遞 MFA 字符取得工作階段字符，並使用它列出該帳戶的 Amazon S3 儲存貯體。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
```

```
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的GetSession權杖以供 Python 使用 (Boto3) 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

AWS STS 使用 AWS SDK 的案例

下列程式碼範例說明如何在 AWS SDK 中 AWS STS 實作常見案例。這些案例會示範如何透過在其中呼叫多個函式來完成特定工作 AWS STS。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

- 假設需要使用 SDK 的 MFA 權杖的 AWS STS IAM AWS 角色
- 使用 SDK AWS STS 為同盟使用者建構 URL AWS
- AWS STS 使用 SDK 獲取需要 MFA 令牌的會話令 AWS 牌

假設需要使用 SDK 的 MFA 權杖的 AWS STS IAM AWS 角色

下列程式碼範例示範如何擔任需要 MFA 權杖的角色。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立一個可授予許可的 IAM 角色，以列出 Amazon S3 儲存貯體。
- 建立 IAM 使用者，該使用者只有在提供 MFA 憑證時才具有擔任該角色的許可。
- 為使用者註冊 MFA 裝置。
- 擔任角色並使用暫時性憑證列出 S3 儲存貯體。

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 IAM 使用者，註冊 MFA 裝置，並建立一個可授予許可以列出 S3 儲存貯體的角色。使用者只有擔任該角色的權利。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
 MFA.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the
 role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.
```

```
    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, roles, and
policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
```

```
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
```

```
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.",
end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device, role
```

說明如果沒有 MFA 字符，不允許擔任角色。

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the
role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

擔任可授予許可以列出 S3 儲存貯體的角色，傳遞所需 MFA 字符，並顯示可列出的儲存貯體。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
```

```
        print(bucket.name)
```

銷毀為示範所建立的資源。

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

使用先前定義的函數執行此案例。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
```

```
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AssumeRole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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使用 SDK AWS STS 為同盟使用者建構 URL AWS

以下程式碼範例顯示做法：

- 建立可對目前帳戶的 Amazon S3 資源授予唯讀存取權的 IAM 角色。
- 從 AWS 聯合端點取得安全性權杖。
- 建構可用來使用聯合憑證存取主控台的 URL。

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。

建立可對目前帳戶的 S3 資源授予唯讀存取權的角色。

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.",
 end="")
```

```
    progress_bar(10)

    return role
```

從 AWS 聯合端點取得安全性權杖，並建構可用來使用聯合認證存取主控台的 URL。

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS
 Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS)
 that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS
 federation
       endpoint, includes the sign-in token for authentication, and redirects to
       the AWS Management Console with permissions defined by the role that was
       specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
 granted.
                            The current user must have permission to assume the
 role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        "sessionId": temp_credentials["AccessKeyId"],
```

```
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
 string
    # before making the request.
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management
 Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

銷毀為示範所建立的資源。

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
```

```
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

使用先前定義的函數執行此案例。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件AssumeRole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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

AWS STS 使用 SDK 獲取需要 MFA 令牌的會話令 AWS 牌

下列程式碼範例示範如何獲取需要 MFA 權杖的工作階段字符。

> **Warning**
>
> 為避免安全風險，在開發專用軟體或使用真實資料時，請勿使用 IAM 使用者進行身分驗證。相反地，搭配使用聯合功能和身分提供者，例如 AWS IAM Identity Center。

- 建立一個可授予許可的 IAM 角色，以列出 Amazon S3 儲存貯體。
- 建立 IAM 使用者，該使用者只有在提供 MFA 憑證時才具有擔任該角色的許可。
- 為使用者註冊 MFA 裝置。
- 提供 MFA 憑證以取得工作階段權杖，並使用暫時性憑證列出 S3 儲存貯體。

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 IAM 使用者，註冊 MFA 裝置，並建立角色，授予許可以便在使用 MFA 憑證時讓使用者僅列出 S3 儲存貯體。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
```

```
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
```

```
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version": "2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
 True}},
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user list
 buckets, "
        f"but only when MFA credentials are present."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

透過傳遞 MFA 字符獲取暫時性工作階段憑證，並使用憑證列出該帳戶的 S3 儲存貯體。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
```

```
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

銷毀為示範所建立的資源。

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
```

```
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

使用先前定義的函數執行此案例。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
```

```
        print("Thanks for watching!")
```

- 如需 API 詳細資訊，請參閱 AWS SDK 中的GetSession權杖以供 Python 使用 (Boto3) 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 IAM。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

IAM 中的安全性和 AWS STS

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要了解適用於 AWS Identity and Access Management (IAM) 的合規計劃，請參閱AWS 合規計劃的合規計劃AWS服務範圍。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用 AWS Identity and Access Management (IAM) 和 AWS Security Token Service (AWS STS) 時套用共同的責任模型。下列主題說明如何設定 IAM，以及如 AWS STS 何符合安全與合規目標。您也會學到如何使用其他可 AWS 協助您監控和保護 IAM 資源的服務。

目錄

AWS 安全認證

與之互動時 AWS，您可以指定 AWS 安全登入資料以驗證您的身分，以及您是否有權存取要求的資源。 AWS 使用安全認證來驗證和授權您的請求。

例如，如果要從 Amazon Simple Storage Service (Amazon S3) 儲存貯體下載受保護的檔案，您的憑證必須允許此存取動作。如果您的認證未顯示您獲得下載檔案的授權，請 AWS 拒絕您的要求。不過，您不需要您的 AWS 安全登入資料即可在公開共用的 Amazon S3 儲存貯體中下載檔案。

中有不同類型的使用者 AWS。所有 AWS 使用者都有安全認證。有帳戶擁有者 (root 使用者) AWS IAM Identity Center 的使用者、聯合身分使用者和 IAM 使用者。

使用者擁有長期或暫時安全憑證。根使用者、IAM 使用者和存取金鑰具有不會過期的長期安全憑證。若要保護長期憑證，您應有現成的程序以管理存取金鑰、變更密碼和啟用 MFA。

IAM 角色和聯合身分使用者具有臨時安全登入資料。 AWS IAM Identity Center 的使用者暫時安全憑證會在定義的一段時間或使用者結束工作階段後過期。暫時憑證的作用幾乎與長期憑證完全相同，而其差異如下：

- 暫時安全憑證是「短期」的，如其名稱所暗示。其可設定為在任何地方持續幾分鐘到幾小時不等。憑據過期後， AWS 不再識別它們或允許從它們發出的 API 請求中進行任何類型的訪問。
- 暫時性安全憑證不會與使用者一起儲存，但會動態產生並在請求時提供給使用者。當暫時性安全憑證到期時 (或者即使在此之前)，使用者可以請求新的憑證，只要使用者的請求仍具有可這麼做的許可。

因此，相較於長期憑證，暫時性憑證具有下列優點：

- 您不需要在應用程式中散佈或內嵌長期 AWS 安全性登入資料。
- 您可以為使用者提供對資 AWS 源的存取權，而不必為使用者定義 AWS 身分。暫時憑證是角色和聯合身分的基礎。
- 臨時安全憑證的存留期有限，因此當不再需要時，您不需要更新它們或明確予以撤銷。暫時性安全憑證到期之後，就無法重複使用。您可以指定憑證的有效期，達到最長限制。

安全考量

建議您在決定適用於 AWS 帳戶的安全規定時，考量下列資訊：

- 當您建立時 AWS 帳戶，我們會建立帳戶 root 使用者。根使用者 (帳戶擁有者) 的憑證可以完整存取帳戶中的所有資源。您對 root 使用者執行的第一項工作是授與您的其他使用者管理權限， AWS 帳戶 以便將 root 使用者的使用量降到最低。
- 您無法使用 IAM 政策來明確拒絕根使用者存取權。您只能使用 AWS Organizations 服務控制原則 (SCP) 來限制根使用者的權限。

- 如果您忘記或遺失根使用者密碼，則必須擁有與您帳戶相關聯之電子郵件地址的存取權，才能重設密碼。
- 如果遺失根使用者存取金鑰，則您必須能夠以根使用者身分登入您的帳戶，才能建立新的金鑰。
- 請勿將您的根使用者用於日常任務。請將其用來執行只能由根使用者執行的任務。如需檢視需要您以根使用者身分登入的任務完整清單，請參閱 需要根使用者憑證的任務。
- 安全登入資料是帳戶專屬的資料。如果您可以存取多個 AWS 帳戶，則每個帳戶都有單獨憑證。
- 策略會決定使用者、角色或使用者群組成員可以執行的動作、對哪些 AWS 資源以及在何種情況下可執行的動作。使用政策，您可以安全地控制 AWS 帳戶. AWS 服務 如果您必須修改或撤銷許可以回應安全性事件，則您可以刪除或修改政策，而不是直接變更身分。
- 請務必將緊急存取 IAM 使用者的登入憑證，以及為程式設計存取而建立的任何存取金鑰儲存在安全的位置。如果您遺失存取金鑰，則必須登入帳戶建立新金鑰。
- 強烈建議您使用 IAM 角色和聯合身分使用者提供的暫時憑證，而不要使用 IAM 使用者和存取金鑰提供的長期憑證。

聯合身分

同盟身分識別是具有外部身分識別的使用者，這些身分被授與可用來存取安全 AWS 帳戶 資源的臨時 AWS 認證。外部身分可以是來自公司身分存放區 (例如 LDAP 或 Windows Active Directory) 或來自第三方 (例如 Login with Amazon, Facebook, or Google)。同盟身分識別不會使用 AWS Management Console 或 AWS 存取入口網站登入。

若要啟用聯合身分登入 AWS，您必須建立包含 `https://signin.aws.amazon.com/federation` 的自訂 URL。如需詳細資訊，請參閱 啟用自訂身分識別代理存取主 AWS 控台。

如需聯合身分的詳細資訊，請參閱 身分提供者與聯合。

多重要素驗證 (MFA)

多重要素驗證 (MFA) 為可以存取 AWS 帳戶的使用者提供多一層級的安全防護。若要提升安全性，我們建議您要求對 AWS 帳戶根使用者 憑證和所有 IAM 使用者進行 MFA。如需詳細資訊，請參閱 在中使用多因素身份驗證（MFA） AWS。

當您啟用 MFA 並登入時 AWS 帳戶，系統會提示您輸入登入憑證，以及 MFA 裝置產生的回應，例如代碼、觸控或輕觸或生物識別掃描。當您新增 MFA 時，您的 AWS 帳戶 設定和資源會更加安全。

預設情況下，MFA 未激活。您可以前往安全憑證頁面或 AWS Management Console中的 IAM 儀表板，針對 AWS 帳戶根使用者 啟用並管理 MFA 裝置。如需針對 IAM 使用者啟用 MFA 的詳細資訊，請參閱 為中的使用者啟用 MFA 裝置 AWS。

如需有關使用多重要素驗證 (MFA) 裝置登入的詳細資訊，請參閱 透過您的 IAM 登入頁面來使用 MFA 裝置。

程式設計存取權

您可以提供 AWS 存取金鑰，以便對 AWS 或進行程式設計呼叫 AWS Command Line Interface 或 AWS Tools for PowerShell。建議盡可能使用短期存取金鑰。

當您建立長期存取金鑰時，會將存取金鑰 ID (例如 AKIAIOSFODNN7EXAMPLE) 和私密存取金鑰 (例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY) 建立為一個集合。只有在您建立的時候才可使用私密存取金鑰進行下載。如果您沒有下載私密存取金鑰，或者遺失，則必須重新建立。

在許多情況下，您不需要永遠不會過期的長期存取金鑰 (就像您為 IAM 使用者建立存取金鑰時一樣)。反之，您可以建立 IAM 角色並產生暫時安全憑證。暫時安全憑證包含存取金鑰 ID 和私密存取金鑰，但其中也包含指出憑證何時到期的安全權杖。臨時安全資料過期之後即不再有效。

開頭為的存取金鑰 ID AKIA 是 IAM 使用者或 AWS 帳戶 根使用者的長期存取金鑰。以開頭的存取金鑰 ID ASIA 是您使用 AWS STS 作業建立的暫時認證存取金鑰。

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。

哪個使用者需要程式設計存取權？	到	By
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

長期存取金鑰的替代方案

對於許多常見使用案例，有可以替代長期存取金鑰的方案。為了提高帳戶安全性，請考慮以下事項。

- 請勿在應用程式程式碼或程式碼儲存庫中內嵌長期存取金鑰和秘密存取金鑰，而是使用 AWS Secrets Manager或其他機密管理解決方案，因此您不必以明文形式對金鑰進行硬式編碼。然後，應用程式或用戶端可以在需要時擷取密碼。如需詳細資訊，請參閱什麼是 AWS Secrets Manager？在《AWS Secrets Manager 使用者指南》中。

- 盡可能使用 IAM 角色產生暫時安全憑證 - 務必盡可能使用機制頒發暫時安全憑證，而非長期存取金鑰。暫時安全憑證更加安全，因為它們不會與使用者一起儲存，但會動態產生並在請求時提供給使用者。臨時安全憑證的生命週期有限，因此您不必對其進行管理或更新。提供暫時存取金鑰的機制包含

IAM 角色或 IAM Identity Center 使用者的身分驗證。對於在您以外執行的電腦，可 AWS 以使用任何地方的AWS Identity and Access Management 角色。

- 針對 AWS Command Line Interface (AWS CLI) 或 **aws-shell** 使用長期存取金鑰的替代方案 – 替代方案包含以下內容。
  - AWS CloudShell是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console您可以 AWS 服務 通過首選的外殼（Bash，PowerShell 或 Z 外殼）運行 AWS CLI 命令。當您這樣做時，無需下載或安裝命令列工具。如需詳細資訊，請參閱《 AWS CloudShell使用者指南》中的什麼是AWS CloudShell ？。
  - AWS CLI 與版本 2 集成 AWS IAM Identity Center （IAM 身份中心）。您可以驗證使用者並提供短期認證來執行 AWS CLI 命令。若要進一步了解，請參閱使用AWS IAM Identity Center 者指南中的AWS CLI 與 IAM 身分中心整合和使 AWS CLI 用AWS Command Line Interface 者指南中的設定使用 IAM 身分中心。
- 請勿為需要存取應用程式的人類使用者建立長期存取金鑰，或者 AWS 服務 — IAM Identity Center 可以產生臨時存取登入資料供外部 IdP 使用者存取 AWS 服務。這樣就不需要在 IAM 中建立和管理長期憑證。在 IAM Identity Center 中，建立可授予外部 IdP 使用者存取權的 IAM Identity Center 許可集。然後，將 IAM 身分中心的群組指派給所選權限集中的群組 AWS 帳戶。如需詳細資訊，請參閱《使用指南》中的「什麼是 AWS IAM Identity Center」、「Connect 至您的外部身分識別提供AWS IAM Identity Center 者」和「權限集」。
- 不要將長期存取金鑰存放在 AWS 運算服務中，而是將 IAM 角色指派給運算資源。這會自動提供暫時憑證以授予存取權。例如，當您建立連接到 Amazon EC2 執行個體的執行個體設定檔時，可以將 AWS 角色指派給執行個體，並將其提供給其所有應用程式。執行個體設定檔包含該角色，並且可讓 Amazon EC2 執行個體上執行的程式取得臨時憑證。若要進一步了解，請參閱利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

AWS 使用您的 AWS 認證存取

AWS 需要不同類型的安全登入資料，具體取決於您存取的方式 AWS 和 AWS 使用者類型。例如，在使用存取金鑰對 AWS進行程式設計呼叫時，使用 AWS Management Console 的登入憑證。此外，您使用的每個身分，無論是帳戶根使用者、 AWS Identity and Access Management (IAM) 使用者、使用者或聯合身分，都在其中 AWS都有唯一的登入資料。 AWS IAM Identity Center

如需如 step-by-step 何 AWS 根據您的使用者類型登入的指示，請參閱AWS 登入使用者指南 AWS中的如何登入。

AWS 安全審計指引

定期稽核您的安全組態，確保其符合您目前的業務需求。稽核的目的是讓您有機會移除不需要的 IAM 使用者、角色、群組和政策，確保使用者和軟體沒有過多的許可。

以下是有系統地檢閱和監控 AWS 資源以取得安全性最佳實務的準則。

> **Tip**
>
> 透過使用 AWS Security Hub 監視您 IAM 的使用狀況，因為它關係到安全最佳實務。Security Hub 會透過安全控制來評估資源組態和安全標準，協助您遵守各種合規架構。如需有關使用 Security Hub 評估 IAM 資源的詳細資訊，請參閱《 AWS Security Hub 使用者指南》中的 AWS Identity and Access Management 控制項。

目錄

何時執行安全性稽核

在下列情況下稽核安全組態：

- 定期。定期執行本文件中所述的步驟，做為確保安全性的最佳實務。
- 如果組織內部有所變化，例如人員離職。
- 如果您已停止使用一或多個個別 AWS 服務，以確認您已移除帳戶中的使用者不再需要的權限。
- 如果您已在帳戶中新增或移除軟體，例如 Amazon EC2 執行個體上的應用程式、 AWS OpsWorks 堆疊、 AWS CloudFormation 範本等。

- 如果您懷疑未經授權的人員可能存取了您的帳戶。

稽核準則

當您檢閱帳戶的安全組態時，請依照以下指導方針：

- 完整徹底。查看安全組態的所有層面，包括很少使用的部分。
- 絕不假設。如果您不熟悉安全組態的某些部分 (例如，特定政策或角色存在背後的原因)，請查清楚業務的需要，直到理解潛在的風險為止。
- 保持簡單。為了讓稽核 (和管理) 能更輕鬆執行，請使用 IAM 群組、IAM 角色、一致的命名機制和直截了當的政策。

檢閱您的 AWS 帳戶憑證

稽核 AWS 帳戶憑證時，請採取下列步驟：

1. 如果您有未使用根帳戶的存取金鑰，可移除它們。我們強烈建議您不要在日常工作中使用根存取金鑰 AWS，而是使用具有臨時登入資料的使用者，例如 AWS IAM Identity Center 的使用者。
2. 如果您需要帳戶的存取金鑰，請確保在需要時更新這些金鑰。

檢閱 IAM 使用者

在稽核現有的 IAM 使用者時，請採取以下步驟：

1. 列出您的使用者，然後刪除不需要的使用者。
2. 從不需要存取權的群組中移除使用者。
3. 檢閱使用者所在群組中所連接的政策。請參閱檢閱 IAM 政策的要訣。
4. 刪除使用者不需要，或者可能已公開的安全登入資料。例如，用於應用程式的 IAM 使用者不需要密碼 (這只需要登入 AWS 網站)。同樣地，如果使用者不使用存取金鑰，也就沒有理由具備存取金鑰。如需詳細資訊，請參閱管理 IAM 使用者的密碼和管理 IAM 使用者的存取金鑰。

   您可以產生並下載「憑證報告」，其中會列出帳戶中的所有 IAM 使用者，及其各種憑證的狀態，包括密碼、存取金鑰和 MFA 裝置。對於密碼和存取金鑰、憑證報告會顯示密碼或存取金鑰最近使用的日期和時間。請考慮從您的帳戶中移除最近未使用的憑證。(請勿移除您的緊急存取使用者。) 如需詳細資訊，請參閱取得 AWS 帳戶的認證報告。

5. 對於需要長期憑證的使用案例，請視需要更新密碼和存取金鑰。如需詳細資訊，請參閱管理 IAM 使用者的密碼和管理 IAM 使用者的存取金鑰。
6. 最佳作法是要求人類使用者與身分識別提供者使用同盟，才能 AWS 使用臨時登入資料進行存取。如果可能，請從 IAM 使用者轉換為聯合身分使用者，例如 IAM Identity Center 的使用者。保留應用程式所需的最少 IAM 使用者數量。

檢閱 IAM 群組

在稽核 IAM 群組時，請採取以下步驟：

1. 列出您的群組，然後刪除未使用的群組。
2. 檢閱每個群組中的使用者，並移除不屬於該群組的使用者。
3. 檢閱群組連接的政策。請參閱檢閱 IAM 政策的要訣。

檢閱 IAM 角色

在稽核 IAM 角色時，請採取以下步驟：

1. 列出您的角色，然後刪除未使用的角色。
2. 檢閱角色的信任政策。確認您知道委託人是誰，而且了解為什麼該帳戶或使用者必須要能擔任該角色。
3. 檢閱角色的存取政策，確認授與適當的權限給能擔任該角色的人，請參閱 檢閱 IAM 政策的要訣。

檢閱 SAML 和 OpenID Connect (OIDC) 的 IAM 提供者

如果您為了建立 SAML 或 OIDC 身分提供者 (IdP) 的信任而建立了 IAM 實體，請採取下列步驟：

1. 刪除未使用的供應商。
2. 下載並檢閱每個 SAML IdP 的 AWS 中繼資料文件，並確保文件能反映您目前的業務需求。
3. 從 SAML 取得最新的中繼資料文件， IdPs 並在 IAM 中更新供應商。

檢閱行動應用程式

如果您已建立提出要求的行動應用程式 AWS，請執行下列步驟：

1. 確認行動應用程式不包含內嵌的存取金鑰，即使位於加密儲存裝置中也是如此。

2. 使用專屬用途的 API 取得應用程式的暫時憑證。

> **Note**
>
> 建議您使用 Amazon Cognito 來管理應用程式中的使用者身分。此服務可讓您使用 Login with Amazon、Facebook、Google 或任何與身分提供者相容的 OpenID Connect (OIDC)，驗證使用者的身分。如需詳細資訊，請參閱《Amazon Cognito 開發人員指南》中的Amazon Cognito 身分集區。

檢閱 IAM 政策的要訣

政策是功能強大而且微妙的工具，因此請務必研究並了解每個政策所授予的許可。檢閱政策時，請使用以下指導方針：

- 將政策連接至群組或角色，而非個別使用者。如果個別使用者具有政策，請確認您了解該使用者需要政策的原因。
- 確保 IAM 使用者、群組和角色具有所需的許可，且沒有任何其他許可。
- 使用 IAM 政策模擬器測試連接到使用者或群組的政策。
- 請記住，使用者的許可是所有適用政策的結果，包括身分型政策 (針對使用者、群組或角色) 和以資源為基礎的政策 (針對 Amazon S3 儲存貯體、Amazon SQS 佇列、Amazon SNS 主題和金鑰等資源)。 AWS KMS 請務必檢查套用到使用者的所有政策，並了解授予個別使用者的完整權限集。
- 請注意，讓使用者建立 IAM 使用者、群組、角色或政策，並將政策連接至主體實體，即表示對該使用者有效授予帳戶中所有資源的所有許可。可以建立政策並將它們連接到使用者、群組或角色的使用者也會授予他們本身任何許可。一般而言，只要是您不信任的使用者或角色，就不要授予能完全存取帳戶中資源的 IAM 許可。執行安全稽核時，請確認將下列 IAM 許可授予受信任身分：
  - `iam:PutGroupPolicy`
  - `iam:PutRolePolicy`
  - `iam:PutUserPolicy`
  - `iam:CreatePolicy`
  - `iam:CreatePolicyVersion`
  - `iam:AttachGroupPolicy`
  - `iam:AttachRolePolicy`
  - `iam:AttachUserPolicy`

- 確認政策不會對您沒在使用的服務授予許可。例如，如果您使用AWS 受管政策，請確定您帳戶中正在使用的 AWS 受管理策略適用於您實際使用的服務。若要瞭解帳戶中正在使用哪些 AWS 受管政策，請使用 IAM GetAccountAuthorizationDetailsAPI (AWS CLI 命令:aws iam get-account-authorization-details)。
- 如果政策授予使用者許可，以啟動 Amazon EC2 執行個體，也可能會允許 iam:PassRole 動作，但若是如此，應該明確列出使用者可傳遞到 Amazon EC2 執行個體的角色。
- 檢查 Action 或 Resource 元素中包含 * 的任何值。如果可能，請授予使用者需要的個別動作和資源的 Allow 存取權。不過，以下原因說明為何在政策中適合使用 *：
  - 此政策旨在授予管理層級的許可。
  - 基於便利性考量，萬用字元會用於一組類似的動作 (例如，Describe*)，而且您放心以此方式參考的完整動作清單。
  - 萬用字元會用來以指示一個類別的資源或資源路徑 (例如，arn:aws:iam::*account-id*:users/division_abc/*)，而且您放心在該類別或路徑中授予所有資源的存取權。
  - 服務動作不支援資源層級權限，而資源的唯一選擇是 *。
- 檢查政策名稱，確認其反映政策的功能。例如，雖然政策名稱中可能會有「唯讀」兩字，但是實際上該政策可能會授予寫入或變更的許可。

如需有關規劃安全性稽核的詳細資訊，請參閱《AWS 架構中心》中的安全性、身分與合規的最佳實務。

資料保護 AWS Identity and Access Management

AWS 共用責任模型適用於中的資料保護 AWS Identity and Access Management。如此模型中所述，AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需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，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。

- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 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 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

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文字欄位中，例如名稱欄位。這包括當您使用主控台、API 或 AWS SDK AWS 服務 使用 IAM 或其他人時。AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。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。

IAM 和中的資料加密 AWS STS

資料加密通常分為兩類：靜態加密和傳輸中加密。

靜態加密

由 IAM 收集和儲存的資料都會進行靜態加密。

- IAM – IAM 內收集並存放的資料包括 IP 地址、客戶帳戶中繼資料，以及包含密碼的客戶識別資料。系統會透過使用 AES 256 或使用 SHA 256 進行雜湊，對客戶帳戶中繼資料和客戶識別資料進行靜態加密。
- AWS STS— 除了將成功、錯誤和錯誤請求記錄到服務的服務記錄外，不 AWS STS 會收集客戶內容。

傳輸中加密

客戶識別資料 (包括密碼) 會在傳輸過程中使用 TLS 1.2 和 1.3 進行加密。所有 AWS STS 端點都支援 HTTPS 來加密傳輸中的資料。如需 AWS STS 端點清單，請參閱區域與端點。

IAM 和金鑰管理 AWS STS

您無法使用 IAM 或 AWS STS. 如需有關加密金鑰的詳細資訊，請參閱什麼是 AWS KMS？ 在 AWS Key Management Service 開發人員指南中

IAM 和中的網際網路流量隱私權 AWS STS

對 IAM 的請求必須使用 Transport Layer Security 通訊協定 (TLS) 提出。您可以使用 VPC 端點保護與 AWS STS 服務的連線安全。如需進一步了解，請參閱介面 VPC 端點。

登錄和監控 AWS Identity and Access Management

監控是維持 AWS Identity and Access Management (IAM)、() 和其他 AWS 解決方案的可靠性、 AWS Security Token Service 可用性和效能的重要組成部分。AWS STS AWS 提供數種工具來監控您的 AWS 資源並回應潛在事件：

- AWS CloudTrail擷取 IAM 和 AWS STS 作為事件的所有 API 呼叫，包括來自主控台和 API 呼叫的呼叫。若要進一步了解如何 CloudTrail 搭配 IAM 使用 AWS STS，請參閱使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail。若要取得有關的更多資訊 CloudTrail，請參閱AWS CloudTrail 使用者指南。
- AWS Identity and Access Management 存取分析器可協助您識別組織和帳戶中與外部實體共用的資源，例如 Amazon S3 儲存貯體或 IAM 角色。這有助於您發現非預期存取資源和資料的情況，避免產生安全性風險。若要進一步了解，請參閱 什麼是 IAM Access Analyzer？。
- Amazon 會即時 CloudWatch監控您的 AWS 資源和執行 AWS 的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以 CloudWatch 追蹤 Amazon EC2 執行個體的 CPU 使用率或其他指標，並在需要時自動啟動新執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon CloudWatch 日誌可協助您從 Amazon EC2 執行個體和其他來源監控 CloudTrail、存放和存取日誌檔。 CloudWatch 記錄檔可以監控記錄檔中的資訊，並在符合特定臨界值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。

如需適用於 IAM 的其他資源和安全最佳實務，請參閱 AWS Identity and Access Management中的安全最佳實務和使用案例。

AWS Identity and Access Management的合規驗證

協力廠商稽核員會評估 AWS Identity and Access Management (IAM) 的安全性與合規性，做為多個 AWS 合規計劃的一部分。這些計劃包括 SOC、PCI、FedRAMP、ISO 等等。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您因應各種合規性需求，例如 PCI DSS，滿足特定合規性架構所規定的入侵偵測需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

韌性 AWS Identity and Access Management

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域具有多個實體分離和隔離的可用區域，這些可用區域透過低延遲、高輸送量和高度備援的網路連線。如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

AWS Identity and Access Management (IAM) 和 AWS Security Token Service (AWS STS) 是自我維持、以區域為基礎的服務，可在全球範圍內使用。

IAM 是非常重要的 AWS 服務。在中執行的每項作業都 AWS 必須經過 IAM 的驗證和授權。IAM 會根據 IAM 中儲存的身分和政策檢查每個請求，以判斷是否允許或拒絕請求。IAM 設計有一個單獨的控制平面和資料平面，以便服務即使在意外故障期間也會進行驗證。授權中使用的 IAM 資源 (例如角色和政策) 會儲存在控制平面中。IAM 客戶可以使用諸如 `DeletePolicy` 和 `AttachRolePolicy` 等 IAM 操作來變更這些資源的組態。這些組態變更請求會移至控制平面。所有商業都有一個 IAM 控制平面 AWS 區域，該平面位於美國東部 (維吉尼亞北部) 區域。然後，IAM 系統會將組態變更傳播至每個已啟用 AWS 區域 的 IAM 資料平面。IAM 資料平面基本上是 IAM 控制平面組態資料的唯讀複本。每個都 AWS 區域 有一個完全獨立的 IAM 資料平面執行個體，可針對來自相同區域的請求執行身份驗證和授權。在每個區域中，IAM 資料平面至少分佈在三個可用區域，並具有足夠的容量來容忍可用區域的損失，而不會造成任何客戶損失。IAM 控制平面和資料平面都是針對零計劃停機而構建，以客戶看不到的方式執行所有軟體更新和擴展操作。

AWS STS 預設情況下，請求始終會移至單一全域端點。您可以使用區域 AWS STS 端點來降低延遲，或是為應用程式提供額外的備援。如需進一步了解，請參閱AWS STS 在一個管理 AWS 區域。

某些事件可能會中斷網路之 AWS 區域 間的通訊。不過，即使您無法與全域 IAM 端點通訊，仍然 AWS STS 可以驗證 IAM 主體，IAM 可以授權您的請求。中斷通訊的事件的具體詳細資料將決定您存取 AWS 服務的能力。在大多數情況下，您可以繼續在 AWS 環境中使用 IAM 登入資料。下列情況可能適用於中斷通訊的事件。

IAM 使用者的存取金鑰

使用長期 IAM 使用者存取金鑰，您可以在區域中無限期地進行驗證。當您使用 AWS Command Line Interface 和 API 時，您可以提供 AWS 存取金鑰， AWS 以便在程式設計要求中驗證您的身分。

> **Important**
>
> 作為最佳實務，我們建議您的使用者使用暫時性憑證而非長期存取金鑰進行登入。

臨時憑證

您至少可以在 24 小時內向 AWS STS 區域服務端點申請新的臨時登入資料。下列 API 操作會產生暫時性憑證。

- AssumeRole
- AssumeRoleWithWebIdentity
- AssumeRoleWith薩姆爾

- GetFederationToken
- GetSessionToken

主體和許可

- 您可能無法在 IAM 中新增、修改或移除主體或許可。
- 您的憑證可能不會反映您最近在 IAM 中套用的許可變更。如需詳細資訊，請參閱 我所做的變更不一定都會立刻生效。

AWS Management Console

- 您可能可以使用區域登入端點以 IAM 使用者身分登入 AWS Management Console 。區域登入端點具有下列 URL 格式。

  `https://{Account ID}.signin.aws.amazon.com/console?region={Region}`

  範例：https://111122223333.signin.aws.amazon.com/console?region=us-west-2
- 您可能無法完成 Universal 2nd Factor (U2F) 多重要素驗證 (MFA)。

IAM 彈性的最佳實務

AWS 已經內置了可用區域的彈性。 AWS 區域 當您在與環境互動的系統中觀察下列 IAM 最佳實務時，可以充分利用該彈性。

1. 使用 AWS STS 區域服務端點，而非預設的全域端點。
2. 檢閱環境的組態，尋找定期建立或修改 IAM 資源的重要資源，並準備使用現有 IAM 資源的備用解決方案。

AWS Identity and Access Management中的基礎設施安全

AWS Identity and Access Management 是受管理的服務，受到 AWS 全球網路安全性的保護。有關 AWS 安全服務以及如何 AWS 保護基礎結構的詳細資訊，請參閱AWS 雲端安全 若要使用基礎架構安全性的最佳做法來設計您的 AWS 環境，請參閱安全性支柱架構良 AWS 好的架構中的基礎結構保護。

您可以使用 AWS 已發佈的 API 呼叫透過網路存取 IAM。使用者端必須支援下列專案：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密(PFS)的密碼套件，例如 DHE (Ephemeral Diffie-Hellman)或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可以透過 IAM HTTPS API 直接向服務發出 HTTPS 請求，以程式設計方式來存取 IAM。查詢 API 會傳回安全憑證等敏感資訊，所以您必須對所有 API 請求使用 HTTPS。當您使用 HTTPS API 時，必須包含使用您的憑證來數位簽署請求的程式碼。

您可從任何網路位置呼叫這些 API 操作，而 IAM 確實可支援資源型存取政策，以供納入依來源 IP 地址為主的限制。您也可以使用 IAM 政策來控制從特定 Amazon Virtual Private Cloud (Amazon VPC) 的端點或特定 VPC的存取。實際上，這會將對指定 IAM 資源的網路存取從網路內的特定 VPC 隔離出來 AWS 。

中的配置和漏洞分析 AWS Identity and Access Management

AWS 處理基本安全性工作，例如客體作業系統 (OS) 和資料庫修補、防火牆組態和嚴重損壞修復。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱以下 資源：

- 共同的責任模型
- Amazon Web Services：安全程序概觀 (白皮書)

下列資源也會解決 AWS Identity and Access Management (IAM) 中的組態和弱點分析：

- AWS Identity and Access Management的合規驗證
- AWS Identity and Access Management中的安全最佳實務和使用案例

AWSAWS Identity and Access Management 存取分析器的受管理原則

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS 受管政策。

IAM ReadOnly 存取權

若要允許 IAM 資源的唯讀存取，請使用 IAMReadOnlyAccess 受管政策。此政策授予取得和列出所有 IAM 資源的許可。它允許檢視使用者、群組、角色、政策、身分提供者和 MFA 裝置的詳細資訊與活動報告。它不包括建立或刪除資源或存取 IAM Access Analyzer 資源的能力。查看政策以了解此政策支援之服務和動作的完整清單。

身分存取UserChange權

使用 IAMUserChangePassword 受管政策可允許 IAM 使用者變更自己的密碼。

您可以設定 IAM 帳戶設定與密碼政策，以允許 IAM 使用者變更其 IAM 帳戶密碼。當您允許此動作時，IAM 會將下列政策連接至每位使用者：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM AccessAnalyzer FullAccess

使用受IAMAccessAnalyzerFullAccess AWS 管政策允許管理員存取 IAM 存取分析器。

許可群組

此政策會根據提供的許可集分組到陳述式中。

- IAM Access Analyzer – 允許對 IAM Access Analyzer 中所有資源的完整管理許可。
- 建立服務連結角色 – 允許管理員建立服務連結角色，允許 IAM Access Analyzer 代表您分析其他服務中的資源。此許可允許建立僅供 IAM Access Analyzer 使用的服務連結角色。
- AWS Organizations – 允許管理員在 AWS Organizations中針對組織使用 IAM Access Analyzer。在中啟用 IAM Access Analyzer 的受信任存取權後 AWS Organizations，管理帳戶的成員可以檢視其組織中的發現結果。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "access-analyzer:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "access-analyzer.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
```

```
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListRoots"
      ],
      "Resource": "*"
    }
  ]
}
```

IAM AccessAnalyzer ReadOnly 存取權

使用受IAMAccessAnalyzerReadOnlyAccess AWS 管政策允許 IAM 存取分析器的唯讀存取權。

若要允許存取 IAM Access Analyzer 的唯讀存取權 AWS Organizations，請建立客戶受管政策，以允許受IAM AccessAnalyzer FullAccess AWS 管政策中的「描述」和「列出」動作。

服務層級許可

此政策提供 IAM Access Analyzer 的唯讀存取權。此政策中不包含任何其他服務許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAccessAnalyzerReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "access-analyzer:CheckAccessNotGranted",
        "access-analyzer:CheckNoNewAccess",
        "access-analyzer:Get*",
        "access-analyzer:List*",
        "access-analyzer:ValidatePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

AccessAnalyzerServiceRole政策

您無法附加 AccessAnalyzerServiceRolePolicy 到 IAM 實體。此政策會連接到服務連結角色，而此角色可讓 IAM Access Analyzer 代表您執行動作。如需詳細資訊，請參閱針對 AWS Identity and Access Management 存取分析器使用服務連結角色。

許可群組

此政策允許存取 IAM Access Analyzer，分析來自多個 AWS 服務的資源中繼資料。

- Amazon DynamoDB 支援 — 允許檢視 DynamoDB 庫串流和表格。
- Amazon Elastic Compute Cloud — 允許描述 IP 地址、快照和 VPC 的許可。
- Amazon Elastic Container Registry — 允許描述映像儲存庫和擷取儲存庫政策的許可。
- Amazon Elastic File System — 允許檢視 Amazon EFS 檔案系統的說明，並檢視 Amazon EFS 檔案系統資源層級政策的許可。
- AWS Identity and Access Management — 允許擷取有關指定角色的資訊，並列出具有指定路徑前置詞的 IAM 角色的許可。允許擷取有關使用者、使用者群組、登入設定檔、存取金鑰和服務上次存取資料的資訊。
- AWS Key Management Service — 允許檢視有關 KMS 金鑰及其金鑰政策和授予詳細資訊的許可。
- AWS Lambda — 允許檢視 Lambda 別名、函數、層和別名相關資訊的許可。
- AWS Organizations— 允許組 Organizations 權限，並允許在 AWS 組織內建立分析器作為信任區域。
- Amazon Relational Database Service — 允許檢視有關 Amazon RDS 資料庫快照和 Amazon RDS 資料庫叢集快照詳細資訊的許可。
- 亞馬遜簡單儲存服務 — 允許許可檢視有關使用 Amazon S3 Express One 儲存類別的 Amazon S3 存取點、儲存貯體和 Amazon S3 目錄儲存貯體的詳細資訊。
- AWS Secrets Manager — 允許檢視有關附加至秘密的秘密和資源政策詳細資訊的許可。
- Amazon Simple Notification Service — 允許檢視有關主題詳細資訊的許可。
- Amazon Simple Queue Service — 允許檢視有關指定佇列詳細資訊的許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessAnalyzerServiceRolePolicy",
      "Effect": "Allow",
```

```
      "Action": [
        "dynamodb:GetResourcePolicy",
        "dynamodb:ListStreams",
        "dynamodb:ListTables",
        "ec2:DescribeAddresses",
        "ec2:DescribeByoipCidrs",
        "ec2:DescribeSnapshotAttribute",
        "ec2:DescribeSnapshots",
        "ec2:DescribeVpcEndpoints",
        "ec2:DescribeVpcs",
        "ec2:GetSnapshotBlockPublicAccessState",
        "ecr:DescribeRepositories",
        "ecr:GetRepositoryPolicy",
        "elasticfilesystem:DescribeFileSystemPolicy",
        "elasticfilesystem:DescribeFileSystems",
        "iam:GetRole",
        "iam:ListEntitiesForPolicy",
        "iam:ListRoles",
        "iam:ListUsers",
        "iam:GetUser",
        "iam:GetGroup",
        "iam:GenerateServiceLastAccessedDetails",
        "iam:GetServiceLastAccessedDetails",
        "iam:ListAccessKeys",
        "iam:GetLoginProfile",
        "iam:GetAccessKeyLastUsed",
        "iam:ListRolePolicies",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListUserPolicies",
        "iam:GetUserPolicy",
        "iam:ListAttachedUserPolicies",
        "iam:GetPolicy",
        "iam:GetPolicyVersion",
        "iam:ListGroupsForUser",
        "kms:DescribeKey",
        "kms:GetKeyPolicy",
        "kms:ListGrants",
        "kms:ListKeyPolicies",
        "kms:ListKeys",
        "lambda:GetFunctionUrlConfig",
        "lambda:GetLayerVersionPolicy",
        "lambda:GetPolicy",
        "lambda:ListAliases",
```

```
        "lambda:ListFunctions",
        "lambda:ListLayers",
        "lambda:ListLayerVersions",
        "lambda:ListVersionsByFunction",
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
        "organizations:ListRoots",
        "rds:DescribeDBClusterSnapshotAttributes",
        "rds:DescribeDBClusterSnapshots",
        "rds:DescribeDBSnapshotAttributes",
        "rds:DescribeDBSnapshots",
        "s3:DescribeMultiRegionAccessPointOperation",
        "s3:GetAccessPoint",
        "s3:GetAccessPointPolicy",
        "s3:GetAccessPointPolicyStatus",
        "s3:GetAccountPublicAccessBlock",
        "s3:GetBucketAcl",
        "s3:GetBucketLocation",
        "s3:GetBucketPolicyStatus",
        "s3:GetBucketPolicy",
        "s3:GetBucketPublicAccessBlock",
        "s3:GetMultiRegionAccessPoint",
        "s3:GetMultiRegionAccessPointPolicy",
        "s3:GetMultiRegionAccessPointPolicyStatus",
        "s3:ListAccessPoints",
        "s3:ListAllMyBuckets",
        "s3:ListMultiRegionAccessPoints",
        "s3express:GetBucketPolicy",
        "s3express:ListAllMyDirectoryBuckets",
        "sns:GetTopicAttributes",
        "sns:ListTopics",
        "secretsmanager:DescribeSecret",
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:ListSecrets",
        "sqs:GetQueueAttributes",
        "sqs:ListQueues"
```

```
      ],
      "Resource": "*"
    }
  ]
}
```

IAM 和 IAM Access Analyzer 對 AWS 受管政策的更新

檢視自服務開始追蹤這些變更以來，IAM 和 AWS 受管政策的更新詳細資料。如需有關此頁面變更的自動提醒，請訂閱 IAM 和 IAM Access Analyzer 文件歷程記錄頁面上的 RSS 摘要。

變更	描述	日期
AccessAnalyzerServiceRole原則 — 新增權限	IAM 存取分析器新增了對權限的AccessAnalyzerServiceRolePolicy 支援，可將 IAM 使用者和角色政策的相關資訊擷取至的服務層級許可。	2024年5月30日
AccessAnalyzerServiceRole原則 — 新增權限	IAM 存取分析器新增了許可支援，可將 Amazon EC2 快照的區塊公共存取目前狀態擷取到的服務層級許可。AccessAnalyzerServiceRolePolicy	2024 年 1 月 23 日
AccessAnalyzerServiceRole原則 — 新增權限	IAM 存取分析器將 DynamoDB 串流和表格的支援新增到的服務層級許可。AccessAnalyzerServiceRolePolicy	2024年1月11日
AccessAnalyzerServiceRole原則 — 新增權限	IAM 存取分析器將 Amazon S3 目錄儲存貯體的AccessAna	2023 年 12 月 1 日

變更	描述	日期
	`lyzerServiceRolePolicy` 支援新增至的服務層級許可。	
IAM AccessAnalyzer ReadOnly 存取 — 新增的許可	IAM Access Analyzer 新增了許可，可用來檢查政策更新是否授予其他存取權。 IAM Access Analyzer 需要此許可，才能對您的政策執行政策檢查。	2023 年 11 月 26 日
AccessAnalyzerServiceRole原則 — 新增權限	IAM Access Analyzer 為 `AccessAnalyzerServiceRolePolicy` 服務層級許可新增了 IAM 動作，以支援下列動作： • 列出政策的實體 • 產生上次存取的服務詳細資訊 • 列出存取金鑰資訊	2023 年 11 月 26 日

變更	描述	日期
AccessAnalyzerServiceRole原則 — 新增權限	IAM Access Analyzer 在 `AccessAnalyzerServiceRolePolicy` 服務層級許可中已新增對下列資源類型的支援： • Amazon EBS 磁碟區快照 • Amazon ECR 儲存庫 • Amazon EFS 檔案系統 • Amazon RDS 資料庫快照 • Amazon RDS 資料庫叢集快照 • Amazon SNS 主題	2022 年 10 月 25 日
AccessAnalyzerServiceRole原則 — 新增權限	IAM Access Analyzer 對 `AccessAnalyzerServiceRolePolicy` 的服務層級許可新增了 `lambda:GetFunctionUrlConfig` 動作。	2022 年 4 月 6 日
AccessAnalyzerServiceRole原則 — 新增權限	IAM Access Analyzer 新增了新的 Amazon S3 動作，以分析與多區域存取點相關聯的中繼資料。	2021 年 9 月 2 日
IAM AccessAnalyzer ReadOnly 存取 — 新增的許可	IAM Access Analyzer 新增了新的動作，以授予 `ValidatePolicy` 許可，以允許您使用政策檢查進行驗證。 IAM Access Analyzer 需要此許可，才能對您的政策執行政策檢查。	2021 年 3 月 16 日

變更	描述	日期
IAM Access Analyzer 開始追蹤變更	IAM 存取分析器開始追蹤其 AWS 受管政策的變更。	2021 年 3 月 1 日

使用 AWS Identity and Access Management Access Analyzer

AWS Identity and Access Management Access Analyzer 提供下列功能：

- IAM Access Analyzer 外部存取權分析器可協助您識別組織與帳戶中與外部實體共用的資源。
- IAM Access Analyzer 未使用的存取權分析器可協助您在組織與帳戶中識別未使用的存取權。
- IAM 存取分析器會根據政策文法和 AWS 最佳實務來驗證 IAM 政策。
- IAM Access Analyzer 自訂政策檢查可協助根據您指定的安全標準驗證 IAM 政策。
- IAM 存取分析器會根據 AWS CloudTrail 日誌中的存取活動產生 IAM 政策。

識別與外部實體共用的資源

IAM Access Analyzer 可協助您識別與外部實體共用的組織和帳戶中資源，例如 Amazon S3 儲存貯體或 IAM 角色。這有助於您識別非預期存取資源和資料的情況，避免產生安全性風險。IAM Access Analyzer 會使用邏輯型推理來分析環境中以資源為基礎的政策，藉此識別與外部主體共用的資源。AWS 針對帳戶外部共用資源的每一個執行個體，IAM Access Analyzer 都會產生一份問題清單。調查結果包括存取權及其被授與存取的外部主體的資訊。您可以檢閱問題清單，判斷此為有意為之且安全的存取，還是非預期且有安全風險的存取。除了協助您識別與外部實體共用的資源外，IAM Access Analyzer 也可讓您使用 IAM Access Analyzer 的問題清單，在部署資源使用權限之前預覽您的政策對您資源的公有和跨帳戶存取權的影響。系統會在視覺化摘要儀表板中整理歸納調查結果。儀表板會反白公有與跨帳戶存取權調查結果之間的分隔線，並提供依資源類型分類的調查結果明細。若要深入了解儀表板，請參閱：檢視 IAM Access Analyzer 調查結果儀表板。

> **Note**
>
> 外部實體可以是其他 AWS 帳戶、根使用者、IAM 使用者或角色、聯合身分使用者、 AWS 服務、匿名使用者或其他可用來建立篩選器的實體。如需詳細資訊，請參閱 AWS JSON 政策元素：主體。

啟用 IAM Access Analyzer 後，您可以為整個組織或帳戶建立分析器。您選擇的組織或帳戶也稱為分析器的信任區域。分析器會監控您信任區域內所有支援的資源。在信任區域內主體對資源的任何存取權都會被視為受信任。啟用後，IAM Access Analyzer 會分析在您信任區域中套用至所有支援之資源的政

策。在第一次分析後，IAM Access Analyzer 會定期分析這些政策。如果新增政策或變更現有政策，則 IAM Access Analyzer 會在大約 30 分鐘內分析新增的或更新的政策。

分析政策時，如果 IAM Access Analyzer 識別到會將存取權授與不在信任區域內的外部主體的政策，就會產生問題清單。每個問題清單都包含資源、具有該資源存取權的外部實體，以及授與之許可的詳細資訊，以便您可以採取適當的動作。您可以檢視問題清單中包含的詳細資訊，以判斷資源存取權是有意為之的，還是您應解決的潛在風險。當您將政策新增至資源或更新現有政策時，IAM Access Analyzer 會分析該政策。IAM Access Analyzer 也會定期分析所有以資源為基礎的政策。

在特定情況下，在極少數情況下，IAM Access Analyzer 不會收到新增或更新政策的通知，這可能會導致產生的發現結果延遲。如果您建立或刪除與 Amazon S3 儲存貯體關聯的多區域存取點，或更新多區域存取點的政策，IAM Access Analyzer 最多可能需要 6 小時才能產生或解決發現結果。此外，如果 AWS CloudTrail 記錄傳遞有傳遞問題，原則變更不會觸發發現項目中報告的資源重新掃描。發生這種情況時，IAM Access Analyzer 會在下次定期掃描 (24 小時內) 期間分析該新增或更新的政策。如果您要確認對政策所做的變更是否可以解決問題清單中報告的存取問題，您可以重新掃描問題清單中報告的資源，方法為使用 Findings (問題清單) 詳細資訊頁面中的 Rescan (重新掃描) 連結，或使用 IAM Analyzer API 的 `StartResourceScan` 操作。如需進一步了解，請參閱解決問題清單。

> **Important**
>
> IAM 存取分析器只會分析套用至啟用該功能之相同 AWS 區域中資源的政策。若要監控 AWS 環境中的所有資源，您必須建立分析器，以便在使用支援 AWS 資源的每個區域中啟用 IAM Access Analyzer。

IAM Access Analyzer 會分析下列資源類型：

- Amazon Simple Storage Service 儲存貯體
- Amazon Simple Storage Service 目錄儲存貯體
- AWS Identity and Access Management 角色
- AWS Key Management Service 鑰匙
- AWS Lambda 函數和圖層
- Amazon Simple Queue Service 佇列
- AWS Secrets Manager 秘密
- Amazon Simple Notification Service 主題
- Amazon Elastic Block Store 磁碟區快照

- Amazon Relational Service 資料庫快照
- Amazon Relational Database Service 資料庫叢集快照
- Amazon Elastic Container Registry 儲存庫
- Amazon Elastic File System 檔案系統
- Amazon DynamoDB 串流
- Amazon DynamoDB 資料表

識別授予 IAM 使用者和角色的未使用存取權

IAM 存取分析器可協助您識別和檢閱 AWS 組織和帳戶中未使用的存取權。IAM Access Analyzer 會持續監控 AWS 組織和帳戶中的所有 IAM 角色和使用者，並針對未使用的存取權產生調查結果。調查結果會反白未使用的角色、IAM 使用者未使用的存取金鑰，以及 IAM 使用者未使用的密碼。調查結果可讓您了解作用中 IAM 角色和使用者未使用的服務和動作。

外部存取權與未使用的存取權分析器調查結果，會在視覺化摘要儀表板中整理歸納。儀表板會反白顯示發現項目最多的您 AWS 帳戶 ，並依類型提供發現項目的明細。如需儀表板中的詳細資訊，請參閱檢視 IAM Access Analyzer 調查結果儀表板。

IAM Access Analyzer 會檢閱 AWS 組織和帳戶中所有角色的上次存取資訊，以協助您識別未使用的存取權。IAM 動作上次存取的資訊可協助識別 AWS 帳戶中的角色未使用的動作。如需詳細資訊，請參閱 AWS 使用上次存取的資訊精簡權限。

根據 AWS 最佳實務驗證政策

您可以使用 IAM Access Analyzer 政策驗證功能提供的基本政策檢查，根據 IAM 政策文法和 AWS 最佳實務來驗證您的政策。您可以使用 IAM 主控台中的 AWS CLI AWS API 或 JSON 政策編輯器建立或編輯政策。您可以檢視政策驗證檢查問題清單，包含安全警告、錯誤、一般警告和政策的建議。這些發現項目提供可行的建議，可協助您撰寫功能正常且符合 AWS 最佳作法的原則。若要進一步了解如何使用政策驗證來驗證政策，請參閱：IAM Access Analyzer 政策驗證。

根據您指定的安全標準驗證政策

您可以使用 IAM Access Analyzer 自訂政策檢查，根據您指定的安全標準來驗證 IAM 政策。您可以使用 IAM 主控台中的 AWS CLI AWS API 或 JSON 政策編輯器建立或編輯政策。您可以透過主控台檢查與現有版本相比，更新版政策是否授予新的存取權。透過 AWS CLI 和 AWS API，您也可以檢查政策

不允許您認為重要的特定 IAM 動作。這些檢查會反白授予新存取權的政策陳述式。您可以更新政策陳述式並重新執行檢查，直到政策符合安全標準為止。若要進一步了解如何使用自訂政策檢查來驗證政策，請參閱：IAM Access Analyzer 自訂政策檢查。

產生政策

IAM Access Analyzer 會分析您的日 AWS CloudTrail 誌，以識別 IAM 實體 (使用者或角色) 在指定日期範圍內使用的動作和服務。然後它會產生以該存取活動為基礎的 IAM 政策。您可以使用產生的政策，將其連接至 IAM 使用者或角色，以進一步調整該實體的許可。若要進一步了解如何使用 IAM Access Analyzer 來產生政策，請參閱 產生 IAM Access Analyzer 政策。

IAM Access Analyzer 定價

IAM Access Analyzer 會根據每月每個分析器所分析的 IAM 角色和使用者數量，收取未使用的存取權分析費用。

- 您需為建立的每個未使用的存取權分析器支付費用。
- 若跨多個區域建立未使用的存取權分析器，您將須為每個分析器支付費用。
- 系統不會針對未使用的存取活動分析服務連結角色，也不會將這些角色包含在已分析的 IAM 角色總數中。

IAM Access Analyzer 會根據向 IAM Access Analyzer 發出的檢查新存取權 API 請求數量，來收取自訂政策檢查費用。

如需 IAM Access Analyzer 的完整費用與定價清單，請參閱 IAM Access Analyzer 定價。

若要查看您的帳單，請前往 AWS Billing and Cost Management 主控台中的帳單與成本管理儀表板。您的帳單內含用量報告的連結，可提供帳單的詳細資訊。若要進一步了解 AWS 帳戶 帳單，請參閱AWS Billing 使用者指南

如果您對帳 AWS 單、帳戶和活動有任何疑問，請聯絡 AWS Support。

外部和未使用的存取權調查結果

IAM Access Analyzer 會針對 AWS 帳戶 或組織中的外部存取權和未使用的存取權產生調查結果。針對外部存取權，若執行個體採用的資源類型政策會將信任區域內的資源存取權授予信任區域以外的主體，

則 IAM Access Analyzer 會為每個執行個體產生調查結果。當您建立外部存取分析器時，您可以選擇 AWS 帳戶 要分析的組織。該分析器會將所選組織或帳戶中的任何主體認定為可信任的。由於相同組織或帳戶中的主體是可信任的，分析器的信任區域就會包含該組織或帳戶內的資源和主體。在信任區域內共享的任何項目都是安全的，因此 IAM Access Analyzer 不會產生問題清單。例如，如果您選擇組織做為分析器的信任區域，則該組織中的所有資源和主體都會在信任區域內。若您將其中一個組織成員帳戶中的 Amazon S3 儲存貯體許可授予另一個組織成員帳戶中的主體，IAM Access Analyzer 並不會產生調查結果。但若將許可授與非組織成員帳戶中的主體，IAM Access Analyzer 就會產生問題清單。

IAM 存取分析器也會針對 AWS 組織和帳戶中授予的未使用存取權產生發現項目。當您建立未使用的存取分析器時，IAM Access Analyzer 會持續監控 AWS 組織和帳戶中的所有 IAM 角色和使用者，並針對未使用的存取產生發現項目。IAM Access Analyzer 會針對未使用的存取權產生下列類型的調查結果：

- 未使用的角色：在指定使用期間內沒有存取活動的角色。
- 未使用的 IAM 使用者存取金鑰和密碼：屬於 IAM 使用者的憑證，可用來存取您的 AWS 帳戶。
- 未使用的許可：指定使用期間內角色未使用的服務層級和動作層級許可。IAM Access Analyzer 會使用連接至角色的身分型政策，來判斷這些角色可存取的服務和動作。IAM Access Analyzer 支援檢閱所有服務層級許可的未使用許可。如需未使用的存取權調查結果支援的動作層級許可完整清單，請參閱：IAM 動作最近存取的資訊服務和動作。

> **Note**
>
> IAM Access Analyzer 免費提供外部存取權調查結果，且會根據每月每個分析器所分析的 IAM 角色和使用者數量，收取未使用的存取權調查結果費用。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

主題

- IAM Access AnalyAnalyzer 問題清單如何運作
- AWS Identity and Access Management Access Analyzer 問題清單入門
- 檢視 IAM Access Analyzer 調查結果儀表板
- 使用問題清單
- 檢閱問題清單
- 篩選問題清單
- 存檔問題清單
- 解決問題清單

- 外部存取權適用的 IAM Access Analyzer 資源類型
- IAM Access Analyzer 的設定
- 封存規則
- AWS Identity and Access Management Access Analyzer 使用 Amazon 監控 EventBridge
- 整合存取分析器 AWS Security Hub
- 使用記錄 IAM 存取分析器 API 呼叫 AWS CloudTrail
- IAM Access Analyzer 篩選鍵
- 使用 AWS Identity and Access Management Access Analyzer的服務連結角色

IAM Access AnalyAnalyzer 問題清單如何運作

本主題說明 IAM 存取分析器中使用的概念和術語，以協助您熟悉 IAM 存取分析器如何監控 AWS 資源的存取權。

外部存取權

對於外部訪問分析儀， AWS Identity and Access Management Access Analyzer 建立在 Zelkova 上，它將 IAM 政策轉換為等效的邏輯語句，並針對問題運行一套通用和專門的邏輯求解器（可滿足性模塊理論）。IAM Access Analyzer 會將 Zelkova 重複套用至具有越來越特定查詢的政策，以根據政策的內容來描述政策允許的行為類別特徵。若要進一步了解可滿足性模數理論的資訊，請參閱可滿足性模數理論。

針對外部存取權分析器，IAM Access Analyzer 不會檢查存取日誌，來判斷外部實體是否存取過您信任區域內的資源。當以資源為基礎的政策允許某資源的存取權時，即使外部實體未存取該資源，也會產生問題清單。IAM Access Analyzer 在作出判斷時也不考慮任何外部帳戶的狀態。也就是說，如果它指出帳戶 111122223333 可以存取您的 Amazon S3 儲存貯體，表示它對該帳戶中的使用者、角色、服務控制政策 (SCP) 和其他相關組態的狀態一無所知。這是考量到客戶的隱私權 – IAM Access Analyzer 不會考慮擁有其他帳戶的人員。這也考量到安全性 – 如果 IAM Access Analyzer 客戶不具有該帳戶，即使帳戶中目前沒有可存取那些資源的主體，您仍必須了解外部實體是否能夠獲得那些資源的存取權。

IAM Access Analyzer 僅考慮外部使用者無法直接影響或對授權具有影響力的某些 IAM 條件金鑰。如需條件金鑰 IAM Access Analyzer 考慮的範例，請參閱 IAM Access Analyzer 篩選金鑰。

IAM 存取分析器目前不會報告來自 AWS 服務主體或內部服務帳戶的發現結果。在 IAM Access Analyzer 無法完全判斷政策陳述式是否會將存取權授與外部實體的極少數情況下，則會在宣告誤報問

題清單時發生錯誤。IAM Access Analyzer 旨在提供在您帳戶中共享的全方位資源檢視，並努力將假否定降到最低。

未使用的存取權

即使您已經建立分析器來為資源產生外部存取權調查結果，還是須為角色建立未使用的存取權調查結果分析器。建立分析器之後，IAM Access Analyzer 會檢閱存取權活動，以識別未使用的存取權。IAM Access Analyzer 會檢閱 AWS 組織和帳戶中所有角色、使用者存取金鑰和使用者密碼的上次存取資訊，以協助您識別未使用的存取權限。對於有效 IAM 角色和使用者，IAM Access Analyzer 會使用 IAM 服務和動作上次存取資訊來識別未使用的許可。您可以使用未使用的存取分析器，在 AWS 組織和帳戶層級調整檢閱程序。您可以使用動作上次存取資訊，對個別角色進行更深入的調查。

「摘要」儀表板

針對外部和未使用的存取權，IAM Access Analyzer 會在摘要儀表板中整理歸納調查結果。針對外部存取權，摘要儀表板會反白公有與跨帳戶存取權調查結果之間的分隔線，並提供依資源類型分類的調查結果明細。對於未使用的存取，儀表板會反白顯示發現項目最多的您 AWS 帳戶 ，並依類型提供發現項目的明細。為外部或未使用的存取權建立分析器之後，IAM Access Analyzer 會在強調具有未使用許可的角色之儀表板中，自動新增新的調查結果。

AWS Identity and Access Management Access Analyzer 問題清單入門

使用本主題中的資訊來了解使用和管理所需的需求 AWS Identity and Access Management Access Analyzer，以及如何啟用 IAM 存取分析器。若要進一步了解 IAM Access Analyzer 的服務連結角色，請參閱 使用 AWS Identity and Access Management Access Analyzer的服務連結角色。

使用 IAM Access Analyzer 的必要許可

若要成功設定和使用 IAM Access Analyzer，您使用的帳戶必須獲得必要的許可。

AWS IAM 存取分析器的受管政策

AWS Identity and Access Management Access Analyzer 提供 AWS 受管理的原則，協助您快速開始使用。

- IAM AccessAnalyzer FullAccess-允許管理員完全訪問 IAM 訪問分析器。此政策也允許建立必要的服務連結角色，讓 IAM Access Analyzer 分析您帳戶或 AWS 組織中的資源。
- IAM AccessAnalyzer ReadOnly 存取-允許對 IAM 存取分析器進行唯讀存取。您必須將其他政策新增到 IAM 身分 (使用者、使用者群組或角色)，以允許他們檢視其問題清單。

IAM Access Analyzer 定義的資源

若要檢視 IAM Access Analyzer 定義的資源，請參閱服務授權參考中的 IAM Access Analyzer 定義的資源類型。

必要的 IAM Access Analyzer 服務許可

IAM Access Analyzer 會使用名為 `AWSServiceRoleForAccessAnalyzer` 的服務連結角色 (SLR)。此 SLR 授予服務唯讀存取權，以利用以 AWS 資源為基礎的政策分析資源，並代表您分析未使用的存取。此服務會在以下情境中為您的帳戶建立角色：

- 您以自己的帳戶為信任區域建立外部存取權分析器。
- 您以自己的帳戶為選定帳戶來建立未使用的存取權分析器。

如需詳細資訊，請參閱 使用 AWS Identity and Access Management Access Analyzer的服務連結角色。

> **Note**
>
> IAM Access Analyzer 是區域性的。針對外部存取權，您必須在每個區域中分別啟用 IAM Access Analyzer。
> 針對未使用的存取權，分析器的調查結果不會因區域而變更。不需要在每個您擁有資源的區域中都建立分析器。

某些情況下，在 IAM Access Analyzer 中建立外部存取權或未使用的存取權分析器後，系統會載入調查結果頁面或儀表板，但不含任何調查結果或摘要。這可能是因為主控台在填入您的問題清單時出現延遲。您可能須手動重新整理瀏覽器，或稍後再回來檢視調查結果或摘要。若仍沒有看到外部存取權分析器的調查結果，是因為您的帳戶沒有外部實體可存取的支援資源。若資源套用的政策會將存取權授與外部實體，則 IAM Access Analyzer 會產生問題清單。

> **Note**
>
> 針對外部存取權分析器，修改政策後，IAM Access Analyzer 最多可能需要 30 分鐘才能分析資源並產生新的調查結果，或者更新資源存取權的現有調查結果。針對外部和未使用的存取權分析器，調查結果的更新內容可能不會立即反映在儀表板中。

檢視調查結果儀表板所需的 IAM Access Analyzer 許可

若要檢視 IAM Access Analyzer 調查結果儀告板，您使用的帳戶必須擁有存取權，才能執行以下必要動作：

- `GetAnalyzer`
- `ListAnalyzers`
- `GetFindingsStatistics`

若要檢視 IAM Access Analyzer 定義的所有動作，請參閱服務授權參考中的 IAM Access Analyzer 定義的動作。

啟用 IAM Access Analyzer

建立以 AWS 帳戶 做為信任區域的外部存取分析器

若要在某區域內啟用外部存取權分析器，您必須在該區域建立分析器。您想監控資源存取權的每個區域，都必須建立外部存取權分析器。

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access analyzer (存取分析器)。
3. 選擇分析器設定。
4. 選擇 Create analyzer (建立分析器)。
5. 在分析區段中，選擇外部存取權分析。
6. 在分析器詳細資訊區段中，確認顯示的區域是您要啟用 IAM Access Analyzer 的區域。
7. 輸入分析器的名稱。
8. 選擇目前的 AWS 帳戶做為分析器的信任區域。

> **Note**
>
> 如果您的帳戶不是 AWS Organizations 管理帳戶或委派的系統管理員帳戶，您只能建立一個以您的帳戶做為信任區域的分析器。

9. 選用。新增您要套用至分析器的標籤。
10. 選擇提交。

建立外部存取權分析器來啟用 IAM Access Analyzer 時，您的帳戶會建立名為 `AWSServiceRoleForAccessAnalyzer` 的服務連結角色。

以組織為信任區域建立外部存取權分析器

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access analyzer (存取分析器)。
3. 選擇分析器設定。
4. 選擇 Create analyzer (建立分析器)。
5. 在分析區段中，選擇外部存取權分析。
6. 在分析器詳細資訊區段中，確認顯示的區域是您要啟用 IAM Access Analyzer 的區域。
7. 輸入分析器的名稱。
8. 選擇目前組織做為分析器的信任區域。
9. 選用。新增您要套用至分析器的標籤。
10. 選擇提交。

當您建立以組織做為信任區域的外部存取權分析器時，組織的每個帳戶都會建立名為 `AWSServiceRoleForAccessAnalyzer` 的服務連結角色。

為目前帳戶建立未使用的存取權分析器

使用下列程序，為單一 AWS 帳戶建立未使用的存取權分析器。針對未使用的存取權，分析器的調查結果不會因區域而變更。不需要在每個您擁有資源的區域中都建立分析器。

IAM Access Analyzer 會根據每月每個分析器分析的 IAM 角色和使用者數量，針對未使用的存取權分析收費。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access analyzer (存取分析器)。
3. 選擇分析器設定。
4. 選擇 Create analyzer (建立分析器)。
5. 在分析區段中，選擇未使用的存取權分析。
6. 輸入分析器的名稱。

7. 在追蹤期間輸入要針對未使用的許可產生調查結果的天數。例如，如果您輸入 90 天，分析器就會針對分析器上次掃描後 90 天或更長時間內任何未使用的許可，為選定帳戶內的 IAM 實體產生調查結果。可選擇 1 到 180 天之間的值。
8. 在選定帳戶選擇目前的 AWS 帳戶

> **Note**
>
> 如果您的帳戶不是 AWS Organizations 管理帳戶或委派管理員帳戶，您只能以您的帳戶建立一個分析器作為選取的帳戶。

9. 選用。新增您要套用至分析器的標籤。
10. 選擇提交。

建立未使用的存取權分析器來啟用 IAM Access Analyzer 時，您的帳戶會建立名為 `AWSServiceRoleForAccessAnalyzer` 的服務連結角色。

使用目前組織建立未使用的存取權分析器

使用下列程序為組織建立未使用的存取分析器，以集中檢閱組織 AWS 帳戶 中的所有內容。針對未使用的存取權分析，分析器的調查結果不會因區域而變更。不需要在每個您擁有資源的區域中都建立分析器。

IAM Access Analyzer 會根據每月每個分析器分析的 IAM 角色和使用者數量，針對未使用的存取權分析收費。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

> **Note**
>
> 如果從組織中移除成員帳戶，未使用的存取權分析器將在 24 小時後停止產生新的調查結果，並更新該帳戶的現有調查結果。與從組織中移除的成員帳戶相關聯的調查結果，將在 90 天後永久移除。

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access analyzer (存取分析器)。
3. 選擇分析器設定。
4. 選擇 Create analyzer (建立分析器)。
5. 在分析區段中，選擇未使用的存取權分析。

6. 輸入分析器的名稱。
7. 在追蹤期間輸入要針對未使用的許可產生調查結果的天數。例如，如果您輸入 90 天，分析器就會針對分析器上次掃描後 90 天或更長時間內任何未使用的許可，為選定組織中所有帳戶的 IAM 實體產生調查結果。可選擇 1 到 180 天之間的值。
8. 在選定帳戶，選擇目前組織做為分析器的選定帳戶。
9. 選用。新增您要套用至分析器的標籤。
10. 選擇提交。

建立未使用的存取權分析器來啟用 IAM Access Analyzer 時，您的帳戶會建立名為 `AWSServiceRoleForAccessAnalyzer` 的服務連結角色。

IAM Access Analyzer 狀態

若要檢視分析器的狀態，請選擇 Analyzers (分析器)。為組織或帳戶建立的分析器可能具備下列狀態：

狀態	描述
作用中	外部存取權分析器會主動監控信任區域內的資源。分析器會主動產生新的問題清單，並更新現有的問題清單。 對於未使用的存取分析器，分析器會主動監視所選組織內或指定追蹤期間 AWS 帳戶 內未使用的存取。分析器會主動產生新的問題清單，並更新現有的問題清單。
正在建立	分析器仍在建立中。建立完成後，分析器就會變成作用中狀態。
已停用	由於管理員採取的處 AWS Organizations 理行動，分析器已停用。例如，移除身分為 IAM Access Analyzer 委派管理員的分析器帳戶。當分析器處於停用狀態時，不會產生新的調查結果或更新現有的調查結果。

狀態	描述
失敗	由於組態發生問題，分析器建立失敗。分析器不會產生任何問題清單。請刪除該分析器並建立新的分析器。

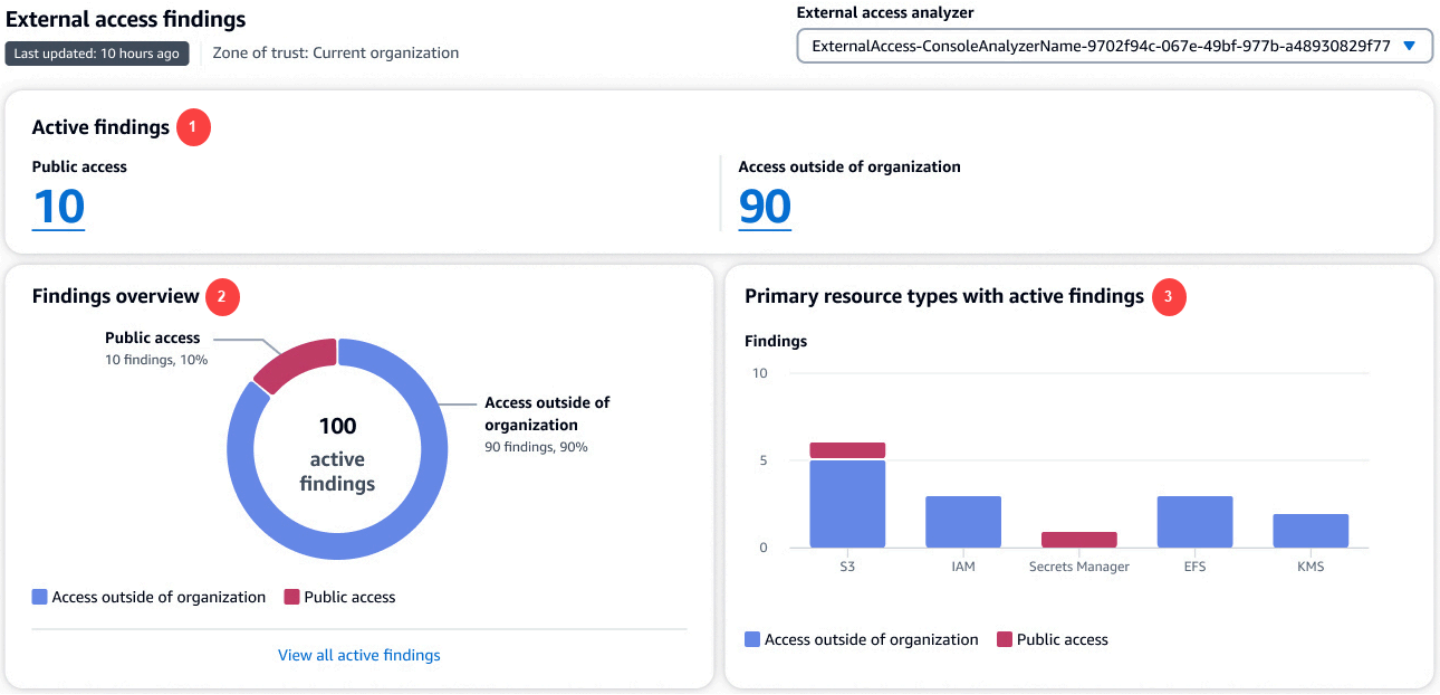
檢視 IAM Access Analyzer 調查結果儀表板

AWS Identity and Access Management Access Analyzer 將外部存取和未使用的存取發現項目組織到視覺化摘要儀表板中。儀表板可協助您大規模了解有效的使用許可，並識別需要注意的帳戶。您可以使用儀表板，依 AWS 組織、帳戶及搜尋結果類型來複查搜尋結果。

檢視外部存取權分析器摘要儀表板

> **Note**
>
> 建立或更新分析器之後，摘要儀表板可能需要一些時間才會反映調查結果的更新內容。

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access analyzer (存取分析器)。隨即會顯示摘要視窗。
3. 在外部存取權分析器下拉式清單中選擇分析器。分析器的調查結果摘要會顯示在外部存取權調查結果區段中。



在上圖中，外部存取權調查結果儀表板顯示在摘要頁面中：

1. 作用中調查結果區段包括供公開存取權的作用中調查結果數，以及提供帳戶或組織外部存取權的作用中調查結果數。選擇一個數字即可列出每個類型的所有作用中調查結果。
2. 調查結果概觀區段包含作用中調查結果類型的明細。選擇檢視所有作用中調查結果，以取得分析器帳戶或組織的作用中調查結果完整清單。
3. 具有作用中調查結果的主要資源類型區段中，包含具有作用中調查結果的主要資源類型明細。這些資訊可協助您決定主要資源調查結果的優先順序。例如，Amazon S3、和. AWS KMS這並非每種資源類型的詳盡清單。您的分析器作用中調查結果可能包含本區段沒有列出的資源類型。

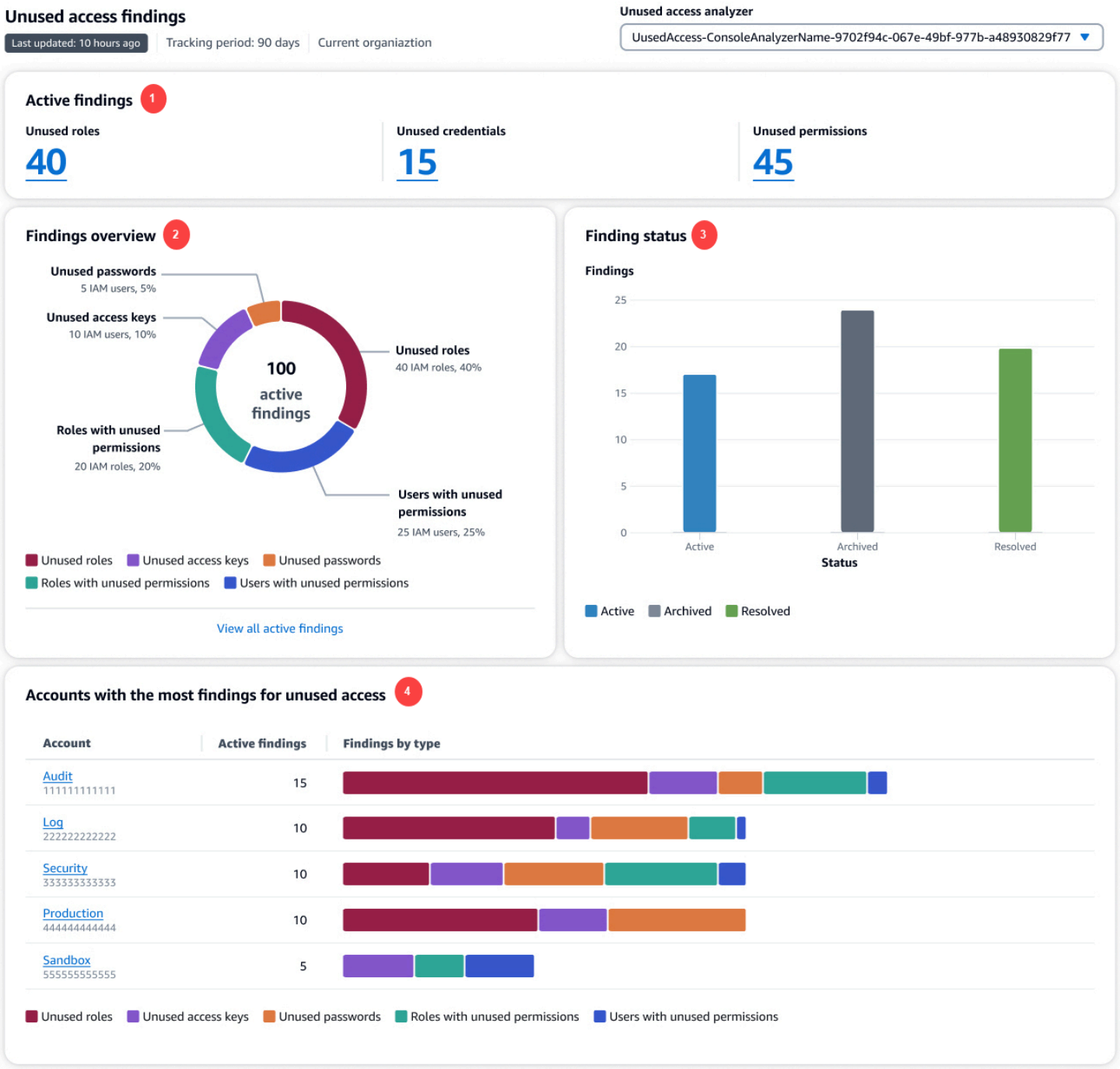
檢視未使用的存取權分析器摘要儀表板

IAM Access Analyzer 會根據每月分析的 IAM 角色和使用者數量，針對未使用的存取權分析收費。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

> **Note**
>
> 根據使用者和角色的數量建立或更新分析器之後，摘要儀表板可能需要一些時間才會反映調查結果的更新內容。

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

2. 選擇 Access analyzer (存取分析器)。隨即會顯示摘要視窗。
3. 在未使用的存取權分析器下拉式清單中選擇分析器。分析器的調查結果摘要會顯示在未使用的存取權調查結果區段中。



在上圖中，外部存取權調查結果儀表板顯示在摘要頁面中：

1. 作用中調查結果區段中，包含帳戶或組織中未使用角色、未使用憑證和未使用的許可作用中調查結果數。未使用的憑證包括未使用的存取金鑰和未使用的密碼調查結果。未使用的許可包括具有未使用許可的使用者和角色。選擇一個數字即可列出每個類型的所有作用中調查結果。
2. 調查結果概觀區段包含作用中調查結果類型的明細。選擇檢視所有作用中調查結果，以取得分析器帳戶或組織的作用中調查結果完整清單。
3. 調查結果狀態區段包含您帳戶或組織的調查結果狀態明細 (作用中、已封存及已解決)。
4. 只有在未使用的存取權分析器的選定帳戶為組織層級時，才會顯示具有最多未使用存取權調查結果的帳戶區段。此區段包含組織中具有最多作用中調查結果的帳戶明細。這並非貴組織中每個帳戶的詳盡清單。您的分析器作用中調查結果可能包含本區段沒有列出的其他帳戶。

使用問題清單

外部存取權調查結果

若某資源在您的信任區域之外共用，其每個執行個體都會產生一份外部存取權調查結果。每次修改資源型政策後，IAM Access Analyzer 都會分析該政策。若更新後政策所共用的資源已被問題清單辨識，但具備不同許可或條件，則該資源共用的執行個體會產生新的問題清單。若第一份調查結果的存取權已移除，該調查結果的狀態會更新為已解決。

所有調查結果的狀態會保持為作用中，直到您將其封存，或者移除產生該調查結果的存取權。移除存取權時，調查結果狀態會更新為已解決。

> **Note**
>
> 修改政策後，IAM Access Analyzer 最多可能需要 30 分鐘才能分析資源並更新外部存取權調查結果。

未使用的存取權調查結果

系統會根據建立分析器時指定的天數，針對選定帳戶或組織內的 IAM 實體產生未使用的存取權調查結果。如果符合下列其中一個條件，分析器下次掃描實體時就會產生新的調查結果：

- 角色在指定天數內處於非作用狀態。
- 未使用的許可、未使用的使用者密碼或未使用的使用者存取金鑰超過指定天數。

您應檢閱帳戶內的所有調查結果，藉以判定外部或未使用的存取權是否正常和經過核准。若調查結果識別的外部或未使用的存取權是正常的，則可將該調查結果封存。封存的調查結果狀態會變更為已封存，並從作用中調查結果清單中移除。但不會刪除該問題清單。您可隨時檢視已存檔的問題清單。請仔細閱讀帳戶內的所有問題清單，直到作用中問題清單數量為零。調查結果數量變成零後，即可知道任何新的作用中調查結果，都是環境內最近的變更所產生。

> **Note**
>
> 未使用的存取發現項目只能使用 ListFindingsV2 API 動作使用。

檢閱問題清單

啟用 IAM Access Analyzer 後，下一個步驟是檢閱問題清單，藉以判定其中的存取權為預期之內或之外。您也可檢閱調查結果來決定類似的存取權調查結果是否在預期之內，之後即可建立封存規則，自動將這些調查結果封存。您也可檢閱已存檔與已解決的問題清單。

檢閱問題清單

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access analyzer (存取分析器)。
3. 此時會顯示調查結果儀表板。為外部或未使用的存取權分析器選取作用中的調查結果。

   如需檢視調查結果儀表板的詳細資訊，請參閱：檢視 IAM Access Analyzer 調查結果儀表板。

> **Note**
>
> 只有在您擁有分析器問題清單的檢視許可時，才會顯示問題清單。

此時會顯示該分析器的所有調查結果。若要檢視該分析器產生的其他調查結果，請從狀態下拉是清單中選擇適當的調查結果類型：

- 選擇 Active (作用中) 以檢視所有該分析器產生的作用中問題清單。
- 若選擇 Archived (已存檔)，則只能檢視該分析器已存檔的問題清單。如需進一步了解，請參閱存檔問題清單。
- 若選擇 Resolved (已解決)，則只能檢視該分析器已解決的問題清單。修復產生調查結果的問題時，該調查結果狀態會變更為已解決。

> **Important**
>
> 已解決的問題清單會在其最後一次更新的 90 天後刪除。作用中和已存檔的問題清單都不會被刪除，除非您將產生該問題清單的分析器刪除。

- 若選擇 All (全部)，即可檢視該分析器產生的所有問題清單 (不論狀態)。

外部存取權調查結果

選擇外部存取權，然後從檢視分析器下拉式清單中選擇外部存取權分析器。外部存取權分析器的調查結果頁面顯示的詳細資訊，為產生問題之共用資源與政策陳述式，如下所示：

問題清單 ID

指派給問題清單的唯一 ID。選擇問題清單 ID 來顯示產生問題之資源與政策陳述式的其他詳細資訊。

Resource

資源類型與部分名稱，該資源套用的政策會將存取權授予您信任區域以外的外部實體。

Resource owner account (資源擁有者帳戶)

只有在您使用組織做為信任區域時，此欄才會顯示。問題清單會回報組織中擁有該資源的帳戶。

External principal (外部主體)

您信任區域以外的主體，受分析之政策授予存取權的對象。有效值包含：

- AWS 帳戶：列出的 AWS 帳戶 中，獲得帳戶管理員許可的所有主體均可存取該資源。
- 任何主參與者 — 符合「條件」欄中包含之條件的任何 AWS 帳戶 主參與者中的所有主參與者都有存取資源的權限。例如，若列出 VPC，這代表任何帳戶內能夠存取該 VPC 的所有主體，均可存取該資源。
- 標準使用者 — 具有列出的規範使用者 ID 中的所有主參 AWS 帳戶 與者都有存取資源的權限。
- IAM role (IAM 角色) – 所列出的 IAM 角色均擁有該資源的存取許可。
- IAM user (IAM 使用者) – 所列出的 IAM 使用者均擁有該資源的存取許可。

Condition

政策陳述式內授予存取權的條件。例如，若 Condition (條件) 欄位內含 Source VPC (來源 VPC)，這代表該資源共用的對象為可存取上列 VPC 的主體。條件可為全域或服務特定。全域條件金鑰字首為 `aws:`。

Shared through (共用方式)

Shared through (共用方式) 欄位會指出產生問題清單之存取權的授予方式。有效值包含：

- 儲存貯體政策 – 連接至 Amazon S3 儲存貯體的儲存貯體政策。
- Access control list (存取控制清單) – 連接到 Amazon S3 儲存貯體的存取控制清單 (ACL)。
- Access point (存取點) – 與 Amazon S3 儲存貯體相關聯的存取點或多區域存取點。存取點的 ARN 會顯示在 Findings (問題清單) 詳細資訊中。

存取層級

資源型政策的動作授予外部實體之存取層級。如需詳細資訊，請檢視問題清單的詳細資訊。存取層級的值如下：

- List (清單) – 列出服務內資源的許可，以判斷物件是否存在。具有此層級存取權的動作，可以列出物件，但無法查看資源的內容。
- Read (讀取) – 可讀取但無法編輯服務內資源的內容和屬性的許可。
- Write (寫入) – 可建立、刪除或修改服務內資源的許可。
- Permissions (許可) – 可授與或修改服務內資源許可的許可。
- Tagging (標記) – 執行僅變更資源標籤狀態之動作的許可。

Updated

問題清單狀態最近一次更新的時間戳記，若未有更新，則顯示問題清單產生的時間與日期。

> **Note**
>
> 修改政策後，IAM Access Analyzer 至多可能需要 30 分鐘才能再次分析資源並更新問題清單。

狀態

問題清單的狀態為 Active (作用中)、Archived (已存檔) 或 Resolved (已解決)。

未使用的存取權調查結果

IAM Access Analyzer 會根據每月分析的 IAM 角色和使用者數量，針對未使用的存取權分析收費。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

選擇未使用的存取權，然後從檢視分析器下拉式清單中選擇未使用的存取權分析器。未使用的存取權分析器調查結果頁面顯示的詳細資訊，為有關產生調查結果的 IAM 實體，如下所示：

問題清單 ID

指派給問題清單的唯一 ID。選擇調查結果 ID 來顯示產生調查結果之 IAM 實體的其他詳細資訊。

調查結果類型

未使用的存取權調查結果類型：未使用的存取金鑰、未使用的密碼、未使用的許可或未使用的角色。

IAM 實體

調查結果中報告的 IAM 實體。可以是 IAM 使用者或角色。

AWS 帳戶 ID

只有當您為組織 AWS 帳戶 中的所有人設定分析器時，才會顯示此欄位。擁有發現項目 AWS 帳戶 中報告之 IAM 實體的組織中。

上次更新

上次更新調查結果中所報告 IAM 實體的時間，或者如果未進行更新，則為實體的建立時間。

狀態

調查結果的狀態為作用中、已封存或已解決。

篩選問題清單

調查結果頁面的預設篩選結果會顯示所有調查結果。若要檢視作用中調查結果，請從狀態下拉式清單選擇作用中。若要檢視已封存的調查結果，請從狀態下拉式清單中選擇已封存。首次使用 IAM Access Analyzer 不會有已存檔的問題清單。

使用篩選器只顯示符合指定屬性條件的調查結果。若要建立篩選條件，請選取要篩選的屬性，然後選擇屬性是等於還是包含值，然後輸入或選擇要篩選的屬性值。例如，若要建立只顯示特定發現項目的篩選器 AWS 帳戶，請選擇「AWS 帳戶」做為屬性，然後選擇「AWS 帳戶 =」，然後輸入您 AWS 帳戶 要檢視其發現項目的帳號。

如需可用來建立或更新存檔規則的篩選鍵清單，請參閱 IAM Access Analyzer 篩選鍵。

篩選外部存取權調查結果

篩選外部存取權調查結果

1. 選擇外部存取權，然後從檢視分析器下拉式清單中選擇分析器。
2. 選擇搜尋方塊以顯示可用屬性清單。
3. 選擇要用來篩選所顯示之問題清單的屬性。
4. 選擇該屬性須符合的值。將會顯示具備該值的問題清單。

   例如，屬性選擇資源，選擇資源:，接著輸入某儲存貯體的部分或完整名稱，然後按 Enter 鍵。僅顯示符合篩選條件的儲存貯體相關調查結果。若要建立僅顯示允許公開存取權之資源的調查結果，您可以選擇公開存取權屬性，選擇公開存取權 =，然後選擇公開存取權 = true。

您可新增其他屬性，進一步篩選所顯示的問題清單。新增其他屬性時，只會顯示符合所有篩選條件的問題清單。問題清單的篩選條件，不支援符合單一屬性「或」另一屬性的這類定義。選擇清除篩選條件以清除已定義的任何篩選條件，並顯示具有指定分析器狀態的所有調查結果。

只有在您檢視以組織做為信任區域的分析器問題清單時，某些欄位才會顯示。

下列屬性可用於定義篩選條件：

- Public access (公有存取) – 若要依允許公有存取之資源的問題清單篩選，請依 Public access (公有存取) 篩選，然後選擇 Public access: true (公有存取：true)。
- Resource (資源) – 輸入資源的完整或部分名稱，即可依資源篩選。
- Resource Type (資源類型) – 從所顯示的清單內選擇類型，即可依資源類型篩選。
- 資源擁有者帳戶：使用此屬性可篩選組織中擁有調查結果所報告資源的帳戶。
- AWS 帳戶 — 使用此內容可篩選原則 AWS 帳戶 陳述式之 [主參與者] 區段中已授與存取權的項目。若要篩選條件 AWS 帳戶，請輸入全部或部分 12 位數 AWS 帳戶 ID，或是外部 AWS 使用者或角色 (可存取目前帳號中資源) 的全部或部分帳號 ARN。
- 正式使用者：輸入為 Amazon S3 儲存貯體定義的正式使用者 ID，即可依正式使用者篩選。若要進一步了解，請參閱 AWS 帳戶識別碼。
- Federated User (聯合身分使用者) – 輸入聯合身分的完整或部分 ARN，即可依聯合身分使用者篩選。若要進一步了解，請參閱身分提供者與聯合。
- 調查結果 ID：請輸入全部或部分調查結果 ID，即可依調查結果 ID 篩選。

- 主體 ARN — 使用此屬性可篩選 aws: PrincipalArn 條件金鑰中使用的主體 (IAM 使用者、角色或群組) 的 ARN。若要依主體 ARN 進行篩選，請從發現項目中 AWS 帳戶 報告的外部輸入 IAM 使用者、角色或群組的全部或部分 ARN。
- Principal OrgID (主體組織 ID) – 針對問題清單列為條件之 AWS 組織的外部主體，輸入與其相關聯的完整或部分組織 ID，即可依主體組織 ID 篩選。若要進一步了解，請參閱 AWS 全域條件內容鍵。
- 主參與者 OrgPaths — 若要依主參與者進行篩選 OrgPaths，請輸入組織或組織單位 (OU) 的全部或部分 ID，以存取屬於指定組織或 OU 帳戶成員的所有外部主參與者，作為策略中的條件。 AWS 若要進一步了解，請參閱 AWS 全域條件內容鍵。
- 來源帳戶 — 若要篩選來源帳戶，請輸入與資源相關聯的全部或部分 AWS 帳戶 ID，如中某些跨服務權限中 AWS所使用的那樣。若要進一步了解，請參閱 AWS 全域條件內容鍵。
- Source ARN (來源 ARN) – 輸入問題清單內列為條件的完整或部分 ARN，即可依來源 ARN 篩選。若要進一步了解，請參閱 AWS 全域條件內容鍵。
- Source IP (來源 IP) – 輸入允許外部實體在使用特定 IP 地址時即可存取目前帳戶內資源的完整或部分 IP 地址，即可依來源 IP 篩選。若要進一步了解，請參閱 AWS 全域條件內容鍵。
- Source VPC (來源 VPC) – 輸入允許外部實體在使用特定 VPC 時即可存取目前帳戶內資源的完整或部分 VPC ID，即可依來源 VPC 篩選。若要進一步了解，請參閱 AWS 全域條件內容鍵。
- 來源 OrgID — 若要依來源 OrgID 進行篩選，請輸入與資源相關聯的全部或部分組織 ID (如中某些跨服務權限中所使用)。 AWS若要進一步了解，請參閱 AWS 全域條件內容鍵。
- 來源 OrgPaths — 若要依來源進行篩選 OrgPaths，請輸入與資源相關聯的全部或部分組織單位 (OU)，如中某些跨服務權限所用。 AWS若要進一步了解，請參閱 AWS 全域條件內容鍵。
- 使用者 ID — 若要依使用者 ID 進行篩選，請輸入允許存取目前帳戶資源的外部 AWS 帳戶 IAM 使用者的全部或部分使用者 ID。若要進一步了解，請參閱 AWS 全域條件內容鍵。
- KMS 金鑰識別碼 — 若要依 KMS 金鑰識別碼進行篩選，請輸入指定為目前帳戶中 AWS KMS加密 Amazon S3 物件存取條件的 KMS 金鑰的全部或部分金鑰識別碼。
- Google Audience (Google 對象) – 輸入目前帳戶內 IAM 角色存取指定為條件的完整或部分 Google 應用程式 ID，即可依 Google 對象篩選。若要進一步了解，請參閱 IAM 和 AWS STS 條件內容金鑰。
- Cognito 對象：輸入目前帳戶內 IAM 角色存取指定為條件的完整或部分 Amazon Cognito 身分池 ID，即可依 Cognito 對象篩選。若要進一步了解，請參閱 IAM 和 AWS STS 條件內容金鑰。
- 來電者帳戶 — 擁有或包含呼叫實體的帳戶 AWS 帳戶 ID，例如 IAM 角色、使用者或帳戶根使用者。這是由服務調用使用 AWS KMS。輸入 AWS 帳戶 的完整或部分 ID，即可依呼叫者帳戶篩選。
- Facebook App ID (Facebook 應用程式 ID) – 關於允許存取目前帳戶內 IAM 角色的 Login with Facebook (使用 Facebook 登入) 聯合身分，輸入列為其條件的完整或部分 Facebook 應用程式 ID

(或網站 ID)，即可依 Facebook 應用程式 ID 篩選。若要進一步了解，請參閱《IAM 和 AWS STS 條件內容金鑰》中的 id 部分。

- Amazon App ID (Amazon 應用程式 ID) – 關於允許存取目前帳戶內 IAM 角色的 Login with Amazon 聯合身分，輸入列為其條件的完整或部分 Amazon 應用程式 ID (或網站 ID)，即可依 Amazon 應用程式 ID 篩選。若要進一步了解，請參閱《IAM 和 AWS STS 條件內容金鑰》中的 id 部分。
- Lambda Event Source Token (Lambda 事件來源字符) – 輸入完整或部分字符字串，即可依使用 Alexa 整合傳入的 Lambda 事件來源字符篩選。

篩選未使用的存取權

篩選未使用的存取權調查結果

1. 選擇未使用的存取權，然後從檢視分析器下拉式清單中選擇分析器。
2. 選擇搜尋方塊以顯示可用屬性清單。
3. 選擇要用來篩選所顯示之問題清單的屬性。
4. 選擇該屬性須符合的值。將會顯示具備該值的問題清單。

   例如，選擇「發現項目類型」作為特性，然後選擇「發現項目類型 =」，然後選擇「發現項目類型」= UnuseDiamRole，只會顯示類型為 un useDiamRole 的發現項目。

您可新增其他屬性，進一步篩選所顯示的問題清單。新增其他屬性時，只會顯示符合所有篩選條件的問題清單。問題清單的篩選條件，不支援符合單一屬性「或」另一屬性的這類定義。選擇清除篩選條件以清除已定義的任何篩選條件，並顯示具有指定分析器狀態的所有調查結果。

只有當您檢視監督未使用存取之分析器的發現項目時，才會顯示下列欄位：

- 發現項目類型 — 若要依尋找項目類型進行篩選，請依發現項目類型篩選，然後選擇發現項目類型。
- Resource (資源) – 輸入資源的完整或部分名稱，即可依資源篩選。
- Resource Type (資源類型) – 從所顯示的清單內選擇類型，即可依資源類型篩選。
- 資源擁有者帳戶：使用此屬性可篩選組織中擁有調查結果所報告資源的帳戶。
- 尋找項目 ID — 若要依尋找 ID 進行篩選，請輸入全部或部分尋找項目 ID。

存檔問題清單

如果調查結果中的資源存取權是有意圖的，就可以封存該調查結果。例如，指出多名使用者將某個 IAM 角色用於經核准的工作流程之外部存取權調查結果，或是指出某存取金鑰可能仍為必要的未使用存取權調查結果。當您封存調查結果時，該項目會從作用中調查結果清單中清除。存檔的問題清單不會刪除。您可篩選調查結果頁面來顯示已封存的調查結果，並隨時將其取消封存。

在 Findings (問題清單) 頁面將問題清單存檔

1. 選取要存檔之一個或多個問題清單旁的核取方塊。
2. 選擇動作，然後選擇封存。

   螢幕頂端會顯示確認訊息。

在調查結果詳細資訊頁面中封存調查結果。

1. 選擇欲存檔的 Finding ID (問題清單 ID)。

2. 選擇 Archive (封存)。

   螢幕頂端會顯示確認訊息。

若要取消問題清單的存檔，請重複上述步驟，但選擇 Unarchive (取消存檔)，不要選擇 Archive (存檔)。問題清單取消存檔後，狀態會變更為作用中。

解決問題清單

外部存取權調查結果

若您不打算允許某個存取權，且希望解決其產生的外部存取權調查結果，請修改政策陳述式，移除允許存取所識別資源的許可。例如，若 Amazon S3 儲存貯體出現調查結果，請使用 Amazon S3 主控台來設定該儲存貯體的許可。若是 IAM 角色，請使用 IAM 主控台來修改所列出之 IAM 角色的信任政策。關於其他支援的資源，請使用主控台來修改產生問題清單的政策陳述式。

進行變更來解決外部存取權調查結果後 (例如修正套用至 IAM 角色的政策)，IAM Access Analyzer 會再次掃描資源。若資源不再於信任區域外共用，則問題清單的狀態會變更為已解決。該調查結果不會再出現於作用中調查結果清單中，而是會顯示在已解決調查結果清單內。

> **Note**
>
> 上述內容不適用於錯誤調查結果。當 IAM Access Analyzer 無法分析資源時，就會產生錯誤問題清單。如果您解決了 IAM Access Analyzer 無法分析資源的問題，錯誤問題清單就會完全移除，而不是變更為已解決的問題清單。

若您進行的變更使得資源以另一種方式 (如不同主體或不同許可) 在信任區域外共用，則 IAM Access Analyzer 會產生新的作用中問題清單。

> **Note**
>
> 修改政策後，IAM Access Analyzer 至多可能需要 30 分鐘才能再次分析資源並更新問題清單。已解決的問題清單會在最後更新至問題清單狀態的 90 天後刪除。

未使用的存取權調查結果

針對未使用的存取分析器發現項目，IAM Access Analyzer 會根據發現項目類型提供建議的步驟來解決發現項目。

在您進行變更以解決未使用的存取權調查結果之後，下次執行未使用的存取權分析器時，調查結果的狀態會變更為已解決。發現項目不會再顯示在作用中的發現項目清單中，而是會顯示在已解析的發現項目清單中。如果您進行的變更僅部分解決未使用的存取權調查結果，現有的調查結果會變更為已解決，但會產生新的調查結果。例如，您只移除調查結果中部分未使用的許可，而非所有未使用的許可。

IAM Access Analyzer 會根據每月分析的 IAM 角色和使用者數量，針對未使用的存取權分析收費。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

解決未使用的權限發

對於未使用的權限發現，IAM Access Analyzer 可以建議從 IAM 使用者或角色移除政策，並提供新政策來取代現有許可政策。下列案例不支援原則建議：

- 未使用的權限尋找項目適用於使用者群組中的 IAM 使用者。
- 未使用的權限尋找項目適用於 IAM 身分中心的 IAM 角色。
- 未使用的權限尋找項目具有包含`notAction`元素的現有權限原則。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。

2. 選擇未使用的存取。
3. 選擇「尋找項目」類型為「未使用」權限的發現項目
4. 在 [建議] 段落中，如果 [建議的原則] 資料欄中列出原則，請選擇預覽原則，以建議的原則來檢視現有原則以取代現有原則的建議原則。如果有多個建議的原則，您可以選擇 [下一個原則] 和 [上一個原則] 來檢視每個現有和建議的原則。
5. 選擇 [下載 JSON] 以下載包含所有建議政策之 JSON 檔案的 .zip 檔案。
6. 建立建議的政策，並將其附加到 IAM 使用者或角色。如需詳細資訊，請參閱變更使用者的權限 (主控台) 和修改角色權限原則 (主控台)。
7. 從 IAM 使用者或角色移除 [現有許可政策] 欄中列出的政策。如需詳細資訊，請參閱移除使用者的權限 (主控台) 和修改角色權限原則 (主控台)。

解決未使用角色發現

對於未使用的角色發現，IAM 存取分析器建議刪除未使用的 IAM 角色。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇未使用的存取。
3. 選擇「搜尋結果」類型為「未使用」角色的搜尋結果
4. 在「建議」區段中，檢閱 IAM 角色的詳細資料。
5. 刪除 IAM 角色。如需詳細資訊，請參閱刪除 IAM 角色 (主控台)。

解決未使用的存取鍵發現

對於未使用的存取金鑰發現項目，IAM Access Analyzer 建議停用或刪除未使用的存取金鑰。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇未使用的存取。
3. 使用「尋找項目」類型的「未使用的存取金鑰」選擇一個
4. 在「建議」段落中，檢閱存取金鑰的詳細資訊。
5. 停用或刪除存取金鑰。如需詳細資訊，請參閱管理存取金鑰 (主控台)。

解決未使用密碼發現

針對未使用的密碼發現項目，IAM 存取分析器建議刪除 IAM 使用者未使用的密碼。

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇未使用的存取。
3. 選擇「尋找項目」類型為「未使用」密碼的搜尋結果
4. 在「建議」區段中，檢閱 IAM 使用者的詳細資料。
5. 刪除 IAM 使用者的密碼。如需詳細資訊，請參閱建立、變更或刪除 IAM 使用者密碼 (主控台)。

外部存取權適用的 IAM Access Analyzer 資源類型

對於外部存取分析儀，IAM Access Analyzer 會分析套用至您啟用 IAM 存取分析器之區域中 AWS 資源的資源型政策。系統只會分析資源型政策。檢閱每個資源的相關資訊，以取得 IAM Access Analyzer 如何為每個資源類型產生問題清單的相關詳細資訊。

> **Note**
>
> 列出的支援資源類型適用於外部存取權分析器。未使用的存取權分析器僅支援 IAM 使用者和角色。如需詳細資訊，請參閱 使用問題清單。

外部存取權支援的資源類型：

- Amazon Simple Storage Service 儲存貯體
- Amazon Simple Storage Service 目錄儲存貯體
- AWS Identity and Access Management 角色
- AWS Key Management Service 鑰匙
- AWS Lambda 函數和圖層
- Amazon Simple Queue Service 佇列
- AWS Secrets Manager 秘密
- Amazon Simple Notification Service 主題
- Amazon Elastic Block Store 磁碟區快照
- Amazon Relational Service 資料庫快照
- Amazon Relational Database Service 資料庫叢集快照
- Amazon Elastic Container Registry 儲存庫
- Amazon Elastic File System 檔案系統
- Amazon DynamoDB 串流

- Amazon DynamoDB 資料表

Amazon Simple Storage Service 儲存貯體

當 IAM Access Analyzer 分析 Amazon S3 儲存貯體時，若套用至儲存貯體的 Amazon S3 儲存貯體政策、ACL 或包括多區域存取點的存取點將存取權授與外部實體，則會產生問題清單。外部實體是您可用來建立不在信任區域內之篩選的委託人或其他實體。例如，如果儲存貯體政策對另一個帳戶授與存取權或允許公有存取權，則 IAM Access Analyzer 會產生問題清單。不過，如果您在儲存貯體上啟用 Block Public Access (封鎖公有存取權)，則可以在帳戶層級或儲存貯體層級封鎖存取權。

> **Note**
>
> IAM Access Analyzer 不會分析連接至跨帳戶存取點的存取點政策，因為存取點及其政策不在分析器帳戶以內。當儲存貯體委派對跨帳戶存取點的存取權，且儲存貯體或帳戶上未啟用「封鎖公開存取」時，IAM Access Analyzer 會產生公開問題清單。當您啟用「封鎖公開存取」時，公開問題清單得以解決，IAM Access Analyzer 將產生跨帳戶存取點的跨帳戶問題清單。

Amazon S3 封鎖公開存取設定會覆寫套用至儲存貯體的儲存貯體政策。這些設定也會覆寫套用至儲存貯體存取點的存取點政策。IAM Access Analyzer 會在政策變更時分析儲存貯體層級的「封鎖公開存取」設定。不過，它僅以 6 小時一次的頻率在帳戶層級評估「封鎖公開存取」設定。這意味著 IAM Access Analyzer 未對儲存貯體產生公有存取權問題清單或進行解決的時間可能長達 6 小時。例如，如果您有允許公有存取權的儲存貯體政策，則 IAM Access Analyzer 會產生該存取權的問題清單。如果您接著啟用「封鎖公開存取」以在帳戶層級封鎖對儲存貯體的所有公開存取權，即使對儲存貯體的所有公開存取權都遭到封鎖，IAM Access Analyzer 可能需要達 6 小時的時間來解決儲存貯體政策之問題清單。一旦您在帳戶層級啟用「封鎖公開存取」，解決跨帳戶存取點的公開問題清單最多也可能需要 6 小時。

對於多區域存取點，IAM Access Analyzer 會使用已建立的政策來產生問題清單。IAM Access Analyzer 每 6 小時會評估一次對多區域存取點的變更。這表示即使您建立或刪除多區域存取點，或更新其政策，IAM Access Analyzer 對此不產生或解決問題清單的時間可達最多 6 小時。

Amazon Simple Storage Service 目錄儲存貯體

Amazon S3 目錄儲存貯體使用 Amazon S3 Express One 儲存體方案，建議用於關鍵效能的工作負載或應用程式。針對 Amazon S3 目錄儲存貯體，IAM Access Analyzer 會分析允許外部實體存取目錄儲存貯體的目錄儲存貯體政策 (包括政策中的條件陳述式)。如需 Amazon S3 目錄儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的目錄儲存貯體。

AWS Identity and Access Management 角色

對於 IAM 角色，IAM Access Analyzer 會分析信任政策。在角色信任政策中，您會定義您信任能夠擔任角色的主體。角色信任政策是在 IAM 中連接至角色的以資源為基礎的必要政策。IAM Access Analyzer 會產生信任區域內角色的問題清單，該問題清單可供位於信任區域外的外部實體存取。

> **Note**
>
> IAM 角色是全域資源。如果角色信任政策授與外部實體存取權，則 IAM Access Analyzer 會在每個啟用的區域中產生問題清單。

AWS Key Management Service 鑰匙

針對 AWS KMS keys，IAM 存取分析器會分析金鑰套用的金鑰政策和授權。如果金鑰政策或授與允許外部實體存取金鑰，IAM Access Analyzer 會產生問題清單。例如，如果您在政策陳述式中使用 kms:CallerAccount 條件金鑰來允許特定 AWS 帳戶中的所有使用者存取，並且指定目前帳戶以外的帳戶 (目前分析器的信任區域)，IAM Access Analyzer 就會產生一個發現項目。若要進一步了解 IAM 政策陳述式中的 AWS KMS 條件金鑰，請參閱AWS KMS 條件金鑰。

IAM Access Analyzer 分析 KMS 金鑰時，它會讀取金鑰中繼資料，例如金鑰政策和授與清單。如果金鑰政策不允許 IAM Access Analyzer 角色讀取金鑰中繼資料，就會產生 Access Denied (存取遭拒) 的錯誤問題清單。例如，若下列範例政策陳述式是金鑰中套用的唯一政策，則會在 IAM Access Analyzer 中產生 Access denied (存取遭拒) 的錯誤問題清單。

```
{
    "Sid": "Allow access for Key Administrators",
    "Effect": "Allow",
    "Principal": {
       "AWS": "arn:aws:iam::111122223333:role/Admin"
    },
    "Action": "kms:*",
    "Resource": "*"
}
```

由於此陳述式只允許 AWS 帳戶 111122223333 中名為 Admin 的角色存取金鑰，因此會產生「拒絕存取」錯誤發現，因為 IAM 存取分析器無法完整分析金鑰。錯誤問題清單在 Findings (問題清單) 資料表中會以紅色文字顯示。問題清單看起來類似下列。

```
{
```

```
    "error": "ACCESS_DENIED",
    "id": "12345678-1234-abcd-dcba-111122223333",
    "analyzedAt": "2019-09-16T14:24:33.352Z",
    "resource": "arn:aws:kms:us-
west-2:1234567890:key/1a2b3c4d-5e6f-7a8b-9c0d-1a2b3c4d5e6f7g8a",
    "resourceType": "AWS::KMS::Key",
    "status": "ACTIVE",
    "updatedAt": "2019-09-16T14:24:33.352Z"
}
```

當您建立 KMS 金鑰時，所授與可存取金鑰的許可取決於金鑰的建立方式。如果您收到金鑰資源的 Access Denied (存取遭拒) 錯誤問題清單，請將下列政策陳述式套用至金鑰資源，以授與 IAM Access Analyzer 存取金鑰的許可。

```
{
    "Sid": "Allow IAM Access Analyzer access to key metadata",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/aws-service-role/access-
analyzer.amazonaws.com/AWSServiceRoleForAccessAnalyzer"
        },
    "Action": [
        "kms:DescribeKey",
        "kms:GetKeyPolicy",
        "kms:List*"
    ],
    "Resource": "*"
},
```

在收到 KMS 金鑰資源的 Access Denied (存取遭拒) 問題清單，然後藉由更新金鑰政策來解決問題清單後，即會將問題清單更新為已解決的狀態。如果有會將許可授與外部實體之金鑰的政策陳述式或金鑰授與，您可能會看到金鑰資源的其他問題清單。

AWS Lambda 函數和圖層

對於 AWS Lambda 功能，IAM Access Analyzer 會分析政策 (包括政策中的條件陳述式)，以便將函數存取權授予外部實體的存取權。使用 Lambda，您可以將獨特的以資源為基礎的政策附加到函數、版本、別名和層。IAM 存取分析器會根據附加到函數和層級的資源型政策來報告外部存取。IAM 存取分析器不會根據附加到別名的資源型政策以及使用合格 ARN 叫用的特定版本來報告外部存取。

如需詳細資訊，請參閱 AWS Lambda 開發人員指南中的對 Lambda 使用以資源為基礎的政策和使用版本。

Amazon Simple Queue Service 佇列

對於 Amazon SQS 佇列，IAM Access Analyzer 會分析允許外部實體存取佇列的政策 (包括政策中的條件陳述式)。

AWS Secrets Manager 秘密

對於機 AWS Secrets Manager 密，IAM Access Analyzer 會分析允許外部實體存取機密的政策 (包括政策中的條件陳述式)。

Amazon Simple Notification Service 主題

IAM Access Analyze 會分析附加至 Amazon SNS 主題的資源型政策，包括允許外部存取主題的政策中的條件陳述式。您可以允許外部帳戶執行 Amazon SNS 動作，例如透過資源型政策訂閱和發佈主題。如果來自信任區域之外的帳戶主體可以對該主題執行操作，則可從外部存取 Amazon SNS 主題。如果您建立 Amazon SNS 主題時在政策中選擇 `Everyone`，則表示該主題可供公眾存取。`AddPermission` 是向允許外部存取的 Amazon SNS 主題新增資源型政策的另一種方法。

Amazon Elastic Block Store 磁碟區快照

Amazon Elastic Block Store 磁碟區快照沒有資源型政策。快照是透過 Amazon EBS 共用許可的方式進行共用。對於 Amazon EBS 磁碟區快照，IAM Access Analyzer 會分析允許外部實體存取快照的存取控制清單。在加密時，Amazon EBS 磁碟區快照可與外部帳戶共用。未加密的磁碟區快照可與外部帳戶共用，並可授予公用存取權。共用設定位於快照的 `CreateVolumePermissions` 屬性中。當客戶預覽 Amazon EBS 快照的外部存取時，他們可以將加密金鑰指定為快照已加密的指標，類似於 IAM Access Analyzer 預覽處理 Secrets Manager 秘密的方式。

Amazon Relational Service 資料庫快照

Amazon RDS 資料庫快照沒有資源型政策。資料庫快照可透過 Amazon RDS 資料庫許可共用，且只能共用手動資料庫快照。對於 Amazon RDS 資料庫快照，IAM Access Analyzer 會分析允許外部實體存取快照的存取控制清單。未加密的資料庫快照可以是公開的。加密的資料庫快照無法公開共用，但可與多達 20 個其他帳戶共用。如需詳細資訊，請參閱建立資料庫快照。IAM Access Analyzer 會將匯出資料庫手動快照 (例如，匯出至 Amazon S3 儲存貯體) 的能力視為受信任的存取。

> **Note**
>
> IAM Access Analyzer 不會識別直接在資料庫本身設定的公用或跨帳戶存取權。IAM Access Analyzer 僅識別 Amazon RDS 資料庫快照上設定的公用或跨帳戶存取權的問題清單。

Amazon Relational Database Service 資料庫叢集快照

Amazon RDS 資料庫叢集快照沒有資源型政策。可透過 Amazon RDS 資料庫叢集許可共用快照。對於 Amazon RDS 資料庫叢集快照，IAM Access Analyzer 會分析允許外部實體存取快照的存取控制清單。未加密的叢集快照可以公開。而加密的叢集快照則無法公開共用。未加密和加密的叢集快照可與最多 20 個其他帳戶共享。如需詳細資訊，請參閱建立資料庫叢集快照。IAM Access Analyzer 會將匯出資料庫叢集快照 (例如，匯出至 Amazon S3 儲存貯體) 的能力視為受信任的存取。

> **Note**
>
> IAM 存取分析器發現項目不包括監控 Amazon RDS 資料庫叢集的任何共用，以及與其他 AWS 帳戶 或組織使用 AWS Resource Access Manager的複製。IAM Access Analyzer 僅識別 Amazon RDS 資料庫叢集快照上設定的公用或跨帳戶存取權的問題清單。

Amazon Elastic Container Registry 儲存庫

對於 Amazon ECR 儲存庫，IAM Access Analyzer 會分析資源型政策，包括政策中的條件陳述式，且這些政策允許外部實體存取儲存庫 (類似於 Amazon SNS 主題和 Amazon EFS 檔案系統等其他資源類型)。對於 Amazon ECR 儲存庫，主體必須擁有透過身分型政策 `ecr:GetAuthorizationToken` 的許可，才能將其視為外部可用。

Amazon Elastic File System 檔案系統

對於 Amazon EFS 檔案系統，IAM Access Analyzer 會分析允許外部實體存取檔案系統的政策 (包括政策中的條件陳述式)。如果來自信任區域之外的帳戶主體可以在該檔案系統上執行操作，則可從外部存取 Amazon EFS 檔案系統。可透過使用 IAM 的檔案系統政策以及檔案系統掛載方式來定義存取權。例如，在其他帳戶掛載 Amazon EFS 檔案系統會被視為可從外部存取，除非該帳戶位於您的組織中，且您已將該組織定義為您的信任區域。如果您從具有公有子網路的虛擬私有雲端掛載檔案系統，則可從外部存取檔案系統。搭配使用 Amazon EFS 時 AWS Transfer Family，如果檔案系統允許公開存取，則從 Transfer Family 統伺服器所擁有的檔案系統存取請求會遭到封鎖。

Amazon DynamoDB 串流

如果 DynamoDB 政策允許至少一個允許外部實體存取 DynamoDB 串流的跨帳戶動作，IAM 存取分析器會產生一個發現。如需 DynamoDB 支援跨帳戶動作的詳細資訊，請參閱 Amazon DynamoDB 開發人員指南中以資源為基礎的政策支援的 IAM 動作。

Amazon DynamoDB 資料表

如果 DynamoDB 政策允許至少一個允許外部實體存取 DynamoDB 表格或索引的跨帳戶動作，則 IAM 存取分析器會產生 DynamoDB 表格的發現項目。如需 DynamoDB 支援跨帳戶動作的詳細資訊，請參閱 Amazon DynamoDB 開發人員指南中以資源為基礎的政策支援的 IAM 動作。

IAM Access Analyzer 的設定

如果您 AWS Identity and Access Management Access Analyzer 在 AWS Organizations 管理帳戶中進行設定，則可以將組織中的成員帳戶新增為委派管理員，以管理組織的 IAM Access Analyzer。委派管理員擁有許可，有權建立及管理以組織做為信任區域的分析器。只有管理帳戶可以新增委派管理員。

IAM Access Analyzer 的委派管理員

IAM Access Analyzer 的委派管理員是組織內擁有許可的成員帳戶，有權建立及管理為整個組織分析存取權的分析器。只有管理帳戶可以新增、移除或變更委派管理員。

如果您新增委派管理員，您可以在稍後將委派管理員變更為不同帳戶。當您這麼做時，先前的委派管理員帳戶會失去許可，無法再管理使用該帳戶建立來為整個組織分析存取權的所有分析器。這類分析器會進入已停用狀態，且不會再產生新的問題清單或更新現有的問題清單。您也無法無法再存取這類分析器的現有問題清單。您未來可透過將該帳戶配置為委派管理員，而再次存取那些問題清單。如果您知道您不會使用該相同的帳戶作為委派管理員，請考慮在變更委派管理員之前刪除分析器。這會刪除所有產生的問題清單。當新的委派管理員建立新的分析器時，則會產生相同問題清單的新執行個體。你不會遺失任何問題清單，它們只是在不同的帳戶中為新的分析器而產生。而且您可以使用組織管理帳戶 (該帳戶也具有管理員許可) 繼續存取組織的問題清單。新委派的管理員必須為 IAM Access Analyzer 建立新的分析器，才能開始在組織中監控資源。

如果委派的管理員離開 AWS 組織，則會從帳戶中移除委派的管理權限。帳戶中所有以組織做為信任區域的分析器都會進入已停用狀態，您也無法無法再存取這類分析器的現有問題清單。

第一次在管理帳戶中設定分析器時，您可以在 IAM Access Analyzer 主控台的分析器設定頁面中，選擇新增委派管理員。

> **Note**
>
> IAM Access Analyzer 會根據每月每個分析器所分析的 IAM 角色和使用者數量，收取未使用的存取權分析器費用。如果您在管理帳戶和委派管理員帳戶中都建立了未使用的存取權分析器，將需支付兩個未使用的存取權分析器的費用。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

使用主控台新增委派管理員

1. 使用組織的管理帳戶登入 AWS 主控台。
2. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
3. 在 Access Analyzer 下，選擇分析器設定。
4. 選擇 Add delegated administrator (新增委派管理員)。
5. 在委派管理員欄位中，輸入組織成員帳戶的 AWS 帳戶 號碼以設為委派管理員。

   此帳戶必須是您組織的成員。
6. 選擇儲存變更。

若要使用 AWS CLI 或 AWS SDK 新增委派的系統管理員

當您使用 AWS CLI、 AWS API (使用 AWS SDK) 在委派的系統管理員帳戶中建立分析器以分析整個組織的存取權限時 AWS CloudFormation，或者您必須使用 AWS Organizations API 來啟用 IAM Access Analyzer 的服務存取權，並將成員帳戶註冊為委派的管理員。

1. 在中啟用 IAM 存取分析器的受信任服務存取 AWS Organizations。請參閱《使用 AWS Organizations 者指南》中的如何啟用或停用受信任的存取。
2. 使用 AWS Organizations `RegisterDelegatedAdministrator`API 作業或`register-delegated-administrator` AWS CLI 命令，將 AWS 組織的有效成員帳戶註冊為委派的系統管理員。

變更委派管理員後，新的管理員必須建立分析器，以開始在組織中監控資源的存取權。

刪除分析器

您可以從分析器設定頁面刪除現有外部和未使用的存取權分析器。刪除分析器後，系統將不再監控分析器中指定的資源，也不會產生新的調查結果。分析器產生的所有調查結果都會刪除。

對於因為產生它們的分析器被刪除而刪除的發現項目，事件會 EventBridge 在分析器被刪除後的兩天內傳送到。刪除分析器後最多可能需要 90 天才能刪除 Security Hub 調查結果。

刪除分析器

1. 前往 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 Access Analyzer 下，選擇分析器設定。

3. 選取要刪除的分析器，然後選擇刪除。
4. 在確認方塊中輸入 **delete**，然後選擇刪除。

封存規則

存檔規則會自動存檔新的問題清單，這些問題清單都符合您建立規則時所定義的條件。您也可以追溯套用封存規則，以封存符合封存規則條件的現有問題清單。例如，您可以建立封存規則，以針對您定期授予存取權的特定 Amazon S3 儲存貯體，自動封存任何調查結果。或者，如果您將多個資源的存取權授與特定主體，則可以建立規則，自動存檔任何新問題清單，這些問題清單都是為授與該主體的存取權而產生。這可讓您只專注於可能表示安全性風險的作用中問題清單。

建立存檔規則時，系統只會自動存檔符合規則條件的新問題清單。系統不會自動存檔現有的問題清單。建立規則時，您最多可以為規則中的每個條件加入 20 個值。如需可用來建立或更新存檔規則的篩選鍵清單，請參閱 IAM Access Analyzer 篩選鍵。

> **Note**
>
> 建立或編輯封存規則時，IAM Access Analyzer 不會驗證您在規則篩選條件中包含的值。例如，如果您新增規則以比對 AWS 帳戶，即使該值不是有效的 AWS 帳戶號碼，IAM Access Analyzer 仍會接受欄位中的任何值。

建立存檔規則

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 選擇 Access Analyzer，然後選擇分析器設定。
3. 在分析器區段中，選擇您要建立封存規則的分析器。
4. 在封存規則索引標籤上，選擇建立封存規則。
5. 如果要變更預設名稱，請輸入規則的名稱。
6. Rule (規則) 區段中 Criteria (條件) 下，選取要與規則比對的屬性。
7. 選擇屬性值的條件，例如包含、是或不等於。

   可用的運算子取決於您選擇的屬性。
8. 為屬性新增其他值，或為規則新增其他條件 (非必要)。針對外部存取權調查結果，若要確保規則不會封存公開存取權的新調查結果，您也可以包含公開存取權條件並設為 false。

若要為條件新增另一個值，請選擇 Add another value (新增另一個值)。若要為規則新增其他條件，請選擇新增。

9. 當您完成新增條件和值時，請選擇 Create rule (建立規則)，以將規則僅套用至新的問題清單。選擇 Create and archive active findings (建立並封存作用中的問題清單)，以根據規則條件來封存新的和現有的問題清單。在 Results (結果) 區段中，您可以檢閱封存規則所套用之作用中問題清單的清單。

例如，若要建立為 Amazon S3 儲存貯體自動封存所有調查結果的規則：選擇資源類型，然後選擇是 條件。接下來，從值清單中選擇 S3 儲存貯體。

若要針對未使用的存取權調查結果建立規則，以自動封存特定帳戶的所有調查結果，請選擇資源擁有者帳戶，然後選擇等於條件。在 [值] 文字方塊中輸入 AWS 帳戶 ID。

繼續定義條件以自訂適合您環境的規則，然後選擇建立規則。

如果您要建立新規則並新增多個條件，您可以透過選擇 Remove this criterion (移除此條件) 來從規則中移除單一條件。您可以透過選擇 Remove value (移除值) 來移除為條件新增的值。

編輯存檔規則

1. 在名稱欄中選擇要編輯的規則之名稱。

   您一次只能編輯一個存檔規則。

2. 為每個條件新增準則或移除現有準則和值。
3. 選擇 Save changes (儲存變更)，將規則僅套用至新的問題清單。選擇 Save and archive active findings (儲存並封存作用中的問題清單)，以根據規則條件來封存新的和現有的問題清單。

刪除存檔規則

1. 選擇您要刪除的規則之核取方塊。
2. 選擇刪除。
3. 在 Delete archive rule (刪除存檔規則) 確認對話方塊中輸入 **`delete`**，然後選擇 Delete (刪除)。

系統僅會在目前區域中將這些規則從分析器中刪除。您必須為您在其他區域中建立的每個分析器分別刪除存檔規則。

AWS Identity and Access Management Access Analyzer 使用 Amazon 監控 EventBridge

使用本主題中的資訊，了解如何使用 Amazon 監控 IAM 存取分析器發現並存取預覽 EventBridge。EventBridge 是 Amazon CloudWatch 活動的新版本。

問題清單事件

IAM Access Analyzer 會 EventBridge 針對每個產生的發現項目傳送事件，以變更現有發現項目的狀態，以及何時刪除發現項目。若要接收有關發現項目的發現項目和通知，您必須在 Amazon 中建立事件規則 EventBridge。建立事件規則時，您也可以根據規則指定要觸發的目標動作。例如，您可以建立事件規則，以便在從 IAM Access Analyzer 收到新問題清單的事件時觸發 Amazon SNS 主題。

存取預覽事件

IAM 存取分析器會 EventBridge 針對每個存取預覽傳送事件，並變更其狀態。這包括第一次建立存取預覽 (狀態為「建立」)、存取預覽完成 (狀態為「已完成」) 或存取預覽建立失敗 (狀態為「失敗」) 時的事件。若要接收有關存取預覽的通知，您必須在中建立事件規則 EventBridge。建立事件規則時，您可以根據規則指定要觸發的目標動作。例如，您可以建立事件規則，以便在從 IAM Access Analyzer 收到的已完成存取預覽時觸發 Amazon SNS 主題。

事件通知頻率

IAM Access Analyzer 會將新發現項目和發現項目的事件傳送至您帳戶中發生事件後大約一小時 EventBridge內的狀態更新。IAM Access Analyzer 也會在已解決的發現項目遭到刪除 EventBridge 時傳送事件，因為保留期已過期。對於因為產生它們的分析器被刪除而刪除的發現項目，事件會傳送到刪除分析器後 EventBridge 大約 24 小時。問題清單遭刪除時，問題清單的狀態不會變更。相反地，`isDeleted` 屬性會設定為 `true`。IAM Access Analyzer 也會將新建立的存取預覽和存取預覽狀態變更的事件傳送至 EventBridge。

外部存取權調查結果事件範例

以下是 IAM 存取分析器外部存取尋找事件傳送至的範例 EventBridge。`id`列出的是中事件的 ID EventBridge。若要深入瞭解，請參閱中的事件和事件模式 EventBridge。

在 `detail` 物件中，`accountId` 和 `region` 屬性的值參考結果所報告的帳戶和區域。`isDeleted` 屬性會指出事件是否來自要刪除的問題清單。`id` 是結果 ID。`resources` 陣列是具有產生結果的分析器 ARN 的單例。

```
{
    "account": "111122223333",
    "detail": {
        "accountId": "111122223333",
        "action": [
            "s3:GetObject"
        ],
        "analyzedAt": "2019-11-21T01:22:22Z",
        "condition": {},
        "createdAt": "2019-11-20T04:58:50Z",
        "id": "22222222-dcba-4444-dcba-333333333333",
        "isDeleted": false,
        "isPublic": false,
        "principal": {
            "AWS": "999988887777"
        },
        "region": "us-west-2",
        "resource": "arn:aws:s3:::my-bucket",
        "resourceType": "AWS::S3::Bucket",
        "status": "ACTIVE",
        "updatedAt": "2019-11-21T01:14:07Z",
        "version": "1.0"
    },
    "detail-type": "Access Analyzer Finding",
    "id": "11111111-2222-4444-aaaa-333333333333",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2019-11-21T01:22:33Z",
    "version": "0"
}
```

IAM 存取分析器也會傳送事件，以取 EventBridge 得錯誤發現項目。錯誤問題清單是 IAM Access Analyzer 無法分析資源時產生的問題清單。錯誤問題清單的事件包括下列範例所示的 `error` 屬性。

```
{
    "account": "111122223333",
    "detail": {
        "accountId": "111122223333",
        "analyzedAt": "2019-11-21T01:22:22Z",
        "createdAt": "2019-11-20T04:58:50Z",
```

```
        "error": "ACCESS_DENIED",
        "id": "22222222-dcba-4444-dcba-333333333333",
        "isDeleted": false,
        "region": "us-west-2",
        "resource": "arn:aws:s3:::my-bucket",
        "resourceType": "AWS::S3::Bucket",
        "status": "ACTIVE",
        "updatedAt": "2019-11-21T01:14:07Z",
        "version": "1.0"
    },
    "detail-type": "Access Analyzer Finding",
    "id": "11111111-2222-4444-aaaa-333333333333",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2019-11-21T01:22:33Z",
    "version": "0"
}
```

未使用的存取權調查結果相關事件範例

以下是 IAM 存取分析器未使用的存取尋找事件傳送至的範例 EventBridge。`id`列出的是中事件的 ID EventBridge。若要深入瞭解，請參閱中的事件和事件模式 EventBridge。

在 `detail` 物件中，`accountId` 和 `region` 屬性的值參考結果所報告的帳戶和區域。`isDeleted` 屬性會指出事件是否來自要刪除的問題清單。`id` 是結果 ID。

```
{
    "version": "0",
    "id": "dc7ce3ee-114b-3243-e249-7f10f9054b21",
    "detail-type": "Unused Access Finding for IAM entities",
    "source": "aws.access-analyzer",
    "account": "123456789012",
    "time": "2023-09-29T17:31:40Z",
    "region": "us-west-2",
    "resources": [
       "arn:aws:access-analyzer:us-west-2:123456789012:analyzer/
integTestLongLivingAnalyzer-DO-NOT-DELETE"
       ],
    "detail": {
        "findingId": "b8ae0460-5d29-4922-b92a-ba956c986277",
```

```
        "resource": "arn:aws:iam::111122223333:role/FindingIntegTestFakeRole",
        "resourceType": "AWS::IAM::Role",
        "accountId": "111122223333",
        "createdAt": "2023-09-29T17:29:18.758Z",
        "updatedAt": "2023-09-29T17:29:18.758Z",
        "analyzedAt": "2023-09-29T17:29:18.758Z",
        "previousStatus": "",
        "status": "ACTIVE",
        "version": "62160bda-8e94-46d6-ac97-9670930d8ffb",
        "isDeleted": false,
        "findingType": "UnusedPermission",
        "numberOfUnusedServices": 0,
        "numberOfUnusedActions": 1
        }
    }
```

IAM 存取分析器也會傳送事件，以取 EventBridge 得錯誤發現項目。錯誤問題清單是 IAM Access Analyzer 無法分析資源時產生的問題清單。錯誤問題清單的事件包括下列範例所示的 `error` 屬性。

```
{
    "version": "0",
    "id": "c2e7aa1a-4df7-7652-f33e-64113b8997d4",
    "detail-type": "Unused Access Finding for IAM entities",
    "source": "aws.access-analyzer",
    "account": "111122223333",
    "time": "2023-10-31T20:26:12Z",
    "region": "us-west-2",
    "resources": [
      "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/ba811f91-
de99-41a4-97c0-7481898b53f2"
      ],
    "detail": {
        "findingId": "b01a34f2-e118-46c9-aef8-0d8526b495c7",
        "resource": "arn:aws:iam::123456789012:role/TestRole",
        "resourceType": "AWS::IAM::Role",
        "accountId": "444455556666",
        "createdAt": "2023-10-31T20:26:08.647Z",
        "updatedAt": "2023-10-31T20:26:09.245Z",
        "analyzedAt": "2023-10-31T20:26:08.525Z",
        "previousStatus": "",
        "status": "ACTIVE",
        "version": "7c7a72a2-7963-4c59-ac71-f0be597010f7",
        "isDeleted": false,
```

```
        "findingType": "UnusedIAMRole",
        "error": "INTERNAL_ERROR"
        }
  }
```

存取預覽事件範例

下列範例顯示建立存取預覽 EventBridge 時傳送至的第一個事件的資料。resources 陣列是存取預覽相關聯的分析器 ARN 的單例。在 detail 物件中，id 指的是存取預覽 ID，而 configuredResources 是指為其建立存取預覽的資源。status 為 Creating，並且是指存取預覽狀態。previousStatus 未指定，因為剛剛建立存取預覽。

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::example-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.00Z",
        "region": "us-west-2",
        "status": "CREATING",
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "aaaabbbb-2222-3333-4444-555566667777",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

下列範例顯示傳送至 EventBridge 以進行存取預覽之事件的資料，其狀態從變更Creating為Completed。在詳細資料物件中，id 是指存取預覽 ID。status 和 previousStatus 是指存取預覽狀態，其中先前的狀態為 Creating，而目前的狀態為 Completed。

```
{
    "account": "111122223333",
```

```
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::example-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.000Z",
        "previousStatus": "CREATING",
        "region": "us-west-2",
        "status": "COMPLETED",
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "11112222-3333-4444-5555-666677778888",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

下列範例顯示傳送至 EventBridge 以進行存取預覽之事件的資料，其狀態從變更`Creating`為`Failed`。在 `detail` 物件中，`id` 是指存取預覽 ID。`status` 和 `previousStatus` 是指存取預覽狀態，其中先前的狀態為 `Creating`，而目前的狀態為 `Failed`。`statusReason` 欄位會提供原因代碼，指出存取預覽因為無效的資源組態而失敗。

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::example-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.00Z",
        "previousStatus": "CREATING",
        "region": "us-west-2",
        "status": "FAILED",
        "statusReason": {
            "code": "INVALID_CONFIGURATION"
        },
        "version": "1.0"
    },
```

```
    "detail-type": "Access Preview State Change",
    "id": "99998888-7777-6666-5555-444433332222",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

使用主控台建立事件規則

下列程序說明如何使用主控台建立事件規則。

1. 在以下位置打開 Amazon EventBridge 控制台 https://console.aws.amazon.com/events/。
2. 使用下列值建立監視尋找事件或存取預覽事件的 EventBridge 規則：
   - 針對規則類型，選擇具有事件模式的規則。
   - 在 Event source (事件來源) 中，選擇 Other (其他)。
   - 在 Event pattern (事件模式) 中，選擇 Custom patterns (JSON editor) (自訂模式 (JSON 編輯器))，並將下列事件模式範例貼到文字區域：
     - 若要根據外部存取權或未使用的存取權調查結果事件建立規則，請使用下列模式範例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Analyzer Finding"
  ]
}
```

     - 若要僅根據未使用的存取發現項目事件建立規則，請使用下列模式範例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Unused Access Finding for IAM entities"
```

```
  ]
}
```

> **Note**
>
> 您無法僅根據外部存取發現項目事件建立規則。

- 若要根據存取預覽事件建立規則，請使用下列模式範例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Preview State Change"
  ]
}
```

- 對於目標類型，請選擇AWS 服務，然後為選取目標選擇一個目標，例如 Amazon SNS 主題或 AWS Lambda 功能。當接收到符合規則中定義之事件模式的事件時，就會觸發目標。

若要進一步了解如何建立規則，請參閱 Amazon EventBridge 使用者指南中的建立可對事件做出反應的 Amazon EventBridge 規則。

使用 CLI 建立事件規則

1. 使用以下內容創建 Amazon EventBridge 使用的規則 AWS CLI。將規則名稱*TestRule*取代為規則的名稱。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"]}"
```

2. 您可以自訂規則，只針對產生的一部分問題清單 (例如具有特定屬性的問題清單) 觸發目標動作。下列範例示範如何建立規則，該規則會僅針對狀態為作用中的問題清單而觸發目標動作。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"],\"detail-type\":[\"Access Analyzer Finding\"],\"detail\":{\"status\":
[\"ACTIVE\"]}}"
```

下列範例示範如何建立規則，該規則會僅針對狀態從 Creating 變更為 Completed 的存取預覽而觸發目標動作。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"],\"detail-type\":[\"Access Preview State Change\"],\"detail\":{\"status
\":[\"COMPLETED\"]}}"
```

3. 若要將 Lambda 函數定義為您建立之規則的目標，請使用下列範例命令。將 ARN 中的區域和函數名稱取代為適合您環境的值。

```
aws events put-targets --rule TestRule --targets Id=1,Arn=arn:aws:lambda:us-
east-1:111122223333:function:MyFunction
```

4. 新增叫用規則目標所需的許可。下列範例示範在遵循上述範例後如何將許可授與 Lambda 函數。

```
aws lambda add-permission --function-name MyFunction --statement-id 1 --action
 'lambda:InvokeFunction' --principal events.amazonaws.com
```

整合存取分析器 AWS Security Hub

AWS Security Hub提供您中安全性狀態的全面檢視， AWS 並協助您根據安全性產業標準和最佳實務來檢查您的環境。Security Hub 會從各個 AWS 帳戶、服務和支援的協力廠商合作夥伴產品收集安全性資料，並協助您分析安全性趨勢並找出最優先順序的安全性問題。

AWS Identity and Access Management Access Analyzer 與 Security Hub 整合時，您可以將發現項目從 IAM 存取分析器傳送到 Security Hub。Security Hub 接著可將這些問題清單納入其安全狀態的分析中。

內容

- IAM Access Analyzer 的典型調查結果
- 啟用與設定整合
- 如何停止傳送問題清單

IAM Access Analyzer 如何將調查結果傳送到 Security Hub

在 Security Hub 中，將安全問題作為問題清單進行追蹤。某些發現項目來自其他 AWS 服務或協力廠商合作夥伴偵測到的問題。Security Hub 也有一組規則，用來偵測安全問題並產生問題清單。

Security Hub 提供用來跨所有這些來源管理問題清單的工具。您可以檢視並篩選問題清單列表，並檢視問題清單的詳細資訊。請參閱 AWS Security Hub 使用者指南中的檢視問題清單。您也可以追蹤問題清單的調查狀態。請參閱 AWS Security Hub 使用者指南中的對問題清單採取動作。

安全性中樞中的所有發現項目都使用稱為 AWS 安全性尋找格式 (ASFF) 的標準 JSON 格式。ASFF 包含問題來源、受影響的資源以及問題清單目前狀態的詳細資訊。請參閱 AWS Security Hub 使用者指南中的 AWS 安全問題清單格式 (ASFF)。

AWS Identity and Access Management Access Analyzer 是將發現項目傳送至 Security Hub 的其中一個 AWS 服務。對於未使用的存取，IAM Access Analyzer 會偵測授與 IAM 使用者或角色的未使用存取權限，並為每個角色產生一個發現。IAM 存取分析器然後將這些發現項目傳送至 Security Hub。對於外部存取，IAM Access Analyzer 會偵測政策聲明，允許公開存取或跨帳戶存取您組織或帳戶中受支援的資源上的外部主體。IAM 存取分析器會產生公用存取的發現項目，然後將其傳送至 Security Hub。對於跨帳戶存取，IAM Access Analyzer 一次會將一個外部主體的單一發現傳送至 Security Hub。如果 IAM Access Analyzer 中有多個跨帳戶發現項目，您必須先解決單一外部主體的「Security Hub」發現，然後 IAM Access Analyzer 提供下一個跨帳戶搜尋結果。如需在分析器信任區域之外具有跨帳戶存取權的外部主體的完整清單，您必須在 IAM Access Analyzer 中檢視發現結果。

IAM Access Analyzer 傳送的調查結果類型

IAM Access Analyzer 會使用 AWS 安全調查結果格式 (ASFF) 將調查結果傳送到 Security Hub。在 ASFF 中，`Types` 欄位提供問題清單類型。來自 IAM Access Analyzer 的調查結果可能具有以下 `Types` 值。

- 外部存取權調查結果：效果/資料暴露/授予的外部存取權
- 外部存取發現項目 — 軟體與組態檢查/AWS 安全性最佳實務/允許外部存取
- 未使用的存取發現項目 — 軟體與組態檢查/AWS 安全性最佳作法/未使用的權限
- 未使用的存取發現項目 — 軟體和組態檢查/AWS 安全最佳實務/未使用的IAM 角色
- 未使用的存取發現項目 — 軟體和組態檢查/AWS 安全最佳實務/未使用的IAM 使用者密碼

- 未使用的存取發現項目 — 軟體和組態檢查/AWS 安全最佳實務/未使用的IAM 使用者存取金鑰

傳送問題清單延遲

IAM Access Analyzer 建立新的調查結果後，通常會在 30 分鐘內傳送到 Security Hub。在符合某些條件的極少數情況下，系統不會通知 IAM Access Analyzer 政策已新增或更新。例如，變更 Amazon S3 帳戶層級封鎖公有存取設定，可能需要最多 12 個小時的時間。此外，如果 AWS CloudTrail 記錄傳遞有傳遞問題，原則變更不會觸發發現項目中報告的資源重新掃描。發生這種情況時，IAM Access Analyzer 會在下次定期掃描期間分析該新增或更新的政策。

無法使用 Security Hub 時重試

如果 Security Hub 無法使用，IAM Access Analyzer 會定期重試傳送調查結果。

更新 Security Hub 中的現有問題清單

將發現項目傳送至 Security Hub 之後， AWS Identity and Access Management Access Analyzer 會傳送更新，以反映對尋找活動的其他觀察結果至 Security Hub。更新會反映在同一個問題清單中。

由於 IAM Access Analyzer 會依資源將外部存取權調查結果分類，如果 IAM Access Analyzer 中的資源至少有一份調查結果處於作用中狀態，Security Hub 中的資源調查結果就會處於作用中狀態。如果 IAM Access Analyzer 資源中的所有調查結果都已封存或解決，則會將 Security Hub 調查結果封存。當您變更公有和跨帳戶存取之間的政策存取權時，會更新 Security Hub 問題清單。此更新可以包含對問題清單類型、標題、描述和嚴重性的變更。

IAM Access Analyzer 不會依資源將未使用的存取權調查結果分類，因此如果在 IAM Access Analyzer 中解決了未使用的存取權調查結果，Security Hub 的調查結果便會解決。當您更新產生未使用存取權調查結果的 IAM 使用者或角色時，Security Hub 調查結果便會更新。

檢視 Security Hub 中的 IAM Access Analyzer 調查結果

若要檢視 Security Hub 中 IAM Access Analyzer 調查結果，請在摘要頁面的 AWS: IAM Access Analyzer 區段中選擇查看調查結果。或者，您可以從導覽面板中選擇 Findings (問題清單)。然後，您可以選擇值為的「產品名稱：」欄位，以篩選 AWS Identity and Access Management Access Analyzer 發現項目以僅顯示發現項目**`IAM Access Analyzer`**。

解譯 Security Hub 中的 IAM Access Analyzer 調查結果名稱

AWS Identity and Access Management Access Analyzer 使用安全性搜尋結果格式 (ASFF)，將發現項目傳送至 AWS 安全性中樞。在 ASFF 中，Types (類型) 欄位提供問題清單類型。ASFF 類型使用的命名配置不同於 AWS Identity and Access Management Access Analyzer。下表包含與 AWS Identity

and Access Management Access Analyzer 發現項目在 Security Hub 中出現時相關聯之所有 ASFF 類型的詳細資訊。

ASFF 問題清單類型	Security Hub 問題清單標題	描述
效果/資料曝光/已授予的外部存取	<resource ARN> 允許公有存取	連接至資源的以資源為基礎的政策對所有外部主體允許資源的公有存取。
軟體與組態檢查/AWS 安全性最佳實務/允許外部存取	<resource ARN> 允許跨帳戶存取	連接至資源的以資源為基礎的政策對分析器信任區域外的外部主體允許跨帳戶存取。
軟體與組態檢查/AWS 安全性最佳作法/未使用權限	<資源 ARN> 包含未使用的許可	使用者或角色包含未使用的服務和動作許可。
軟體和組態檢查/AWS 安全最佳實務/未使用的IAM 角色	<資源 ARN> 包含未使用的 IAM 角色	使用者或角色包含未使用的 IAM 角色。
軟體和組態檢查/AWS 安全最佳實務/未使用的IAM 使用者密碼	<資源 ARN> 包含未使用的 IAM 使用者密碼	使用者或角色包含未使用的 IAM 使用者密碼。
軟體和組態檢查/AWS 安全最佳實務/未使用的IAM 使用者存取金鑰	<資源 ARN> 包含未使用的 IAM 使用者存取金鑰	使用者或角色包含未使用的 IAM 使用者存取金鑰。

IAM Access Analyzer 的典型調查結果

IAM Access Analyzer 會使用 AWS 安全調查結果格式 (ASFF) 將調查結果傳送到 Security Hub。

這是來自 IAM Access Analyzer 的典型外部存取權調查結果範例。

```
{
    "SchemaVersion": "2018-10-08",
    "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/my-analyzer/
arn:aws:s3:::my-bucket",
    "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
    "GeneratorId": "aws/access-analyzer",
    "AwsAccountId": "111122223333",
```

```
    "Types": ["Software and Configuration Checks/AWS Security Best Practices/External
 Access Granted"],
    "CreatedAt": "2020-11-10T16:17:47Z",
    "UpdatedAt": "2020-11-10T16:43:49Z",
    "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
    },
    "Title": "AwsS3Bucket/arn:aws:s3:::my-bucket/ allows cross-account access",
    "Description": "AWS::S3::Bucket/arn:aws:s3:::my-bucket/ allows cross-account access
 from AWS 444455556666",
    "Remediation": {
        "Recommendation": {"Text": "If the access isn't intended, it indicates a
 potential security risk. Use the console for the resource to modify or remove the
 policy that grants the unintended access. You can use the Rescan button on the Finding
 details page in the Access Analyzer console to confirm whether the change removed the
 access. If the access is removed, the status changes to Resolved."}
    },
    "SourceUrl": "https://console.aws.amazon.com/access-analyzer/home?region=us-
west-2#/findings/details/dad90d5d-63b4-6575-b0fa-ef9c556ge798",
    "Resources": [
        {
            "Type": "AwsS3Bucket",
            "Id": "arn:aws:s3:::my-bucket",
            "Details": {
                "Other": {
                    "External Principal Type": "AWS",
                    "Condition": "none",
                    "Action Granted": "s3:GetObject,s3:GetObjectVersion",
                    "External Principal": "444455556666"
                }
            }
        }
    ],
    "WorkflowState": "NEW",
    "Workflow": {"Status": "NEW"},
    "RecordState": "ACTIVE"
}
```

這是來自 IAM Access Analyzer 的典型未使用的存取權調查結果範例。

```
{
```

```
    "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-
DO-NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
      "ProductName": "IAM Access Analyzer",
      "CompanyName": "AWS",
      "Region": "us-west-2",
      "GeneratorId": "aws/access-analyzer",
      "AwsAccountId": "111122223333",
      "Types": [
        "Software and Configuration Checks/AWS Security Best Practices/Unused
 Permission"
      ],
      "CreatedAt": "2023-09-18T16:29:09.657Z",
      "UpdatedAt": "2023-09-21T20:39:16.651Z",
      "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
      },
      "Title": "AwsIamRole/arn:aws:iam::111122223333:role/IsengardRole-DO-NOT-DELETE/
 contains unused permissions",
      "Description": "AWS::IAM::Role/arn:aws:iam::111122223333:role/IsengardRole-DO-
NOT-DELETE/ contains unused service and action-level permissions",
      "Remediation": {
        "Recommendation": {
          "Text":"If the unused permissions aren’t required, delete the permissions to
 refine access to your account. Use the IAM console to modify or remove the policy that
 grants the unused permissions. If all the unused permissions are removed, the status
 of the finding changes to Resolved."
        }
      },
      "SourceUrl": "https://us-west-2.console.aws.amazon.com/access-analyzer/
home?region=us-west-2#/unused-access-findings?resource=arn%3Aaws%3Aiam%3A
%3A903798373645%3Arole%2FTestRole",
      "ProductFields": {
      "numberOfUnusedActions": "256",
      "numberOfUnusedServices": "15",
      "resourceOwnerAccount": "111122223333",
      "findingId": "DEMO24d8d-0d3f-4d3d-99f4-299fc8a62ee7",
      "findingType": "UnusedPermission",
```

```
      "aws/securityhub/FindingId": "arn:aws:securityhub:us-west-2::product/aws/access-
analyzer/arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-DO-
NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "aws/securityhub/ProductName": "AM Access Analyzer",
      "aws/securityhub/CompanyName": "AWS"
    },
    "Resources": [
    {
      "Type": "AwsIamRole",
      "Id": "arn:aws:iam::111122223333:role/TestRole"
    }
  ],
  "WorkflowState": "NEW",
  "Workflow": {
    "Status": "NEW"
    },
  "RecordState": "ARCHIVED",
  "FindingProviderFields": {
    "Severity": {
      "Label": "LOW"
    },
    "Types": [
    "Software and Configuration Checks/AWS Security Best Practices/Unused Permission"
    ]
  }
  }
]
}
```


啟用與設定整合

若要使用與 Security Hub 的整合，您必須啟用 Security Hub。如需有關如何啟用 Security Hub 的資訊，請參閱 AWS Security Hub 使用者指南中的設定 Security Hub。

當您同時啟用 IAM Access Analyzer 和 Security Hub 時，會自動啟用整合。IAM Access Analyzer 會立即開始將調查結果傳送到 Security Hub。

如何停止傳送問題清單

若要停止將問題清單傳送至 Security Hub，您可以使用 Security Hub 主控台或 API。

請參閱 AWS Security Hub 使用者指南 中的停用和啟用整合中的問題清單流程 (主控台) 或停用來自整合的問題清單流程 (Security Hub API、 AWS CLI)。

使用記錄 IAM 存取分析器 API 呼叫 AWS CloudTrail

IAM 存取分析器整合了這項服務 AWS CloudTrail，可提供 IAM 存取分析器中使用者、角色或 AWS 服務所採取的動作記錄。 CloudTrail 將 IAM 存取分析器的所有 API 呼叫擷取為事件。擷取的呼叫包括來自 IAM Access Analyzer 主控台的呼叫，以及對 IAM Access Analyzer API 操作進行的程式碼呼叫。

如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 IAM 存取分析器的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。

使用收集的資訊 CloudTrail，您可以判斷向 IAM Access Analyzer 發出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

IAM 存取分析器資訊 CloudTrail

CloudTrail 在您創建 AWS 帳戶時，您的帳戶已啟用。當活動發生在 IAM Access Analyzer 中時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續的事件記錄 (包括 IAM Access Analyzer 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

所有 IAM 存取分析器動作均由記錄， CloudTrail 並記錄在 IAM 存取分析器 API 參考中。例如，呼叫`CreateArchiveRule`和`ListFindings`動作會`CreateAnalyzer`在 CloudTrail 記錄檔中產生項目。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 要求是使用根使用者登入資料還是 AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 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。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

了解 IAM Access Analyzer 日誌檔案項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示的 CloudTrail 記錄項目會示範由名為「2021 年 6 月 14 日」的假定角色工`CreateAnalyzer`作階段所執`Alice-tempcreds`行的作業。角色工作階段是由名為 `admin-tempcreds` 的角色發行。

```
{
  "eventVersion": "1.05",
  "userIdentity":   {
    "type": "AssumedRole",
    "principalId": "AROAIBKEVSQ6C2EXAMPLE:Alice-tempcreds",
    "arn": "arn:aws:sts::111122223333:assumed-role/admin-tempcreds/Alice-tempcreds",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "true",
        "creationDate": "2021-06-14T22:54:20Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin-tempcreds",
        "accountId": "111122223333",
        "userName": "admin-tempcreds"
      },
     "webIdFederationData": {},
    }
```

```
  },
  "eventTime": "2021-06-14T22:57:36Z",
  "eventSource": "access-analyzer.amazonaws.com",
  "eventName": "CreateAnalyzer",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.179",
  "userAgent": "aws-sdk-java/1.12.79 Linux/5.4.141-78.230 OpenJDK_64-
Bit_Server_VM/25.302-b08 java/1.8.0_302 vendor/Oracle_Corporation cfg/retry-mode/
standard",
  "requestParameters": {
    "analyzerName": "test",
    "type": "ACCOUNT",
    "clientToken": "11111111-abcd-2222-abcd-222222222222",
        "tags": {
            "tagkey1": "tagvalue1"
        }
  },
  "responseElements": {
    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/test"
  },
  "requestID": "22222222-dcba-4444-dcba-333333333333",
  "eventID": "33333333-bcde-5555-bcde-444444444444",
  "readOnly": false,
  "eventType": "AwsApiCall",,
  "managementEvent": true,
  "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

IAM Access Analyzer 篩選鍵

您可以使用下列篩選鍵來定義存檔規則 (CreateArchiveRule)、更新封存規則 (UpdateArchiveRule)、擷取發現項目清單 (ListFindings和 ListFindingsV2)，或擷取資源的存取預覽發現項目清單 (ListAccessPreviewFindings)。使用 IAM API 和 AWS CloudFormation 設定封存規則沒有區別。

Criterion	Description (描述)	類型	封存規則	列出問題清單	列出存取預覽問題清單
資源	ARN 唯一識別外部主體可存取的資源。若要進一步了解，請參閱 Amazon 資源名稱 (ARN)。	字串	 是	 是	 是
resourceType `AWS::IAM::Role | AWS::KMS::Key | AWS::Lambda::Function | AWS::Lambda::LayerVersion | AWS::S3::Bucket | AWS::S3Express::DirectoryBucket | AWS::SQS::Queue | AWS::SecretsManager::Secret | AWS::EFS::FileSyst`	外部主體可存取的資源類型。	字串	 是	 是	 是

Criterion	Description (描述)	類型	封存規則	列出問題清單	列出存取預覽問題清單
em | AWS::EC2::Snapshot | AWS::ECR::Repository | AWS::RDS::DBSnapshot | AWS::RDS::DBClusterSnapshot | AWS::SNS::Topic | AWS::DynamoDB::Stream | AWS::DynamoDB::Table					
resourceOwnerAccount	擁有資源的 12 位數 AWS 帳號 ID。若要進一步了解，請參閱 AWS 帳戶識別碼。	字串	是	是	是
isPublic	指出問題清單是否回報一具有允許公有存取之政策的資源。	Boolean	是	是	是

Criterion	Description (描述)	類型	封存規則	列出問題清單	列出存取預覽問題清單
尋找類型 `UnusedIAMRole | UnusedIAMUserAccessKey | UnusedIAMUserPassword | UnusedPermission`	問題清單的類型。您只能依尋找未使用存取發現項目的類型進行篩選。	字串	 是	 是	 是
status `ACTIVE | ARCHIVED | RESOLVED`	問題清單的目前狀態。	字串	 否	 是	 是
error	指出針對問題清單所報告的錯誤。	字串	 是	 是	 是
principal.AWS	已授與問題清單 `Principal` 欄位中資源存取的帳戶。輸入 12 位數的 AWS 帳號 ID 或外部 AWS 使用者或角色的 ARN。若要進一步了解，請參閱 AWS 帳戶識別碼。	字串	 是	 是	 是

Criterion	Description (描述)	類型	封存規則	列出問題清單	列出存取預覽問題清單
principal.Federated	可存取問題清單中資源的聯合身分 ARN。若要進一步了解，請參閱身分提供者與聯合	字串	是	是	是
條件.AWS：PrincipalArn	表示為資源存取條件的主體 (IAM 使用者、角色或群組) 之 ARN。若要進一步了解，請參閱 AWS 全域條件內容鍵。	字串	是	是	是
條件.AWS：識別碼PrincipalOrg	表示為資源存取條件的主體組織識別碼。若要進一步了解，請參閱 AWS 全域條件內容金鑰。	字串	是	是	是
條件.AWS：PrincipalOrgPaths	表示為資源存取條件的組織或組織單位 (OU) ID。若要進一步了解，請參閱 AWS 全域條件內容金鑰。	字串	是	是	是
條件.AWS：SourceIp	使用指定的 IP 地址時，允許主體存取資源的 IP 地址。若要進一步了解，請參閱 AWS 全域條件內容金鑰。	IP 地址	是	是	是
條件.AWS：SourceVpc	使用指定 VPC 時，允許主體存取資源的 VPC ID。若要進一步了解，請參閱 AWS 全域條件內容金鑰。	字串	是	是	是

Criterion	Description (描述)	類型	封存規則	列出問題清單	列出存取預覽問題清單
條件. AWS：UserId	表示為存取資源條件且來自外部帳戶的 IAM 使用者之使用者 ID。若要進一步了解，請參閱 AWS 全域條件內容金鑰。	字串	 是	 是	 是
condition .cognito-identity. amazonaws .com:aud	指定為問題清單中 IAM 角色存取條件的 Amazon Cognito 身分集區 ID。若要進一步了解，請參閱 IAM 和 AWS STS 條件內容金鑰。	字串	 是	 是	 是
condition .graph.fa cebook.co m:app_id	指定為允許以 Facebook 聯合身分存取問題清單中 IAM 角色之條件的 Facebook 應用程式 ID (或網站 ID)。若要進一步了解，請參閱 IAM 和 AWS STS 條件內容金鑰。	字串	 是	 是	 是
condition .accounts .google.c om:aud	指定為存取 IAM 角色之條件的 Google 應用程式 ID。若要進一步了解，請參閱 IAM 和 AWS STS 條件內容金鑰。	字串	 是	 是	 是
條件. 公里：Caller Account	擁有服務呼叫所使用之呼叫實體 (IAM 使用者、角色或帳戶根使用者) 的帳戶 ID AWS KMS。AWS 若要深入瞭解，請參閱的條件鍵 AWS Key Management Service。	字串	 是	 是	 是

Criterion	Description (描述)	類型	封存規則	列出問題清單	列出存取預覽問題清單
condition.www.amazon.com:app_id	指定為允許使用 Login with Amazon 聯合存取角色之條件的 Amazon 應用程式 ID (或網站 ID)。如需進一步了解，請參閱	字串	 是	 是	 是
id	問題清單的 ID。	字串	 否	 是	 是
ChangeType	提供對存取預覽問題清單與 IAM Access Analyzer 中所識別的現有存取進行比較的內容。	字串	 否	 否	 是
existingFindingId	IAM Access Analyzer 中問題清單的現有 ID，僅針對存取預覽中現有的問題清單提供。	字串	 否	 否	 是
existingFindingStatus	Access Analyzer 中問題清單的現有狀態，僅針對存取預覽中現有的問題清單提供。	字串	 否	 否	 是

使用 AWS Identity and Access Management Access Analyzer的服務連結角色

AWS Identity and Access Management Access Analyzer 使用 IAM 服務連結角色。服務連結角色是直接連結至 IAM Access Analyzer 的一種特殊 IAM 角色類型。服務連結角色由 IAM Access Analyzer 預先定義，並包含該功能代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓設定 IAM Access Analyzer 更為簡單，因為您不必手動新增必要的許可。IAM Access Analyzer 會定義其服務連結角色的許可，除非另有定義，否則僅有 IAM Access Analyzer 可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能連接到任何其他 IAM 實體。

如需關於支援服務連結角色的其他服務的資訊，請參閱可搭配 IAM 運作的AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示為 Yes (是) 的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

AWS Identity and Access Management Access Analyzer的服務連結角色許可

AWS Identity and Access Management Access Analyzer 使用名為 AWSServiceRoleForAccessAnalyzer— 允許存取分析器的服務連結角色來分析資源中繼資料以供外部存取，並分析活動以識別未使用的存取。

服 AWSServiceRoleForAccessAnalyzer 務連結角色會信任下列服務擔任該角色：

- `access-analyzer.amazonaws.com`

名為 `AccessAnalyzerServiceRolePolicy` 的角色許可政策允許 IAM Access Analyzer 對特定資源完成動作。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 IAM Access Analyzer 的服務連結角色

您不需要手動建立一個服務連結角色。當您在 AWS Management Console 或 AWS API 中啟用存取分析器時，IAM 存取分析器會為您建立服務連結角色。您啟用 IAM Access Analyzer 的所有區域中都會使用相同的服務連結角色。外部存取和未使用的存取權調查結果都使用相同的服務連結角色。

> **Note**
>
> IAM Access Analyzer 是區域性的。您必須在每個區域中個別啟用 IAM Access Analyzer。

如果您刪除此服務連結角色，在下一次建立分析器時，IAM Access Analyzer 會重新建立角色。

您也可以使用 IAM 主控台透過 Access Analyzer 使用案例以建立一個服務連結角色。在 AWS CLI 或 AWS API 中，使用`access-analyzer.amazonaws.com`服務名稱建立服務連結角色。如需詳細資

訊，請參閱 IAM 使用者指南中的建立服務連結角色。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

編輯 IAM Access Analyzer 的服務連結角色

IAM 存取分析器不允許您編輯 AWSServiceRoleForAccessAnalyzer 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 IAM Access Analyzer 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 如果 IAM Access Analyzer 在您試圖刪除資源時正在使用該角色，刪除動作可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

若要刪除 IAM 存取分析器所使用的資源 AWSServiceRoleForAccessAnalyzer

1. 前往網址 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台。
2. 在 Access reports (存取報告) 區段的 Access analyzer (存取分析器) 底下，選擇 Analyzers (分析器)。
3. 在 Analyzers (分析器) 資料表中，選擇分析器清單左上方的核取方塊，以便選取所有分析器。
4. 選擇 Delete (刪除)。
5. 若要確認您要刪除分析器，請輸入 **`delete`**，然後選擇 Delete (刪除)。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForAccessAnalyzer 服務連結角色。 AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

IAM Access Analyzer 服務連結角色的支援區域

IAM Access Analyzer 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 AWS 區域與端點。

預覽存取

除了協助您識別與外部實體共用的資源之外， AWS IAM Access Analyzer 還會在部署資源許可之前向您顯示 IAM Access Analyzer 發現項目的預覽，以便您驗證政策變更是否僅授與預定的公開和跨帳戶存取權限。這可協助您從預期的外部存取資源開始。

您可以在 Amazon S3 主控台中預覽和驗證對 Amazon S3 儲存貯體的公有和跨帳戶存取權。您也可以使用 IAM 存取分析器 API，為您的資源提供建議的許可，預覽 Amazon S3 儲存貯體、 AWS KMS 金鑰、IAM 角色、Amazon SQS 佇列和 Secrets Manager 機密的公開和跨帳戶存取權限。

主題

預覽 Amazon S3 主控台中的存取

在 Amazon S3 主控台中完成儲存貯體政策後，您可以選擇預覽對 Amazon S3 儲存貯體的公有和跨帳戶存取。您可以驗證您的政策變更僅授與預定的外部存取，然後再選擇儲存變更。這個選擇性步驟可讓您預覽值區的 AWS Identity and Access Management Access Analyzer 發現項目。您可以驗證政策變更是否引進新的問題清單，或是解決外部存取的現有問題清單。您可以隨時略過此驗證步驟並儲存 Amazon S3 儲存貯體政策。

若要預覽儲存貯體的外部存取，您必須在儲存貯體區域中擁有作用中的帳戶分析器，並將帳戶做為信任區域。您也必須具有使用 IAM Access Analyzer 和預覽存取所需的許可。如需啟用 IAM Access Analyzer 和所需許可的詳細資訊，請參閱 啟用 IAM Access Analyzer。

建立或編輯儲存貯體政策時，預覽 Amazon S3 儲存貯體的存取

1. 完成建立或編輯儲存貯體政策後，請確保您的政策是有效的 Amazon S3 儲存貯體政策。政策 ARN 必須符合儲存貯體 ARN 且政策元素必須有效。
2. 在政策之下的預覽外部存取，選擇作用中的帳戶分析器，然後選擇預覽。IAM Access Analyzer 問題清單的預覽會針對您的儲存貯體產生。預覽會分析顯示的 Amazon S3 儲存貯體政策以及現有的儲存貯體許可。這包括儲存貯體和帳戶 BPA 設定、儲存貯體 ACL、連接至儲存貯體的 Amazon S3 存取點和多區域存取點，以及其政策和 BPA 設定。
3. 當存取預覽完成時，隨即顯示 IAM Access Analyzer 問題清單的預覽。儲存政策後，每個問題清單都會報告可存取您儲存貯體的帳戶以外的主體執行個體。您可以檢閱每個問題清單來驗證儲存貯體的存取。問題清單標頭會提供存取的摘要，您可以展開問題清單以檢閱問題清單詳細資訊。問題清

單徽章提供了關於保存儲存貯體政策如何變更對儲存貯體存取的內容。例如，其可協助您驗證政策變更是否引進新的問題清單，或是解決外部存取的現有問題清單：

a. 新的 – 表示政策將引進的新外部存取問題清單。
b. 已解決 – 表示政策將移除之現有外部存取的問題清單。
c. 已封存 – 表示新外部存取的問題清單會根據分析器的封存規則自動封存，該分析器定義何時應將問題清單標示為預期。
d. 現有的 – 表示外部存取的現有問題清單會維持不變。
e. 公有 – 如果問題清單是用於公有存取該資源，它除了上述其中一個徽章之外，還將具有公有徽章。

4. 如果您識別不打算引進或移除的外部存取，您可以修訂政策，然後再次選擇預覽，直到您達到您想要的外部存取為止。如果您具有標籤為公有的問題清單，建議您修改政策以移除公有存取，然後再選擇儲存變更。預覽存取是選擇性的步驟，您隨時可以選擇儲存變更。

使用 IAM Access Analyzer API 預覽存取

您可以使用 IAM 存取分析器 API 來預覽 Amazon S3 儲存貯體、 AWS KMS 金鑰、IAM 角色、Amazon SQS 佇列和機 Secrets Manager 的公開和跨帳戶存取。您可以為您擁有的現有資源或要部署的新資源提供建議的許可來預覽存取。

若要預覽資源的外部存取，您必須擁有資源帳戶和區域的作用中帳戶分析器。您也必須具有使用 IAM Access Analyzer 和預覽存取所需的許可。如需啟用 IAM Access Analyzer 和所需許可的詳細資訊，請參閱 啟用 IAM Access Analyzer。

若要預覽資源的存取，您可以使用 `CreateAccessPreview` 操作，並提供資源的分析器 ARN 和存取控制組態。服務會傳回存取預覽的唯一 ID，您可以使用透過 `GetAccessPreview` 操作該 ID 來檢查存取預覽的狀態。當狀態為 `Completed` 時，您可以使用 `ListAccessPreviewFindings` 操作來擷取針對存取預覽產生的問題清單。`GetAccessPreview` 和 `ListAccessPreviewFindings` 操作會擷取約 24 小時內建立的存取預覽和問題清單。

擷取的每個問題清單都包含描述存取的問題清單詳細資訊。說明問題清單在許可部署之後是否為 `Active`、`Archived` 或 `Resolved` 以及 `changeType` 的問題清單預覽狀態。`changeType` 提供對存取預覽問題清單與 IAM Access Analyzer 中所識別的現有存取進行比較的內容：

- 新的 – 問題清單是針對新引進的存取。
- 未變更 – 預覽問題清單是保持不變的現有問題清單。
- 已變更 – 預覽問題清單是狀態變更的現有問題清單。

status 與 changeType 可協助您了解資源組態如何變更現有資源存取。如果 changeType 是 Unchanged 或已變更，則問題清單也會包含 IAM Access Analyzer 中問題清單的現有 ID 和狀態。例如，包含預覽狀態 Resolved 和現有狀態 Active 的 Changed 問題清單表示資源的現有 Active 問題清單將成為 Resolved，做為提議許可變更的結果。

您可以使用 ListAccessPreviews 操作來擷取指定分析器的存取預覽清單。此操作會擷取約一小時內建立的存取預覽相關資訊。

一般而言，如果存取預覽是針對現有資源而您未指定組態選項，則依預設，存取預覽將使用現有的資源組態。如果存取預覽是針對新資源而您未指定組態選項，則存取預覽將根據資源類型使用預設值。如需每種資源類型的組態案例，請參閱下列項目。

預覽 Amazon S3 儲存貯體的存取

如需為新的 Amazon S3 儲存貯體或您擁有的現有 Amazon S3 儲存貯體建立存取預覽，您可以指定連接到儲存貯體的 Amazon S3 儲存貯體政策、儲存貯體 ACL、儲存貯體 BPA 設定和 Amazon S3 存取點 (包括多區域存取點) 來建議儲存貯體組態。

> **Note**
>
> 在嘗試為新儲存貯體建立存取預覽之前，建議您呼叫 Amazon S3 HeadBucket操作以檢查具名儲存貯體是否已存在。此操作對於判斷儲存貯體是否存在以及您是否有許可加以存取很有用。

儲存貯體政策 – 如果組態適用於現有 Amazon S3 儲存貯體，而您未指定 Amazon S3 儲存貯體政策，則存取預覽會使用附加到儲存貯體的現有政策。如果存取預覽適用於新資源，且您未指定 Amazon S3 儲存貯體政策，則存取預覽會假設沒有政策的儲存貯體。若要提議刪除現有儲存貯體政策，您可以指定空字串。如需受支援儲存貯體政策限制的詳細資訊，請參閱儲存貯體政策範例。

儲存貯體 ACL 授與 – 您可以提議每個儲存貯體最多 100 個 ACL 授與。如果提議的授與組態適用於現有儲存貯體，則存取預覽會使用提議的授與組態清單來取代現有的授與。否則，存取預覽會使用儲存貯體的現有授與。

儲存貯體存取點 – 分析支援每個儲存貯體最多 100 個存取點 (包括多區域存取點)，包括每個儲存貯體最多可提議的 10 個新存取點。如果提議的 Amazon S3 存取點組態適用於現有儲存貯體，則存取預覽會使用提議的存取點組態來取代現有的存取點。若要提議沒有政策的存取點，您可以提供空字串做為存取點政策。如需存取點政策限制的詳細資訊，請參閱存取點的法規與限制。

封鎖公有存取組態 – 如果提議的組態適用於現有 Amazon S3 儲存貯體，而您未指定組態，則存取預覽會使用現有的設定。如果提議的組態適用於新儲存貯體，且您未指定儲存貯體 BPA 組態，則存取預覽

會使用 `false`。如果提議的組態是針對新的存取點或多區域存取點，而您未指定存取點 BPA 組態，則存取預覽會使用 `true`。

預覽存取您的 AWS KMS 金鑰

若要為您擁有的新 AWS KMS 金鑰或現有 AWS KMS 金鑰建立存取預覽，您可以透過指定 AWS KMS 金鑰原則和 AWS KMS 授與組態來提出金鑰組態。

AWS KMS 金鑰原則 — 如果組態適用於現有金鑰，而您未指定金鑰原則，則存取預覽會使用金鑰的現有原則。如果存取預覽是針對新資源，而您未指定金鑰政策，則存取預覽會使用預設金鑰政策。提議的金鑰政策不得為空字串。

AWS KMS grants — 分析每個組態最多支援 100 KMS 授權 *.* 如果提議的授權組態適用於現有金鑰，則存取預覽會使用建議的授權組態清單來取代現有授權。否則，存取預覽會使用金鑰的現有授與。

預覽您 IAM 角色的存取

若要建立新 IAM 角色的存取預覽或您擁有的現有 IAM 角色，您可以指定金鑰政策來提議一個 IAM 角色組態。

角色信任政策 – 如果組態適用於新的 IAM 角色，您必須指定信任政策。如果組態適用於您所擁有的現有 IAM 角色，且您未提議信任政策，則存取預覽會使用該角色的現有信任政策。提議的信任政策不得為空字串。

預覽您 Amazon SQS 佇列的存取

若要為新的 Amazon SQS 佇列或您擁有的現有 Amazon SQS 佇列建立存取預覽，您可以指定佇列的 Amazon SQS 政策來提議 Amazon SQS 佇列組態。

Amazon SQS 佇列政策 – 如果組態適用於現有 Amazon SQS 佇列，而您未指定 Amazon SQS 政策，則存取預覽會將現有的 Amazon SQS 政策用於佇列。如果存取預覽是針對新資源，而您未指定政策，則存取預覽會假設 Amazon SQS 佇列不包含政策。若要提議刪除現有的 Amazon SQS 佇列政策，您可以為 Amazon SQS 政策指定空字串。

預覽您 Secrets Manager 秘密的存取

若要為新的 Secrets Manager 密碼或您擁有的現有 Secrets Manager 密碼建立存取預覽，您可以透過指定密碼原則和選用的 AWS KMS 加密金鑰來建議 Secrets Manager 密碼組態。

秘密政策 – 如果組態是針對現有秘密，而您未指定秘密政策，則存取預覽會針對秘密使用現有的政策。如果存取預覽是針對新資源，而您未指定政策，則存取預覽會假設秘密不包含政策。若要提議刪除現有政策，您可以指定空字串。

AWS KMS 加密金鑰 — 如果提議的組態適用於新密碼，而您未指定金 AWS KMS 鑰 ID，則存取預覽會使用 AWS 帳戶的預設 KMS 金鑰。如果您為金 AWS KMS 鑰 ID 指定空字串，存取預覽會使用 AWS 帳戶的預設 KMS 金鑰。

驗證政策檢查

IAM Access Analyzer 提供政策檢查功能，協助您在將 IAM 政策連接到實體之前驗證這些政策。包括政策驗證提供的基本政策檢查，根據政策文法和 AWS 最佳實務來驗證您的政策。您可以檢視政策驗證檢查問題清單，包含安全警告、錯誤、一般警告和政策的建議。

您可以使用自訂政策檢查，根據您的安全標準來檢查是否有新的存取權。每次檢查新存取權都會收取費用。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

主題

- IAM Access Analyzer 政策驗證
- IAM Access Analyzer 自訂政策檢查

IAM Access Analyzer 政策驗證

您可以使用政策驗證來驗證您的 AWS Identity and Access Management Access Analyzer 政策。您可以使用 IAM 主控台中的 AWS CLI AWS API 或 JSON 政策編輯器建立或編輯政策。IAM Access Analyzer 會根據 IAM 政策文法和 AWS 最佳實務來驗證您的政策。您可以檢視政策驗證檢查問題清單，包含安全警告、錯誤、一般警告和政策的建議。這些問題清單提供可行的建議，協助您撰寫具有功能性且符合安全最佳實務的政策。若要檢視 IAM Access Analyzer 執行的基本政策檢查清單，請參閱：Access Analyzer 政策檢查參考。

在 IAM (主控台) 中驗證政策

您在 IAM 主控台中建立或編輯受管政策時，可以檢視 IAM Access Analyzer 政策驗證產生的調查結果。您也可以檢視內嵌使用者或角色政策的這些問題清單。IAM Access Analyzer 不會為內嵌群組政策產生調查結果。

檢視由 IAM JSON 政策的政策檢查產生的問題清單

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 使用以下其中一個方法開始建立或編輯政策：

a. 若要建立新的受管政策，請移至 Policies (政策) 頁面並建立新政策。如需詳細資訊，請參閱正在使用 JSON 編輯器建立政策。

b. 若要檢視現有客戶管理政策的政策檢查，請移至政策頁面，選擇政策的名稱，然後選擇編輯。如需詳細資訊，請參閱 編輯客戶受管政策 (主控台)。

c. 若要檢視使用者或角色內嵌政策的政策檢查，請移至使用者或角色頁面，選擇使用者或角色的名稱，然後選擇許可索引標籤上的政策名稱，然後選擇編輯。如需詳細資訊，請參閱 編輯客戶受管政策 (主控台)。

3. 在政策編輯器中，選擇 JSON 標籤。

4. 在政策下方的政策驗證窗格中，選擇下列一或多個標籤。標籤名稱也會指出政策的每個問題清單類型數目。

   - 安全性 — 如果您的原則允許存取因為存取過於寬鬆而 AWS 考慮安全性風險的存取，請檢視警告。
   - 錯誤 – 如果您的政策包含使政策無法運作的行，請檢視錯誤。
   - 警告 – 如果您的政策不符合最佳做法，但問題不屬於安全風險，請檢視警告。
   - 建議 – 如果 AWS 建議不影響政策之許可的改進功能，請檢視警告。

5. 檢閱 IAM Access Analyzer 政策檢查所提供的問題清單詳細資訊。每個問題清單都會指出報告問題的位置。若要深入了解造成問題的原因以及如何解決問題，請選擇報告問題旁的 Learn more (進一步了解) 連結。您也可以在 Access Analyzer policy checks (Access Analyzer 政策檢查) 參考頁面中搜尋與每個問題清單相關聯的政策檢查。

6. 選用。如果您正在編輯現有政策，可以執行自訂政策檢查，以判斷更新版政策與現有版本相比，是否會授予新的存取權。在政策下方的政策驗證窗格中，選擇檢查新的存取權索引標籤，然後選擇檢查政策。如果修改後的許可會授予新存取權，該陳述式會在政策驗證窗格中反白顯示。如果您不打算授予新的存取權，請更新政策陳述式並選擇檢查政策，直到沒有偵測到新的存取權為止。如需詳細資訊，請參閱 IAM Access Analyzer 自訂政策檢查。

   > **Note**
   >
   > 每次檢查新存取權都會收取費用。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

7. 更新您的政策以解決問題清單。

> **Important**
>
> 在您的生產工作流程中實作新的或已編輯的政策之前，請先完整測試。

8. 完成時，選擇 Next (下一步)。政策驗證器會報告 IAM Access Analyzer 未報告的所有語法錯誤。

> **Note**
>
> 您可以隨時在視覺化和 JSON 索引標籤之間切換。不過，如果您進行變更或在視覺化索引標籤中選擇下一步，IAM 可能會調整您的政策結構，以針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

9. 若使用新政策，在檢閱與建立頁面上，為您正在建立的政策輸入政策名稱與描述 (選用)。檢閱此政策中定義的許可，來查看您的政策所授予的許可。然後選擇 Create policy (建立政策) 來儲存您的工作。

   若使用現有的政策，在檢閱與儲存頁面上，檢閱在此政策中定義的許可來查看您的政策所授予的許可。選擇將此新版本設定為預設值核取方塊，將更新版政策儲存為預設版政策。選擇儲存變更以儲存編輯內容。

使用 IAM 存取分析器 (AWS CLI 或 AWS API) 驗證政策

您可以從 AWS Command Line Interface (AWS CLI) 檢視 IAM Access Analyzer 政策驗證所產生的調查結果。

檢視 IAM 存取分析器政策驗證 (AWS CLI 或 AWS API) 產生的發現項目

請使用下列其中一個：

- AWS CLI：aws accessanalyzer validate-policy
- AWS API：ValidatePolicy

Access Analyzer 政策檢查參考

您可以使用政策驗證來驗證您的 AWS Identity and Access Management Access Analyzer 政策。您可以使用 IAM 主控台中的 AWS CLI AWS API 或 JSON 政策編輯器建立或編輯政策。IAM Access Analyzer 會根據 IAM 政策文法和 AWS 最佳實務來驗證您的政策。您可以檢視政策驗證檢查問題清單，包含安全警告、錯誤、一般警告和政策的建議。這些問題清單提供可行的建議，協助您撰寫具有功

能性且符合安全最佳實務的政策。IAM Access Analyzer 提供的基本政策檢查清單如下。執行政策驗證檢查不會額外收費。若要進一步了解如何使用政策驗證來驗證政策，請參閱：IAM Access Analyzer 政策驗證。

錯誤 – 不允許 ARN 帳戶

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
ARN account not allowed: The service {{service}} does not support specifying an account
 ID in the resource ARN.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The service {{service}} does not support specifying an account ID in
 the resource ARN."
```

解決錯誤

從資源 ARN 中移除帳戶 ID。某些 AWS 服務的資源 ARN 不支援指定帳號 ID。

例如，Amazon S3 不支援帳戶 ID 做為儲存貯體 ARN 中的命名空間。Amazon S3 儲存貯體名稱是全域唯一的，所有 AWS 帳戶都共用該命名空間。若要檢視 Amazon S3 中可用的所有資源類型，請參閱服務授權參考中的 Amazon S3 定義的資源類型。

相關用語

- 政策資源
- 帳戶識別符
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 – 不允許 ARN 區域

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
ARN Region not allowed: The service {{service}} does not support specifying a Region in
 the resource ARN.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The service {{service}} does not support specifying a Region in the
 resource ARN."
```

解決錯誤

從資源 ARN 中移除區域。某些 AWS 服務的資源 ARN 不支援指定區域。

例如，IAM 是全球服務。IAM 資源 ARN 的「區域」部分一律保持空白。IAM 資源是全球性的，就像現在的 AWS 帳戶一樣。例如，以 IAM 使用者身分登入後，您就可以存取任何地理區域的 AWS 服務。

- 政策資源
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 – 資料類型不符

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Data type mismatch: The text does not match the expected JSON data type {{data_type}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The text does not match the expected JSON data type {{data_type}}."
```

解決錯誤

更新文字以使用支援的資料類型。

例如，`Version` 全域條件鍵需要 `String` 資料類型。如果您提供日期或整數，則資料類型將不相符。

相關用語

- 全域條件鍵
- IAM JSON 政策元素：條件運算子

錯誤 – 重複鍵使用不同的大小寫

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Duplicate keys with different case: The condition key {{key}} appears more than once
 with different capitalization in the same condition block. Remove the duplicate
 condition keys.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key {{key}} appears more than once with different
 capitalization in the same condition block. Remove the duplicate condition keys."
```

解決錯誤

檢閱相同條件區塊中的類似條件鍵，並針對所有執行個體使用相同的大小寫。

條件區塊是政策聲明 `Condition` 元素內的文字。條件鍵 names 不區分大小寫。條件鍵 values 是否區分大小寫取決於您使用的條件運算子。如需條件鍵中區分大小寫的詳細資訊，請參閱 IAM JSON 政策元素：Condition。

相關用語

- 條件
- 條件區塊
- 全域條件鍵
- AWS 服務條件索引鍵

錯誤 - 無效的動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid action: The action {{action}} does not exist. Did you mean {{valid_action}}?
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The action {{action}} does not exist. Did you mean
 {{valid_action}}?"
```

解決錯誤

您指定的動作無效。如果您輸入錯誤的服務字首或動作名稱，就會發生這種情況。對於某些常見問題，政策檢查會傳回建議的動作。

相關用語

- 政策動作
- AWS 服務動作

AWS 具有此錯誤的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

下列 AWS 受管理的策略在其策略陳述式中包含無效的動作。無效的動作並不會影響政策所授與的許可。使用 AWS 受管原則做為建立受管理原則的參考時，建 AWS 議您從原則中移除無效的動作。

- 亞馬遜電子FullAccessPolicy產品 _v2
- CloudWatchSyntheticsFullAccess

錯誤 - 無效的 ARN 帳戶

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid ARN account: The resource ARN account ID {{account}} is not valid. Provide a
 12-digit account ID.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The resource ARN account ID {{account}} is not valid. Provide a 12-
digit account ID."
```

解決錯誤

更新資源 ARN 中的帳戶 ID。帳戶 ID 是 12 位數的整數。若要瞭解如何檢視您的帳戶 ID，請參閱尋找您的 AWS 帳戶 ID。

相關用語

- 政策資源
- 帳戶識別符
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 - 無效的 ARN 字首

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid ARN prefix: Add the required prefix (arn) to the resource ARN.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add the required prefix (arn) to the resource ARN."
```

解決錯誤

AWS 資源 ARN 必須包含必要的`arn:`前置詞。

相關用語

- 政策資源
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 - 無效的 ARN 區域

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid ARN Region: The Region {{region}} is not valid for this resource. Update the
 resource ARN to include a supported Region.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The Region {{region}} is not valid for this resource. Update the
 resource ARN to include a supported Region."
```

解決錯誤

資源類型在指定區域中不支援。如需每個區域支援的 AWS 服務表格，請參閱 Re gion 資料表。

相關用語

- 政策資源

- 資源 ARN
- 區域名稱和代碼

錯誤 - 無效的 ARN 資源

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid ARN resource: Resource ARN does not match the expected ARN format. Update the
 resource portion of the ARN.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Resource ARN does not match the expected ARN format. Update the
 resource portion of the ARN."
```

解決錯誤

資源 ARN 必須符合已知資源類型的規格。若要檢視服務的預期 ARN 格式，請參閱服務的動作、資源和條件金鑰。 AWS 選擇服務的名稱，以檢視其資源類型和 ARN 格式。

相關用語

- 政策資源
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 - 無效的 ARN 服務案例

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid ARN service case: Update the service name ${service} in the resource ARN to use
 all lowercase letters.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Update the service name ${service} in the resource ARN to use all
 lowercase letters."
```

解決錯誤

資源 ARN 中的服務必須符合服務字首的規格 (包括大小寫)。若要檢視服務的前置詞，請參閱服務的動作、資源和條件索 AWS 引鍵。選擇服務的名稱，然後在第一句中找到該服務的字首。

相關用語

- 政策資源
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 - 無效的條件資料類型

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid condition data type: The condition value data types do not match. Use condition
 values of the same JSON data type.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition value data types do not match. Use condition values of
 the same JSON data type."
```

解決錯誤

條件鍵值組中的值必須符合條件鍵和條件運算子的資料類型。若要檢視服務的條件索引鍵資料類型，請參閱服務的動作、資源和條件索 AWS 引鍵。選擇服務的名稱，以檢視該服務的條件鍵。

例如，`CurrentTime`全域條件鍵支援 `Date` 條件運算子。如果您為條件區塊中的值提供字串或整數，則資料類型將不相符。

相關用語

- 條件
- 條件區塊
- IAM JSON 政策元素：條件運算子
- 全域條件鍵
- AWS 服務條件索引鍵

錯誤 - 無效的條件鍵格式

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid condition key format: The condition key format is not valid. Use the format
 service:keyname.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key format is not valid. Use the format
 service:keyname."
```

解決錯誤

條件鍵值組中的鍵必須符合服務的規格。若要檢視服務的條件索引鍵，請參閱服務的動作、資源和條件索 AWS 引鍵。選擇服務的名稱，以檢視該服務的條件鍵。

相關用語

- 條件
- 全域條件鍵
- AWS 服務條件索引鍵

錯誤 - 無效條件多重布林值

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid condition multiple Boolean: The condition key does not support multiple Boolean
 values. Use a single Boolean value.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key does not support multiple Boolean values. Use a
 single Boolean value."
```

解決錯誤

條件鍵值對中的鍵需要單一布林值。當您提供多個布林值時，條件比對可能不會傳回您預期的結果。

若要檢視服務的條件索引鍵，請參閱服務的動作、資源和條件索 AWS 引鍵。選擇服務的名稱，以檢視該服務的條件鍵。

- 條件
- 全域條件鍵
- AWS 服務條件索引鍵

錯誤 - 無效的條件運算子

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid condition operator: The condition operator {{operator}} is not valid. Use a
 valid condition operator.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition operator {{operator}} is not valid. Use a valid
 condition operator."
```

解決錯誤

更新條件以使用支援的條件運算子。

相關用語

- IAM JSON 政策元素：條件運算子
- 條件元素
- JSON 政策概觀

錯誤 — 效果的無效

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid effect: The effect {{effect}} is not valid. Use Allow or Deny.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The effect {{effect}} is not valid. Use Allow or Deny."
```

解決錯誤

更新 `Effect` 元素以使用有效的效果。`Effect` 的有效值為 **`Allow`** 和 **`Deny`**。

相關用語

- 效果元素
- JSON 政策概觀

錯誤 — 無效的全域條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid global condition key: The condition key {{key}} does not exist. Use a valid
 condition key.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key {{key}} does not exist. Use a valid condition
 key."
```

解決錯誤

更新條件鍵值組中的條件鍵，以使用支援的全域條件鍵。

全域條件索引鍵是具有`aws:`前置字元的條件索引鍵。 AWS 服務可以支援全域條件金鑰，或提供包含其服務前置詞的服務特定金鑰。例如，IAM 條件鍵會包含 `iam:` 字首。如需詳細資訊，請參閱AWS 服務的動作、資源和條件金鑰，然後選擇您要檢視其金鑰的服務。

相關用語

- 全域條件鍵

錯誤 — 無效的分割區

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid partition: The resource ARN for the service {{service}} does not support the
 partition {{partition}}. Use the supported values: {{partitions}}
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The resource ARN for the service {{service}} does not support the
 partition {{partition}}. Use the supported values: {{partitions}}"
```

解決錯誤

更新資源 ARN 以包含支援的分割區。如果您包含支援的分割區，則服務或資源可能不支援您包含的分割區。

分割區是一組 AWS 區域。每個 AWS 帳戶的範圍是一個分區。在傳統區域中，使用 aws 分割區。在中國區域，使用 aws-cn。

相關用語

- Amazon 資源名稱 (ARN) - 分割區

錯誤 — 無效的政策元素

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid policy element: The policy element {{element}} is not valid.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The policy element {{element}} is not valid."
```

解決錯誤

更新政策以僅包含支援的 JSON 政策元素。

相關用語

- JSON 政策元素：

錯誤 — 無效的主體格式

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid principal format: The Principal element contents are not valid. Specify a key-
value pair in the Principal element.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The Principal element contents are not valid. Specify a key-value
 pair in the Principal element."
```

解決錯誤

更新主體，以使用支援的鍵值組格式。

您可以在以資源為基礎的政策中指定主體，但不能在以身分為基礎的政策中指定。

例如，若要定義 AWS 帳戶中每個人的存取權限，請在您的政策中使用下列主體：

```
"Principal": { "AWS": "123456789012" }
```

相關用語

- JSON 政策元素：主體
- 以身分為基礎的政策和以資源為基礎的政策

錯誤 - 無效的主體鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid principal key: The principal key {{principal-key}} is not valid.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The principal key {{principal-key}} is not valid."
```

解決錯誤

更新主體鍵值組中的鍵，以使用支援的主體鍵。以下是支援的主體鍵：

- AWS
- CanonicalUser
- 聯合
- 服務

相關用語

- 主體元素

錯誤 - 無效的區域

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid Region: The Region {{region}} is not valid. Update the condition value to a
 suported Region.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The Region {{region}} is not valid. Update the condition value to a
 suported Region."
```

解決錯誤

更新條件鍵值組的值，以包含支援的區域。如需每個區域支援的 AWS 服務表格，請參閱 Re gion 資料表。

相關用語

- 政策資源
- 資源 ARN
- 區域名稱和代碼

錯誤 - 無效的服務

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid service: The service {{service}} does not exist. Use a valid service name.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The service {{service}} does not exist. Use a valid service name."
```

解決錯誤

動作或條件鍵中的服務字首必須符合服務字首的規格 (包括大小寫)。若要檢視服務的前置詞，請參閱服務的動作、資源和條件索 AWS 引鍵。選擇服務的名稱，然後在第一句中找到該服務的字首。

相關用語

- 已知的服務及其動作、資源和條件鍵

錯誤 - 無效的服務條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid service condition key: The condition key {{key}} does not exist in the service
 {{service}}. Use a valid condition key.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key {{key}} does not exist in the service {{service}}.
 Use a valid condition key."
```

解決錯誤

更新條件鍵值組中的鍵，以使用服務的已知條件鍵。全域條件鍵名稱是以 aws 字首為開頭。 AWS 服務可提供包含其服務字首的服務專屬鍵。若要檢視服務的前置詞，請參閱服務的動作、資源和條件索 AWS 引鍵。

相關用語

- 全域條件鍵
- 已知的服務及其動作、資源和條件鍵

錯誤 - 動作中的服務無效

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid service in action: The service {{service}} specified in the action does not
 exist. Did you mean {{service2}}?
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The service {{service}} specified in the action does not exist. Did
 you mean {{service2}}?"
```

解決錯誤

動作中的服務字首必須符合服務字首的規格 (包括大小寫)。若要檢視服務的前置詞，請參閱服務的動作、資源和條件索 AWS 引鍵。選擇服務的名稱，然後在第一句中找到該服務的字首。

相關用語

- 動作元素
- 已知的服務及其行動

錯誤 — 運算子的無效變數

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid variable for operator: Policy variables can only be used with String and ARN
 operators.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Policy variables can only be used with String and ARN operators."
```

解決錯誤

您可以在 Resource 元素中使用政策變數，也可以在 Condition 元素中使用字串比較。當您使用字串運算子或 ARN 運算子時，條件支援變數。字串運算子包括 StringEquals、StringLike 和 StringNotLike。ARN 運算子包括 ArnEquals 和 ArnLike。您無法使用政策變數搭配其他運算子，例如 Numeric、Date、Boolean、Binary、IP Address 或 Null 運算子。

相關用語

- 在「條件」元素中使用政策變數
- 條件元素

錯誤 - 無效的版本

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid version: The version ${version} is not valid. Use one of the following
 versions: ${versions}
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The version ${version} is not valid. Use one of the following
 versions: ${versions}"
```

解決錯誤

`Version`原則元素會指定 AWS 用來處理策略的語言語法規則。若要使用所有可用的政策功能，請在所有政策的 `Statement` 元素前面包含最新的 `Version` 元素。

```
"Version": "2012-10-17"
```

相關用語

- 版本元素

錯誤 – Json 語法錯誤

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Json syntax error: Fix the JSON syntax error at index {{index}} line {{line}} column
 {{column}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Fix the JSON syntax error at index {{index}} line {{line}} column
 {{column}}."
```

解決錯誤

您的政策包含語法錯誤。檢查您的 JSON 語法。

相關用語

- JSON 驗證器

- IAM JSON 政策元素參考
- JSON 政策概觀

錯誤 – Json 語法錯誤

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Json syntax error: Fix the JSON syntax error.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Fix the JSON syntax error."
```

解決錯誤

您的政策包含語法錯誤。檢查您的 JSON 語法。

相關用語

- JSON 驗證器
- IAM JSON 政策元素參考
- JSON 政策概觀

錯誤 - 缺少動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing action: Add an Action or NotAction element to the policy statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add an Action or NotAction element to the policy statement."
```

解決錯誤

AWS JSON 政策必須包含`Action`或`NotAction`元素。

相關用語

- 動作元素
- NotAction 元素
- JSON 政策概觀

錯誤 – 缺少 ARN 欄位

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing ARN field: Resource ARNs must include at least {{fields}} fields in the
 following structure: arn:partition:service:region:account:resource
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Resource ARNs must include at least {{fields}} fields in the
 following structure: arn:partition:service:region:account:resource"
```

解決錯誤

資源 ARN 中的所有欄位必須符合已知資源類型的規格。若要檢視服務的預期 ARN 格式，請參閱服務的動作、資源和條件金鑰。 AWS 選擇服務的名稱，以檢視其資源類型和 ARN 格式。

相關用語

- 政策資源
- 資源 ARN
- AWS 具有 ARN 格式的服務資源

錯誤 – 缺少 ARN 區域

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing ARN Region: Add a Region to the {{service}} resource ARN.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a Region to the {{service}} resource ARN."
```

解決錯誤

大部分 AWS 服務的資源 ARN 都需要您指定 [區域]。如需每個區域支援的 AWS 服務表格，請參閱 Region 資料表。

相關用語

- 政策資源
- 資源 ARN
- 區域名稱和代碼

錯誤 – 缺少效果

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing effect: Add an Effect element to the policy statement with a value of Allow or
 Deny.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add an Effect element to the policy statement with a value of Allow
 or Deny."
```

解決錯誤

AWS JSON 政策必須包含值為**`Allow`**和的`Effect`元素**`Deny`**。

相關用語

- 效果元素
- JSON 政策概觀

錯誤 – 缺少主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing principal: Add a Principal element to the policy statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a Principal element to the policy statement."
```

解決錯誤

以資源為基礎的政策必須包含 `Principal` 元素。

例如，若要定義 AWS 帳戶中每個人的存取權限，請在您的政策中使用下列主體：

```
"Principal": { "AWS": "123456789012" }
```

相關用語

- 主體元素
- 以身分為基礎的政策和以資源為基礎的政策

錯誤 – 缺少限定詞

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing qualifier: The request context key ${key} has multiple values. Use the
 ForAllValues or ForAnyValue condition key qualifiers in your policy.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The request context key ${key} has multiple values. Use the
 ForAllValues or ForAnyValue condition key qualifiers in your policy."
```

解決錯誤

在 `Condition` 元素中，您所建置的表達式使用條件運算子 (例如等於或小於) 來比較政策中的條件鍵與值，以及請求內容中的鍵與值。對於包含單一條件鍵之多個值的請求，您必須像陣列一樣，在括號內括住條件 ("Key2":["Value2A", "Value2B"])。您還必須使用 `ForAllValues` 或 `ForAnyValue` 設定運算子搭配 `StringLike` 條件運算子。這些限定詞新增設定操作功能到條件運算子，以便您可以針對多個條件值測試多個請求值。

相關用語

- 多值內容鍵
- 條件元素

AWS 具有此錯誤的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

下列 AWS 受管理的政策在其原則陳述式中包含條件索引鍵遺失的限定詞。使用 AWS 受管政策作為建立客戶管理政策的參考時，建 AWS 議您將`ForAllValues`或`ForAnyValue`條件索引鍵限定詞新增至`Condition`元素。

- AWSGlueConsoleSageMakerNotebookFullAccess

錯誤 – 缺少資源

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing resource: Add a Resource or NotResource element to the policy statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a Resource or NotResource element to the policy statement."
```

解決錯誤

除角色信任原則以外的所有原則都必須包含`Resource`或`NotResource`元素。

相關用語

- 資源元素
- NotResource 元素
- 以身分為基礎的政策和以資源為基礎的政策
- JSON 政策概觀

錯誤 – 缺少陳述式

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing statement: Add a statement to the policy
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a statement to the policy"
```

解決錯誤

JSON 政策必須包含陳述式。

相關用語

- JSON 政策元素：

錯誤 – 如果存在，則為空

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Null with if exists: The Null condition operator cannot be used with the IfExists
 suffix. Update the operator or the suffix.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The Null condition operator cannot be used with the IfExists suffix.
 Update the operator or the suffix."
```

解決錯誤

除 `Null` 條件運算子外，您可在任何條件運算子名稱的結尾新增 `IfExists`。使用 `Null` 條件運算子檢查授權時是否有條件鍵。使用 `...ifExists` 來表示「如果請求的內容中存在政策鍵，則依照政策所述來處理鍵。如果該鍵不存在，則評估條件元素為 true。」

相關用語

- ... IfExists 條件運算子
- Null 條件運算子
- 條件元素

錯誤 – SCP 語法錯誤動作萬用字元

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
SCP syntax error action wildcard: SCP actions can include wildcards (*) only at the end
 of a string. Update {{action}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "SCP actions can include wildcards (*) only at the end of a string.
 Update {{action}}."
```

解決錯誤

AWS Organizations 服務控制原則 (SCP) 支援指定`Action`或`NotAction`元素中的值。不過，這些值只能在字串結尾包含萬用字元 (*)。這表示您可指定 `iam:Get*` 而不是 `iam:*role`。

若要指定多個動作， AWS 建議您個別列出它們。

相關用語

- SCP 動作和元素 NotAction
- SCP 評估
- AWS Organizations 服務控制政策
- IAM JSON 政策元素：動作

錯誤 – SCP 語法錯誤允許條件

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
SCP syntax error allow condition: SCPs do not support the Condition element with effect
 Allow. Update the element Condition or the effect.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "SCPs do not support the Condition element with effect Allow. Update
 the element Condition or the effect."
```

解決錯誤

AWS Organizations 只有在您使`"Effect": "Deny"`用時，服務控制原則 (SCP) 才支援在`Condition`元素中指定值。

若要只允許單一動作，除了您使用 `...NotEquals` 版本的條件運算子進行指定的情況以外，您可以拒絕存取所有項目。這會否定運算子所做的比較。

相關用語

- SCP 條件元素
- SCP 評估
- AWS Organizations 服務控制政策
- 範例政策： AWS 根據要求的區域拒絕存取
- IAM JSON 政策元素：條件運算子
- IAM JSON 政策元素：條件

錯誤 — 允許 SCP 語法錯誤 NotAction

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
SCP syntax error allow NotAction: SCPs do not support NotAction with effect Allow.
 Update the element NotAction or the effect.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "SCPs do not support NotAction with effect Allow. Update the element
 NotAction or the effect."
```

解決錯誤

AWS Organizations 服務控制原則 (SCP) 不支援將`NotAction`元素與`"Effect": "Allow"`.

您必須重新撰寫邏輯，以允許動作清單，或拒絕每個未列出的動作。

相關用語

- SCP 動作和元素 NotAction

- SCP 評估
- AWS Organizations 服務控制政策
- IAM JSON 政策元素：動作

錯誤 – SCP 語法錯誤允許資源

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
SCP syntax error allow resource: SCPs do not support Resource with effect Allow. Update
 the element Resource or the effect.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "SCPs do not support Resource with effect Allow. Update the element
 Resource or the effect."
```

解決錯誤

AWS Organizations 只有在您使`"Effect": "Deny"`用時，服務控制原則 (SCP) 才支援在`Resource`元素中指定值。

您必須重寫邏輯，以允許所有資源，或拒絕列出的每個資源。

相關用語

- SCP 資源元素
- SCP 評估
- AWS Organizations 服務控制政策
- IAM JSON 政策元素：資源

錯誤 — SCP 語法錯誤 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
SCP syntax error NotResource: SCPs do not support the NotResource element. Update the
 policy to use Resource instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "SCPs do not support the NotResource element. Update the policy to
 use Resource instead."
```

解決錯誤

AWS Organizations 服務控制策略（SCP）不支持該NotResource元素。

您必須重寫邏輯，以允許所有資源，或拒絕列出的每個資源。

相關用語

- SCP 資源元素
- SCP 評估
- AWS Organizations 服務控制政策
- IAM JSON 政策元素：資源

錯誤 – SCP 語法錯誤主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
SCP syntax error principal: SCPs do not support specifying principals. Remove the
 Principal or NotPrincipal element.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "SCPs do not support specifying principals. Remove the Principal or
 NotPrincipal element."
```

解決錯誤

AWS Organizations 服務控制策略（SCP）不支持Principal或NotPrincipal元素。

您可以使用 Condition 元素中的 aws:PrincipalArn 全域條件鍵來指定 Amazon Resource Name (ARN)。

相關用語

- SCP 語法
- 主體的全域條件鍵

錯誤 – 需要唯一 Sid

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unique Sids required: Duplicate statement IDs are not supported for this policy type.
 Update the Sid value.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Duplicate statement IDs are not supported for this policy type.
 Update the Sid value."
```

解決錯誤

對於某些政策類型，陳述式 ID 必須是唯一的。Sid (陳述式 ID) 元素允許您輸入您為政策陳述式提供的可選識別符。您可以使用 SID 元素將陳述式 ID 值指派給陳述式陣列中的每個陳述式。在允許您指定 ID 元素的服務中 (例如 SQS 和 SNS)，Sid 值只是政策文件 ID 的子 ID。例如，在 IAM 中，Sid 值在 JSON 政策中必須是唯一的。

相關用語

- IAM JSON 政策元素：Sid

錯誤 - 政策中不支援的動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解決錯誤

在連接至不同資源類型的資源型政策中，部分動作在其 `Action` 元素中不受支援。例如，Amazon S3 儲存貯體政策不支援 AWS Key Management Service 動作。指定連接至資源型政策的資源類型支援的動作。

相關用語

- JSON 政策元素：動作

錯誤 – 不支援的元素組合

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported element combination: The policy elements ${element1} and ${element2} can
 not be used in the same statement. Remove one of these elements.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The policy elements ${element1} and ${element2} can not be used in
 the same statement. Remove one of these elements."
```

解決錯誤

JSON 政策元素的某些組合無法一起使用。例如，您不能在同一政策陳述式中同時使用 `Action` 與 `NotAction`。相互排斥的其他組合包括 `Principal/NotPrincipal` 和 `Resource/NotResource`。

相關用語

- IAM JSON 政策元素參考
- JSON 政策概觀

錯誤 – 不支援的全域條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported global condition key: The condition key aws:ARN is not supported. Use
 aws:PrincipalArn or aws:SourceArn instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key aws:ARN is not supported. Use aws:PrincipalArn or
 aws:SourceArn instead."
```

解決錯誤

AWS 不支持使用指定的全局條件鍵。視您的使用案例而定，您可以使用 `aws:PrincipalArn` 或 `aws:SourceArn` 全域條件鍵。例如，並非使用 `aws:ARN`，而是使用 `aws:PrincipalArn` 來將提出請求主體的 Amazon Resource Name (ARN)) 與您在政策中所指定的 ARN 進行比較。或者，使用`aws:SourceArn`全域條件金鑰將發出 service-to-service 請求的資源的 Amazon 資源名稱 (ARN) 與您在政策中指定的 ARN 進行比較。

相關用語

- AWS 全域條件內容索引鍵

錯誤 – 不支援的主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported principal: The policy type ${policy_type} does not support the Principal
 element. Remove the Principal element.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The policy type ${policy_type} does not support the Principal
 element. Remove the Principal element."
```

解決錯誤

`Principal` 元素會指定允許或拒絕存取資源的主體。您無法在以 IAM 身分為基礎的政策中使用 `Principal` 元素。您可以在 IAM 角色和以資源為基礎政策的信任政策中使用該元素。資源型政策是您直接內嵌在資源中的政策。例如，您可以在 Amazon S3 儲存貯體或 AWS KMS 金鑰中內嵌政策。

相關用語

- AWS JSON 政策元素：主體
- IAM 中的跨帳戶資源存取

錯誤 – 政策中不支援的資源 ARN

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-based policy attached to the resource type {{resourceType}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The resource ARN is not supported for the resource-based policy attached to the resource type {{resourceType}}."
```

解決錯誤

當資源型政策連接至不同的資源類型時，部分資源 ARN 在此政策的 `Resource` 元素中不受支援。例如，Amazon S3 儲存貯體政策的`Resource`元素不支援 AWS KMS ARN。指定連接至資源型政策的資源類型支援的資源 ARN。

相關用語

- JSON 政策元素：動作

錯誤 – 不支援的 Sid

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported Sid: Update the characters in the Sid element to use one of the following character types: [a-z, A-Z, 0-9]
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Update the characters in the Sid element to use one of the following character types: [a-z, A-Z, 0-9]"
```

解決錯誤

`Sid` 元素支援大寫字母，小寫字母和數字。

相關用語

- IAM JSON 政策元素：Sid

錯誤 – 主體中不支援的萬用字元

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported wildcard in principal: Wildcards (*, ?) are not supported with the
 principal key {{principal_key}}. Replace the wildcard with a valid principal value.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Wildcards (*, ?) are not supported with the principal key
 {{principal_key}}. Replace the wildcard with a valid principal value."
```

解決錯誤

`Principal` 元素結構支援使用鍵值組。政策中指定的主體值包括萬用字元 (*)。您不能在指定的主體鍵中包含萬用字元。例如，當您在 `Principal` 元素中指定使用者時，您無法使用萬用字元來表示「所有使用者」。您必須命名特定的一個或多個使用者。同樣地，當您指定擔任的角色工作階段時，您無法使用萬用字元來表示「所有工作階段」。您必須命名特定工作階段。您同樣不能使用萬用字元來符合部分名稱或 ARN。

若要解決此問題清單，請移除萬用字元，並提供更具體的主體。

相關用語

- AWS JSON 政策元素：主體

錯誤 – 變數中缺少大括號

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing brace in variable: The policy variable is missing a closing curly brace. Add }
 after the variable text.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The policy variable is missing a closing curly brace. Add } after
 the variable text."
```

解決錯誤

政策變數結構支援使用 $ 字首，後面接著一對大括號 ({ })。在 ${ } 字元內，包含要在政策中想要使用的請求中的值的名稱。

若要解決此問題清單，請新增遺失的大括號，以確定一組完整的大括號開頭和結尾已存在。

相關用語

- IAM 政策元素：變數

錯誤 – 變數中缺少引號

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing quote in variable: The policy variable default value must begin and end with a
 single quote. Add the missing quote.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The policy variable default value must begin and end with a single
 quote. Add the missing quote."
```

解決錯誤

將變數新增至政策時，您可以指定變數的預設值。如果變數不存在， AWS 會使用您提供的預設文字。

若要將預設值新增到一個變數，請以單引號 (' ') 括住預設值，並使用逗號和空格 (,) 分隔變數文字和預設值。

例如，如果主體標記為 team=yellow，他們可以存取名為 DOC-EXAMPLE-BUCKET-yellow 的 DOC-EXAMPLE-BUCKET Amazon S3 儲存貯體。具有此資源的政策可讓團隊成員存取其自己的資源，但不能存取其他團隊的資源。對於沒有團隊標籤的使用者，您可以將預設值設定為 company-wide。這些使用者只能存取 DOC-EXAMPLE-BUCKET-company-wide 儲存貯體，其中他們可以檢視廣泛的資訊，例如加入團隊的指示。

```
"Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET-${aws:PrincipalTag/team, 'company-wide'}"
```

相關用語

- IAM 政策元素：變數

錯誤 – 變數中不支援的空間

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported space in variable: A space is not supported within the policy variable
 text. Remove the space.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "A space is not supported within the policy variable text. Remove the
 space."
```

解決錯誤

政策變數結構支援使用 $ 字首，後面接著一對大括號 ({ })。在 ${ } 字元內，包含要在政策中想要使用的請求中的值的名稱。雖然您在指定預設變數時可以包含空格，但不能在變數名稱中包含空格。

相關用語

- IAM 政策元素：變數

錯誤 – 空的變數

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty variable: Empty policy variable. Remove the ${ } variable structure or provide a
 variable within the structure.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Empty policy variable. Remove the ${ } variable structure or provide
 a variable within the structure."
```

解決錯誤

政策變數結構支援使用 $ 字首，後面接著一對大括號 ({ })。在 ${ } 字元內，包含要在政策中想要使用的請求中的值的名稱。

相關用語

- IAM 政策元素：變數

錯誤 – 元素中不支援變數

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Variable unsupported in element: Policy variables are supported in the Resource and
 Condition elements. Remove the policy variable {{variable}} from this element.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Policy variables are supported in the Resource and Condition
 elements. Remove the policy variable {{variable}} from this element."
```

解決錯誤

您可以在 Resource 元素中使用政策變數，也可以在 Condition 元素中使用字串比較。

相關用語

- IAM 政策元素：變數

錯誤 – 版本中不支援變數

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Variable unsupported in version: To include variables in your policy, use the policy
 version 2012-10-17 or later.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "To include variables in your policy, use the policy version
 2012-10-17 or later."
```

解決錯誤

若要使用政策變數，必須包含 Version 元素，並將其設定為支援政策變數的版本。變數已導入版本 2012-10-17。舊版的政策語言不支援政策變數。如果您未將 Version 設定為 2012-10-17 或更高版本，則 ${aws:username} 等變數會被視為政策中的常值字串。

Version 政策元素與政策版本不同。Version 政策元素是在政策內使用，並定義政策語言的版本。政策版本會在您在 IAM 中變更客戶受管政策時建立。變更的政策不會覆寫現有的政策。反而，IAM 會建立新版本的受管政策。

相關用語

- IAM 政策元素：變數
- IAM JSON 政策元素：版本

錯誤 – 私有 IP 地址

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Private IP address: aws:SourceIp works only for public IP address ranges. The values
 for condition key aws:SourceIp include only private IP addresses and will not have the
 desired effect. Update the value to include only public IP addresses.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "aws:SourceIp works only for public IP address ranges. The values
 for condition key aws:SourceIp include only private IP addresses and will not have the
 desired effect. Update the value to include only public IP addresses."
```

解決錯誤

全域條件鍵 aws:SourceIp 僅適用於公有 IP 地址範圍。當您的政策只允許私有 IP 地址時，您會收到此錯誤。在這種情況下，條件永遠不會符合。

- aws：SourceIp 全局條件鍵
- IAM JSON 政策元素：條件

錯誤 — 私人 NotIpAddress

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Private NotIpAddress: The values for condition key aws:SourceIp include only private
 IP addresses and has no effect. aws:SourceIp works only for public IP address ranges.
 Update the value to include only public IP addresses.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The values for condition key aws:SourceIp include only private IP
 addresses and has no effect. aws:SourceIp works only for public IP address ranges.
 Update the value to include only public IP addresses."
```

解決錯誤

全域條件鍵 `aws:SourceIp` 僅適用於公有 IP 地址範圍。當您使用 `NotIpAddress` 條件運算子，並只列出私有 IP 位址時，您會收到這個錯誤。在此情況下，條件會一律符合且為無效。

- aws：SourceIp 全局條件鍵
- IAM JSON 政策元素：條件

錯誤 – 政策大小超過 SCP 配額

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Policy size exceeds SCP quota: The {{policySize}} characters in the service control
 policy (SCP) exceed the {{policySizeQuota}} character maximum for SCPs. We recommend
 that you use multiple granular policies.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{policySize}} characters in the service control policy (SCP)
 exceed the {{policySizeQuota}} character maximum for SCPs. We recommend that you use
 multiple granular policies."
```

解決錯誤

AWS Organizations 服務控制原則 (SCP) 支援指定`Action`或`NotAction`元素中的值。不過，這些值只能在字串結尾包含萬用字元 (*)。這表示您可指定 `iam:Get*` 而不是 `iam:*role`。

若要指定多個動作， AWS 建議您個別列出它們。

相關用語

- Organ AWS izations 的配額
- AWS Organizations 服務控制政策

錯誤 — 無效的服務委託人格式

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid service principal format: The service principal does not match the expected
 format. Use the format {{expectedFormat}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The service principal does not match the expected format. Use the
 format {{expectedFormat}}."
```

解決錯誤

條件鍵值組中的值必須符合已定義的服務主體格式。

服務主體是用來將許可授予給服務的識別符。您可以在 `Principal` 元素中指定服務主體，或將其作為某些全域條件鍵和服務特定鍵的值。服務主體是由每個服務定義。

服務主體的識別符包含服務名稱，而且通常採用以下全小寫字母格式：

service-name.amazonaws.com

某些服務特定鍵可能會針對服務主體使用不同的格式。例如，`kms:ViaService` 條件鍵需要下列全小寫字母格式的服務主體：

service-name.AWS_region.amazonaws.com

相關用語

- 服務主體
- AWS 全域條件索引鍵
- `kms:ViaService` 條件鍵

錯誤 – 條件中缺少標籤鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing tag key in condition: The condition key {{conditionKeyName}} must include a tag
 key to control access based on tags. Use the format {{conditionKeyName}}tag-key and
 specify a key name for tag-key.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key {{conditionKeyName}} must include a tag key to
 control access based on tags. Use the format {{conditionKeyName}}tag-key and specify a
 key name for tag-key."
```

解決錯誤

若要根據標籤控制存取，則在政策的條件元素中提供標籤資訊。

例如，若要控制對 AWS 資源的存取，請包含aws:ResourceTag條件索引鍵。此鍵需要格式 aws:ResourceTag/*tag-key*。如需指定條件中的標籤鍵 owner 和標籤值 JaneDoe，請使用下列格式。

```
"Condition": {
    "StringEquals": {"aws:ResourceTag/owner": "JaneDoe"}
}
```

相關用語

- 使用標籤控制存取權限
- 條件
- 全域條件鍵
- AWS 服務條件索引鍵

錯誤 - vpc 格式無效

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid vpc format: The VPC identifier in the condition key value is not valid. Use the
 prefix 'vpc-' followed by 8 or 17 alphanumeric characters.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The VPC identifier in the condition key value is not valid. Use the
 prefix 'vpc-' followed by 8 or 17 alphanumeric characters."
```

解決錯誤

此 aws:SourceVpc 條件鍵必須使用字首 vpc-，後跟 8 或 17 個英數字元，例如 vpc-11223344556677889 或 vpc-12345678。

相關用語

- AWS 全局條件密鑰：aws：SourceVpc

錯誤 - vpce 格式無效

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid vpce format: The VPCE identifier in the condition key value is not valid.  Use
 the prefix 'vpce-' followed by 8 or 17 alphanumeric characters.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The VPCE identifier in the condition key value is not valid.  Use
 the prefix 'vpce-' followed by 8 or 17 alphanumeric characters."
```

解決錯誤

此 aws:SourceVpce 條件鍵必須使用字首 vpce-，後跟 8 或 17 個英數字元，例如 vpce-11223344556677889 或 vpce-12345678。

相關用語

- AWS 全局條件密鑰：aws：SourceVpce

錯誤 - 不支援聯合身分主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Federated principal not supported: The policy type does not support a federated
 identity provider in the principal element. Use a supported principal.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The policy type does not support a federated identity provider in
 the principal element. Use a supported principal."
```

解決錯誤

此 Principal 元素將聯合身分主體用於連接至 IAM 角色的信任政策，以透過聯合身分提供存取權。身分政策和其他資源型曾策不支援 Principal 元素中的聯合身分提供者。例如，您無法在 Amazon S3 儲存貯體政策中使用 SAML 主體。將 Principal 元素變更為支援的主體類型。

相關用語

- 為聯合身分建立角色
- JSON 政策元素：主體

錯誤 - 條件鍵不支援的動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported action for condition key: The following actions: {{actions}} are not
 supported by the condition key {{key}}. The condition will not be evaluated for these
 actions. We recommend that you move these actions to a different statement without
 this condition key.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The following actions: {{actions}} are not supported by the
 condition key {{key}}. The condition will not be evaluated for these actions. We
 recommend that you move these actions to a different statement without this condition
 key."
```

解決錯誤

請確保政策陳述式的 Condition 元素中的條件鍵會套用至 Action 元素中的每個動作。若要確保政策有效地允許或拒絕您指定的動作，您應該將不支援的動作移至不同的陳述式，而不使用條件鍵。

> **Note**
>
> 如果 Action 元素具有包含萬用字元的動作，則 IAM Access Analyzer 不會評估這些動作是否存在此錯誤。

相關用語

- JSON 政策元素：動作

錯誤 - 政策中不支援的動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解決錯誤

在連接至不同資源類型的資源型政策中，部分動作在其 `Action` 元素中不受支援。例如，Amazon S3 儲存貯體政策不支援 AWS Key Management Service 動作。指定連接至資源型政策的資源類型支援的動作。

相關用語

- JSON 政策元素：動作

錯誤 – 政策中不支援的資源 ARN

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The resource ARN is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解決錯誤

當資源型政策連接至不同的資源類型時，部分資源 ARN 在此政策的 `Resource` 元素中不受支援。例如，Amazon S3 儲存貯體政策的`Resource`元素不支援 AWS KMS ARN。指定連接至資源型政策的資源類型支援的資源 ARN。

相關用語

- JSON 政策元素：動作

錯誤 - 服務主體不支援的條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unsupported condition key for service principal: The following condition keys are not
 supported when used with the service principal: {{conditionKeys}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The following condition keys are not supported when used with the
 service principal: {{conditionKeys}}."
```

解決錯誤

您可以使用服務主體 (服務的識別碼) AWS 服務 在以資源為基礎的政策`Principal`項目中指定。您無法將某些條件鍵與特定服務主體搭配使用。例如，您無法將 `aws:PrincipalOrgID` 條件鍵與服務主體 `cloudfront.amazonaws.com` 搭配使用。您應該移除不適用於 `Principal` 元素中服務主體的條件鍵。

相關用語

- 服務主體
- JSON 政策元素：主體

錯誤 – 角色信任政策語法錯誤：NotPrincipal

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Role trust policy syntax error notprincipal: Role trust policies do not support
 NotPrincipal. Update the policy to use a Principal element instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Role trust policies do not support NotPrincipal. Update the policy
 to use a Principal element instead."
```

解決錯誤

角色信任政策是連接至 IAM 角色的以資源為基礎的政策。信任政策會定義哪些主體實體 (帳戶、使用者、角色和聯合身分使用者) 可擔任該角色。角色信任原則不支援 NotPrincipal。更新政策以改為使用 Principal 元素。

相關用語

- JSON 政策元素：主體
- 政策元素： NotPrincipal

錯誤 – 角色信任政策不支援在主體中使用萬用字元

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Role trust policy unsupported wildcard in principal: "Principal:" "*" is not supported
 in the principal element of a role trust policy. Replace the wildcard with a valid
 principal value.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": ""Principal:" "*" is not supported in the principal element of a role
 trust policy. Replace the wildcard with a valid principal value."
```

解決錯誤

角色信任政策是連接至 IAM 角色的以資源為基礎的政策。信任政策會定義哪些主體實體 (帳戶、使用者、角色和聯合身分使用者) 可擔任該角色。在角色信任政策的 Principal 元素中不支援 "Principal:" "*"。以有效的主體值取代萬用字元。

相關用語

- JSON 政策元素：主體

錯誤 – 角色信任政策語法錯誤：Resource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Role trust policy syntax error resource: Role trust policies apply to the role
 that they are attached to. You cannot specify a resource. Remove the Resource or
 NotResource element.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Role trust policies apply to the role that they are attached to. You
 cannot specify a resource. Remove the Resource or NotResource element."
```

解決錯誤

角色信任政策是連接至 IAM 角色的以資源為基礎的政策。信任政策會定義哪些主體實體 (帳戶、使用者、角色和聯合身分使用者) 可擔任該角色。角色信任政策會套用至它們所連接的角色。您不能在角色信任政策中指定 Resource 或 NotResource 元素。移除 Resource 或 NotResource 元素。

- JSON 政策元素：Resource
- 政策元素： NotResource

錯誤 – 類型與 IP 範圍不符

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Type mismatch IP range: The condition operator {{operator}} is used with an invalid IP
 range value. Specify the IP range in standard CIDR format.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition operator {{operator}} is used with an invalid IP range
 value. Specify the IP range in standard CIDR format."
```

解決錯誤

更新文字以使用 CIDR 格式的 IP 地址條件運算子資料類型。

相關用語

- IP 地址條件運算子
- IAM JSON 政策元素：條件運算子

錯誤 – 缺少條件鍵動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing action for condition key: The {{actionName}} action must be in the action block
 to allow setting values for the condition key {{keyName}}. Add {{actionName}} to the
 action block.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{actionName}} action must be in the action block to allow
 setting values for the condition key {{keyName}}. Add {{actionName}} to the action
 block."
```

解決錯誤

除非指定的動作位於 `Action` 元素中，否則不會評估政策陳述式的 `Condition` 元素中的條件鍵。若要確保政策有效地允許或拒絕您指定的條件鍵，請將動作新增至 `Action` 元素。

相關用語

- JSON 政策元素：動作

錯誤 – 角色信任政策中的聯合主體語法無效

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid federated principal syntax in role trust policy: The principal value specifies
 a federated principal that does not match the expected format. Update the federated
 principal to a domain name or a SAML metadata ARN.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The principal value specifies a federated principal that does not
 match the expected format. Update the federated principal to a domain name or a SAML
 metadata ARN."
```

解決錯誤

主體值指定與預期格式不相符的聯合主體。將聯合主體的格式更新為有效的網域名稱或 SAML 中繼資料 ARN。

相關用語

- 聯合身分使用者和角色

錯誤 – 主體的不相符動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Mismatched action for principal: The {{actionName}} action is invalid with the
 following principal(s): {{principalNames}}. Use a SAML provider principal with
 the sts:AssumeRoleWithSAML action or use an OIDC provider principal with the
 sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use
 either of the two options.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{actionName}} action is invalid with the following
 principal(s): {{principalNames}}. Use a SAML provider principal with the
 sts:AssumeRoleWithSAML action or use an OIDC provider principal with the
 sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use
 either of the two options."
```

解決錯誤

對於 `Principal` 元素中指定的主體，政策陳述式的 `Action` 元素中指定的動作無效。例如，您無法將 SAML 提供者主體與 `sts:AssumeRoleWithWebIdentity` 動作結合使用。您應該將 SAML 提供者主體與 `sts:AssumeRoleWithSAML` 動作結合使用，或者將 OIDC 提供者主體與 `sts:AssumeRoleWithWebIdentity` 動作結合使用。

相關用語

- AssumeRoleWith薩姆爾
- AssumeRoleWithWebIdentity

錯誤 – 缺少 Roles Anywhere 信任政策的動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing action for roles anywhere trust policy: The rolesanywhere.amazonaws.com service
 principal requires the sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession
 permissions to assume a role. Add the missing permissions to the policy.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The rolesanywhere.amazonaws.com service principal requires the
 sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession permissions to assume a
 role. Add the missing permissions to the policy."
```

解決錯誤

為了讓 IAM Roles Anywhere 能夠擔任角色並提供臨時的 AWS 憑證，角色必須信任 IAM Roles Anywhere 服務主體。IAM Roles Anywhere 服務主體需要 `sts:AssumeRole`、`sts:SetSourceIdentity` 和 `sts:TagSession` 許可來擔任角色。如果缺少這些許可中的任何一個，您必須將它們新增至政策。

相關用語

- 任何 AWS Identity and Access Management 角色中的信任模型

一般警告 — 使用創建單反 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Create SLR with NotResource: Using the iam:CreateServiceLinkedRole action with
 NotResource can allow creation of unintended service-linked roles for multiple
 resources. We recommend that you specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with NotResource can
 allow creation of unintended service-linked roles for multiple resources. We recommend
 that you specify resource ARNs instead."
```

解決一般警告

此動作會`iam:CreateServiceLinkedRole`授予建立 IAM 角色的權限，以允許 AWS 服務代表您執行動作。在政策中使用 `iam:CreateServiceLinkedRole` 與 `NotResource` 元素可允許為多個資源建立非預期的服務連結角色。 AWS 建議您改為在 `Resource` 元素中指定允許的 ARN。

- CreateServiceLinkedRole 操作
- IAM 政策元素： NotResource

- IAM JSON 政策元素：資源

一般警告 — 創建單反與明星在行動和 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Create SLR with star in action and NotResource: Using an action with a wildcard(*) and
 NotResource can allow creation of unintended service-linked roles because it can allow
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using an action with a wildcard(*) and NotResource can
 allow creation of unintended service-linked roles because it can allow
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead."
```

解決一般警告

此動作會`iam:CreateServiceLinkedRole`授予建立 IAM 角色的權限，以允許 AWS 服務代表您執行動作。`Action` 中包含萬用字元 (*) 與包含 `NotResource` 元素的政策可允許為多個資源建立非預期的服務連結角色。 AWS 建議您改為在 `Resource` 元素中指定允許的 ARN。

- CreateServiceLinkedRole 操作
- IAM 政策元素： NotResource
- IAM JSON 政策元素：資源

一般警告 — 使 NotAction 用和創建單反 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Create SLR with NotAction and NotResource: Using NotAction with NotResource
 can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using NotAction with NotResource can allow creation of unintended
 service-linked roles because it allows iam:CreateServiceLinkedRole permissions on
 multiple resources. We recommend that you specify resource ARNs instead."
```

解決一般警告

此動作會iam:CreateServiceLinkedRole授予建立 IAM 角色的權限，以允許 AWS 服務代表您執行動作。使用 NotAction 元素與 NotResource 元素可允許為多個資源建立非預期的服務連結角色。 AWS 建議您重寫政策，以改為在 Resource 元素中允許有限 ARN 清單上的 iam:CreateServiceLinkedRole。您也可以將 iam:CreateServiceLinkedRole 新增到 NotAction 元素。

- CreateServiceLinkedRole 操作
- IAM 政策元素： NotAction
- IAM JSON 政策元素：動作
- IAM 政策元素： NotResource
- IAM JSON 政策元素：資源

一般警告 – 使用資源中的星號建立 SLR

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Create SLR with star in resource: Using the iam:CreateServiceLinkedRole action with
 wildcards (*) in the resource can allow creation of unintended service-linked roles.
 We recommend that you specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with wildcards (*) in
 the resource can allow creation of unintended service-linked roles. We recommend that
 you specify resource ARNs instead."
```

解決一般警告

此動作會iam:CreateServiceLinkedRole授予建立 IAM 角色的權限，以允許 AWS 服務代表您執行動作。在政策中使用 iam:CreateServiceLinkedRole 與 Resource 元素中的萬用字元 (*) 可允許為多個資源建立非預期的服務連結角色。 AWS 建議您改為在 Resource 元素中指定允許的 ARN。

- CreateServiceLinkedRole 操作
- IAM JSON 政策元素：資源

AWS 具有此一般警告的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

其中一些使用案例適用於您帳戶中的進階使用者。下列 AWS 受管理的策略提供進階使用者存取權，並授與權限，以便為任何 AWS 服務建立服務連結角色。 AWS 建議您將下列 AWS 受管政策附加到您認為是進階使用者的 IAM 身分。

- PowerUserAccess
- AlexaForBusinessFullAccess
- AWSOrganizationsServiceTrustPolicy— 此 AWS 受管理的原則會提供 AWS Organizations 服務連結角色使用的權限。此角色可讓組 Organizations 為 AWS 組織中的其他服務建立其他服務連結角色。

一般警告 – 使用動作中的星號和資源建立 SLR

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Create SLR with star in action and resource: Using wildcards (*) in the action and
 the resource can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using wildcards (*) in the action and the resource
 can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead."
```

解決一般警告

此動作會iam:CreateServiceLinkedRole授予建立 IAM 角色的權限，以允許 AWS 服務代表您執行動作。Action 和 Resource 元素中包含萬用字元 (*) 的政策可允許為多個資源建立非預期的服務連結角色。這可讓您在指定"Action": "*"、"Action": "iam:*"或"Action": "iam:Create*"時建立服務連結角色。 AWS 建議您改為在Resource元素中指定允許的 ARN。

- CreateServiceLinkedRole 操作
- IAM JSON 政策元素：動作
- IAM JSON 政策元素：資源

AWS 具有此一般警告的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

其中一些使用案例適用於您帳戶中的系統管理員。下列 AWS 受管理策略可提供系統管理員存取權，並授與建立任何 AWS 服務之服務連結角色的權限。 AWS 建議您將下列 AWS 受管政策附加到您認為是管理員的 IAM 身分。

- AdministratorAccess
- IAM FullAccess

一般警告 — 在資源中創建帶有星號的單反相機 NotAction

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Create SLR with star in resource and NotAction: Using a resource with wildcards (*)
 and NotAction can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can allow creation
 of unintended service-linked roles because it allows iam:CreateServiceLinkedRole
 permissions on all resources. We recommend that you specify resource ARNs instead."
```

解決一般警告

此動作會`iam:CreateServiceLinkedRole`授予建立 IAM 角色的權限，以允許 AWS 服務代表您執行動作。在 `Resource` 元素中包含萬用字元 (*) 的政策中使用 `NotAction` 元素可允許為多個資源建立非預期的服務連結角色。 AWS 建議您改為在 `Resource` 元素中指定允許的 ARN。您也可以將 `iam:CreateServiceLinkedRole` 新增到 `NotAction` 元素。

- CreateServiceLinkedRole 操作
- IAM 政策元素： NotAction

- IAM JSON 政策元素：動作
- IAM JSON 政策元素：資源

一般警告 — 已取代的全域條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Deprecated global condition key: We recommend that you update aws:ARN to use the newer
 condition key aws:PrincipalArn.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "We recommend that you update aws:ARN to use the newer condition key
 aws:PrincipalArn."
```

解決一般警告

政策包含已取代的全域條件鍵。更新條件鍵值組中的條件鍵，以使用支援的全域條件鍵。

- 全域條件鍵

一般警告 – 無效的日期值

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid date value: The date {{date}} might not resolve as expected. We recommend that
 you use the YYYY-MM-DD format.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The date {{date}} might not resolve as expected. We recommend that
 you use the YYYY-MM-DD format."
```

解決一般警告

Unix Epoch 時間描述了自 1970 年 1 月 1 日以來經過的時間點，減去閏秒。紀元時間可能無法解析到您所期望的確切時間。 AWS 建議您對日期和時間格式使用 W3C 標準。例如，您可以指定完整的日期，例如 YYYY-MM-DD (1997-07-16)，或者您也可以將時間附加到秒，例如 YYYY-MM-DDThh:mm:ssTZD (1997-07-16T19:20:30+01:00)。

- W3C 日期和時間格式
- IAM JSON 政策元素：版本
- aws：CurrentTime 全局條件鍵

一般警告 – 角色參考無效

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid role reference: The Principal element includes the IAM role ID {{roleid}}. We
 recommend that you use a role ARN instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The Principal element includes the IAM role ID {{roleid}}. We
 recommend that you use a role ARN instead."
```

解決一般警告

AWS 建議您為 IAM 角色指定 Amazon 資源名稱 (ARN)，而不是指定其主要 ID。IAM 儲存政策時，會將 ARN 轉換為現有角色的主體識別碼。 AWS 包括安全預防措施。如果有人刪除並重新建立角色，則會有新的 ID，且政策不會符合新角色的 ID。

- 指定主體：IAM 角色
- IAM ARN
- IAM 唯一 ID

一般警告 – 無效的使用者參考

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Invalid user reference: The Principal element includes the IAM user ID {{userid}}. We
 recommend that you use a user ARN instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The Principal element includes the IAM user ID {{userid}}. We
 recommend that you use a user ARN instead."
```

解決一般警告

AWS 建議您為 IAM 使用者指定 Amazon 資源名稱 (ARN)，而不是指定其主要 ID。IAM 儲存政策時，會將 ARN 轉換為現有使用者的主體識別碼。 AWS 包括安全預防措施。如果有人刪除並重新建立使用者，則會有新的 ID，且政策不會符合新使用者的 ID。

- 指定主體：IAM 使用者
- IAM ARN
- IAM 唯一 ID

一般警告 – 遺失版本

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing version: We recommend that you specify the Version element to help you with
 debugging permission issues.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "We recommend that you specify the Version element to help you with
 debugging permission issues."
```

解決一般警告

AWS 建議您在原則中加入選用`Version`參數。如果您未包含 Version 元素，則值預設為 `2012-10-17`，但較新功能 (例如政策變數) 將無法使用您的政策。例如，`${aws:username}` 這類變數無法辨識為變數，而是視為政策中的文字字串。

- IAM JSON 政策元素：版本

一般警告 – 建議使用獨特 Sid

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Unique Sids recommended: We recommend that you use statement IDs that are unique to
 your policy. Update the Sid value.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "We recommend that you use statement IDs that are unique to your
 policy. Update the Sid value."
```

解決一般警告

AWS 建議您使用唯一的陳述式 ID。Sid (陳述式 ID) 元素允許您輸入您為政策陳述式提供的可選識別符。您可以使用 SID 元素將陳述式 ID 值指派給陳述式陣列中的每個陳述式。

相關用語

- IAM JSON 政策元素：Sid

一般警告 – 沒有類似運算子的萬用字元

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Wildcard without like operator: Your condition value includes a * or ? character. If
 you meant to use a wildcard (*, ?), update the condition operator to include Like.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Your condition value includes a * or ? character. If you meant to
 use a wildcard (*, ?), update the condition operator to include Like."
```

解決一般警告

Condition 元素結構會要求您使用條件運算子和鍵值組。當您指定使用萬用字元 (*、?) 的條件值時，必須使用 Like 版本的條件運算子。例如，並非使用 StringEquals 字串條件運算子,而是使用 StringLike。

```
"Condition": {"StringLike": {"aws:PrincipalTag/job-category": "admin-*"}}
```

- IAM JSON 政策元素：條件運算子
- IAM JSON 政策元素：條件

AWS 具有此一般警告的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

下列 AWS 受管理的政策在其條件值中包含萬用字元，而不包含Like用於模式比對的條件運算子。使用 AWS 受管政策作為建立客戶管理政策的參考時，建 AWS 議您使用支援與萬用字元 (*、?) 模式比對的條件運算子 ，例如StringLike。

- AWSGlueConsoleSageMakerNotebookFullAccess

一般警告 – 政策大小超過身分政策配額

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Policy size exceeds identity policy quota: The {{policySize}} characters in the
 identity policy, excluding whitespace, exceed the {{policySizeQuota}} character
 maximum for inline and managed policies. We recommend that you use multiple granular
 policies.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{policySize}} characters in the identity policy, excluding
 whitespace, exceed the {{policySizeQuota}} character maximum for inline and managed
 policies. We recommend that you use multiple granular policies."
```

解決一般警告

您最多可以將 10 個受管政策連接到 IAM 身分 (使用者、使用者群組或角色)。但是，每個受管政策的大小不可超過 6,144 個字元的預設配額。在對此配額計算政策的大小時，IAM 不會計算空格數。配額 (在中 AWS也稱為限制) 是您 AWS 帳戶中資源、動作和項目的最大值。

此外，您可以新增任意數量的內嵌政策到 IAM 身分。不過，每個身分的所有內嵌政策大小總計不能超過指定的配額。

如果您的政策大於配額，您可以將政策組織成多個陳述式，並將這些陳述式群組成多個政策。

相關用語

- IAM 和 AWS STS 角色配額
- 多項陳述式和多個政策
- IAM 客戶受管政策
- JSON 政策概觀
- IAM JSON 政策文法

AWS 具有此一般警告的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

下列 AWS 受管理的策略授與許多 AWS 服務中的動作的權限，並超過策略大小上限。當您使用 AWS 受管政策做為建立受管政策的參考時，必須將政策分割為多個政策。

- ReadOnlyAccess
- AWSSupportServiceRolePolicy

一般警告 – 政策大小超過資源政策配額

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Policy size exceeds resource policy quota: The {{policySize}} characters in the
 resource policy exceed the {{policySizeQuota}} character maximum for resource
 policies. We recommend that you use multiple granular policies.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{policySize}} characters in the resource policy exceed the
 {{policySizeQuota}} character maximum for resource policies. We recommend that you use
 multiple granular policies."
```

解決一般警告

資源型政策是連接到資源 (如 Amazon S3 儲存貯體) 的 JSON 政策文件。這些政策會授予指定的主體許可，允許在該資源上執行特定的動作，並且定義資源所適用的條件。資源型政策的大小不能超過為該資源設定的配額。配額 (在中 AWS也稱為限制) 是您 AWS 帳戶中資源、動作和項目的最大值。

如果您的政策大於配額，您可以將政策組織成多個陳述式，並將這些陳述式群組成多個政策。

相關用語

- 資源型政策
- 使用 Amazon S3 儲存貯體政策
- 多項陳述式和多個政策
- JSON 政策概觀

- IAM JSON 政策文法

一般警告 – 類型不符

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Type mismatch: Use the operator type {{allowed}} instead of operator {{operator}} for
 the condition key {{key}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Use the operator type {{allowed}} instead of operator {{operator}}
 for the condition key {{key}}."
```

解決一般警告

更新文字以使用支援的條件運算子資料類型。

例如，`aws:MultiFactorAuthPresent` 全域條件鍵需要資料類型為 `Boolean` 的條件運算子。如果您提供日期或整數，則資料類型將不相符。

相關用語

- 全域條件鍵
- IAM JSON 政策元素：條件運算子

一般警告 – 類型不符布林值

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Type mismatch Boolean: Add a valid Boolean value (true or false) for the condition
 operator {{operator}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a valid Boolean value (true or false) for the condition operator
 {{operator}}."
```

解決一般警告

更新文字以使用布林值條件運算子資料類型，例如 `true` 或 `false`。

例如，`aws:MultiFactorAuthPresent` 全域條件鍵需要資料類型為 `Boolean` 的條件運算子。如果您提供日期或整數，則資料類型將不相符。

相關用語

- 布林值條件運算子
- IAM JSON 政策元素：條件運算子

一般警告 – 類型不符日期

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Type mismatch date: The date condition operator is used with an invalid value. Specify
 a valid date using YYYY-MM-DD or other ISO 8601 date/time format.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The date condition operator is used with an invalid value. Specify a
 valid date using YYYY-MM-DD or other ISO 8601 date/time format."
```

解決一般警告

更新文字以使用 `YYYY-MM-DD` 或其他 ISO 8601 日期時間格式的日期條件運算子資料類型。

相關用語

- 日期條件運算子
- IAM JSON 政策元素：條件運算子

一般警告 – 類型不符號碼

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Type mismatch number: Add a valid numeric value for the condition operator
 {{operator}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a valid numeric value for the condition operator {{operator}}."
```

解決一般警告

更新文字以使用數字條件運算子資料類型。

相關用語

- 數位條件運算子
- IAM JSON 政策元素：條件運算子

一般警告 – 類型不符字串

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Type mismatch string: Add a valid base64-encoded string value for the condition
 operator {{operator}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a valid base64-encoded string value for the condition operator
 {{operator}}."
```

解決一般警告

更新文字以使用字串條件運算子資料類型。

相關用語

- 字串條件運算子
- IAM JSON 政策元素：條件運算子

一般警告 – 建議使用特定的 github 儲存庫和分支

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Specific github repo and branch recommended: Using a wildcard (*) in
 token.actions.githubusercontent.com:sub can allow requests from more sources than
```

```
you intended. Specify the value of token.actions.githubusercontent.com:sub with the
repository and branch name.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using a wildcard (*) in token.actions.githubusercontent.com:sub
 can allow requests from more sources than you intended. Specify the value of
 token.actions.githubusercontent.com:sub with the repository and branch name."
```

解決一般警告

如果您用 GitHub 作 OIDC IdP，最佳做法是限制可擔任與 IAM IdP 相關聯角色的實體。當您在角色信任原則中包含`Condition`陳述式時，可以將角色限制為特定 GitHub 組織、存放庫或分支。您可以使用條件鍵 `token.actions.githubusercontent.com:sub` 來限制存取。建議您將條件限制為一組特定的儲存庫或分支。如果您在中使用萬用字元 (*)`token.actions.githubusercontent.com:sub`，則來自您控制範圍以外的組織或儲存庫的 GitHub 動作可以在您的 AWS 帳戶中扮演與 GitHub IAM IdP 相關聯的角色。

相關用語

- 設定 GitHub OIDC 身分識別提供者的角色

一般警告 – 政策大小超過角色信任政策配額

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Policy size exceeds role trust policy quota: The characters in the role trust policy,
 excluding whitespace, exceed the character maximum. We recommend that you request a
 role trust policy length quota increase using Service Quotas and AWS Support Center.
 If the quotas have already been increased, then you can ignore this warning.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The characters in the role trust policy, excluding whitespace,
 exceed the character maximum. We recommend that you request a role trust policy length
 quota increase using Service Quotas and AWS Support Center. If the quotas have already
 been increased, then you can ignore this warning."
```

解決一般警告

IAM 並 AWS STS 具有限制角色信任政策大小的配額。角色信任政策中的字元 (不包括空白) 超過字元上限。我們建議您使用 Service Quotas 和 AWS Support Center Console要求增加角色信任政策長度配額。

相關用語

- IAM 和 AWS STS 配額、名稱要求和字元限制

安全性警告 — 允許 NotPrincipal

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Allow with NotPrincipal: Using Allow with NotPrincipal can be overly permissive. We
 recommend that you use Principal instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using Allow with NotPrincipal can be overly permissive. We recommend
 that you use Principal instead."
```

解決安全警告

使用 "Effect": "Allow" 與 NotPrincipal 可能太過寬鬆。例如，這可以將權限授與匿名主體。AWS 建議您使用Principal元素指定需要存取的主參與者。或者，您可以允許廣泛存取，然後新增另一個使用 NotPrincipal 元素與 “Effect”: “Deny” 的陳述式。

- AWS JSON 政策元素：主體
- AWS 政策元素： NotPrincipal

安全警告 — ForAllValues 使用單一值金鑰

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
ForAllValues with single valued key: Using ForAllValues qualifier with the single-
valued condition key {{key}} can be overly permissive. We recommend that you remove
 ForAllValues:.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using ForAllValues qualifier with the single-valued condition key
 {{key}} can be overly permissive. We recommend that you remove ForAllValues:."
```

解決安全警告

AWS 建議您ForAllValues僅搭配多重值條件使用。ForAllValues 集合運算子會測試請求集每個成員的值是否為條件鍵集的子集。如果請求中每個鍵值至少符合政策中的一個值，則條件會傳回 true。如果請求中沒有鍵，或鍵值解析為 null 資料集 (例如空白字串)，則也會傳回 true。

若要了解條件是支援單一值還是多個值，請檢閱服務的動作、資源及條件鍵頁面。具有 ArrayOf 資料類型字首的條件鍵是多重值條件鍵。例如，Amazon SES 支援具有單一值 (String) 和 ArrayOfString 多重值資料類型的鍵。

- 多值內容鍵

安全性警告 — 將角色傳遞給 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Pass role with NotResource: Using the iam:PassRole action with NotResource can be
 overly permissive because it can allow iam:PassRole permissions on multiple resources.
 We recommend that you specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the iam:PassRole action with NotResource can be overly
 permissive because it can allow iam:PassRole permissions on multiple resources. We
 recommend that you specify resource ARNs instead."
```

解決安全警告

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。若要允許這樣做，您必須授與 iam:PassRole 許可給身分 (使用者、使用者群組或角色)。在原則iam:PassRole中搭配NotResource元素使用可讓您的主體存取比您預期更多的服務或功能。 AWS 建議您改為在Resource元素中指定允許的 ARN。此外，您可以使用 iam:PassedToService 條件鍵將許可降低為單一服務。

- 將角色傳遞到服務

- IAM：PassedToService
- IAM 政策元素： NotResource
- IAM JSON 政策元素：資源

安全警告 — 在行動中與明星傳遞角色 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Pass role with star in action and NotResource: Using an action with a wildcard (*) and
 NotResource can be overly permissive because it can allow iam:PassRole permissions on
 multiple resources. We recommend that you specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using an action with a wildcard (*) and NotResource can be overly
 permissive because it can allow iam:PassRole permissions on multiple resources. We
 recommend that you specify resource ARNs instead."
```

解決安全警告

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。若要允許這樣做，您必須授與 `iam:PassRole` 許可給身分 (使用者、使用者群組或角色)。包含元素的萬用字`NotResource`元 (*) `Action` 的原則可讓您的主體存取比您預期更多的服務或功能。 AWS 建議您改為在`Resource`元素中指定允許的 ARN。此外，您可以使用 `iam:PassedToService` 條件鍵將許可降低為單一服務。

- 將角色傳遞到服務
- IAM：PassedToService
- IAM 政策元素： NotResource
- IAM JSON 政策元素：資源

安全性警告 — 將角色傳遞給 NotAction 和 NotResource

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Pass role with NotAction and NotResource: Using NotAction with NotResource can
 be overly permissive because it can allow iam:PassRole permissions on multiple
 resources.. We recommend that you specify resource ARNs instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using NotAction with NotResource can be overly permissive because
 it can allow iam:PassRole permissions on multiple resources.. We recommend that you
 specify resource ARNs instead."
```

解決安全警告

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。若要允許這樣做，您必須授與 `iam:PassRole` 許可給身分 (使用者、使用者群組或角色)。使用NotAction元素並列出元NotResource素中的某些資源，可讓您的主參與者存取比您預期更多的服務或功能。 AWS 建議您改為在Resource元素中指定允許的 ARN。此外，您可以使用 `iam:PassedToService` 條件鍵將許可降低為單一服務。

- 將角色傳遞到服務
- IAM：PassedToService
- IAM 政策元素： NotAction
- IAM JSON 政策元素：動作
- IAM 政策元素： NotResource
- IAM JSON 政策元素：資源

安全性警告 – 使用資源中的星號傳遞角色

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Pass role with star in resource: Using the iam:PassRole action with wildcards (*)
 in the resource can be overly permissive because it allows iam:PassRole permissions
 on multiple resources. We recommend that you specify resource ARNs or add the
 iam:PassedToService condition key to your statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the iam:PassRole action with wildcards (*) in the resource can
 be overly permissive because it allows iam:PassRole permissions on multiple resources.
 We recommend that you specify resource ARNs or add the iam:PassedToService condition
 key to your statement."
```

解決安全警告

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。若要允許這樣做，您必須授與 `iam:PassRole` 許可給身分 (使用者、使用者群組或角色)。允許`iam:PassRole`且在元素中包含萬用字`Resource`元 (*) 的原則可讓您的主體存取比您預期更多的服務或功能。 AWS 建議您改為在`Resource`元素中指定允許的 ARN。此外，您可以使用 `iam:PassedToService` 條件鍵將許可降低為單一服務。

有些 AWS 服務會在其角色名稱中包含其服務命名空間。此政策檢查會在分析政策以產生問題清單時，將這些慣例納入考量。例如，下列資源 ARN 可能不會產生問題清單：

```
arn:aws:iam::*:role/Service*
```

- 將角色傳遞到服務
- IAM：PassedToService
- IAM JSON 政策元素：資源

AWS 具有此安全警告的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

這些使用案例其中一個適用於您帳戶中的系統管理員。下列 AWS 受管政策提供管理員存取權，並授與許可，以將任何 IAM 角色傳遞給任何服務。 AWS 建議您僅將下列 AWS 受管理政策附加至您認為是管理員的 IAM 身分。

- AdministratorAccess-Amplify

下列 AWS 受管政策包含資源中`iam:PassRole`具有萬用字元 (*) 的權限，且位於取代路徑上。針對這些原則，我們都更新了權限指引，例如建議支援使用案例的新 AWS 受管理原則。若要檢視這些政策的替代方案，請參閱指南以瞭解每個服務。

- AWSElasticBeanstalkFullAccess
- AWSElasticBeanstalkService
- AWSLambdaFullAccess
- AWSLambdaReadOnlyAccess
- AWSOpsWorksFullAccess
- AWSOpsWorksRole
- AWSDataPipelineRole

- AmazonDynamo資料庫 FullAccesswithDataPipeline
- AmazonElasticMapReduceFullAccess
- AmazonDynamo資料庫 FullAccesswithDataPipeline
- 亚马逊 ContainerServiceFullAccess

下列 AWS 受管理的策略僅提供服務連結角色的權限，這些角色可讓 AWS 服務代表您執行動作。您可以將這些政策連接到 IAM 身分。

- AWSServiceRoleForAmazonEKSNodegroup

安全性警告 – 使用動作中的星號和資源傳遞角色

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Pass role with star in action and resource: Using wildcards (*) in the action and the
 resource can be overly permissive because it allows iam:PassRole permissions on all
 resources. We recommend that you specify resource ARNs or add the iam:PassedToService
 condition key to your statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using wildcards (*) in the action and the resource can be overly
 permissive because it allows iam:PassRole permissions on all resources. We recommend
 that you specify resource ARNs or add the iam:PassedToService condition key to your
 statement."
```

解決安全警告

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。若要允許這樣做，您必須授與 `iam:PassRole` 許可給身分 (使用者、使用者群組或角色)。和元素中含有萬用字`Resource`元 (*) `Action` 的原則可讓您的主體存取比您預期更多的服務或功能。 AWS 建議您改為在`Resource`元素中指定允許的 ARN。此外，您可以使用 `iam:PassedToService` 條件鍵將許可降低為單一服務。

- 將角色傳遞到服務
- IAM：PassedToService
- IAM JSON 政策元素：動作
- IAM JSON 政策元素：資源

AWS 具有此安全警告的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

其中一些使用案例適用於您帳戶中的系統管理員。下列 AWS 受管政策提供管理員存取權，並授與許可，以將任何 IAM 角色傳遞給任何 AWS 服務。 AWS 建議您將下列 AWS 受管政策附加到您認為是管理員的 IAM 身分。

- AdministratorAccess
- IAM FullAccess

安全警告 — 在資源和資源中傳遞星號角色 NotAction

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Pass role with star in resource and NotAction: Using a resource with wildcards (*) and
 NotAction can be overly permissive because it allows iam:PassRole permissions on all
 resources. We recommend that you specify resource ARNs or add the iam:PassedToService
 condition key to your statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can be overly
 permissive because it allows iam:PassRole permissions on all resources. We recommend
 that you specify resource ARNs or add the iam:PassedToService condition key to your
 statement."
```

解決安全警告

若要設定許多 AWS 服務，您必須將 IAM 角色傳遞給服務。若要允許這樣做，您必須授與 `iam:PassRole` 許可給身分 (使用者、使用者群組或角色)。在`NotAction`元素中使用包含萬用字元 (*) 的原則中的`Resource`元素，可讓您的主體存取比您預期更多的服務或功能。 AWS 建議您改為在`Resource`元素中指定允許的 ARN。此外，您可以使用 `iam:PassedToService` 條件鍵將許可降低為單一服務。

- 將角色傳遞到服務
- IAM：PassedToService
- IAM 政策元素： NotAction
- IAM JSON 政策元素：動作

- IAM JSON 政策元素：資源

安全性警告 – 遺失配對的條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing paired condition keys: Using the condition key {{conditionKeyName}}
 can be overly permissive without also using the following condition keys:
 {{recommendedKeys}}. Condition keys like this one are more secure when paired with
 a related key. We recommend that you add the related condition keys to the same
 condition block.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the condition key {{conditionKeyName}} can be overly
 permissive without also using the following condition keys: {{recommendedKeys}}.
 Condition keys like this one are more secure when paired with a related key. We
 recommend that you add the related condition keys to the same condition block."
```

解決安全警告

當與其他相關條件鍵配對時，某些條件鍵會更安全。 AWS 建議您在與現有條件鍵相同的條件區塊中包含相關的條件鍵。這會使透過政策授與的許可更安全。

例如，您可以使用 aws:VpcSourceIp 條件鍵，將從中提出請求的 IP 地址與您在政策中指定的 IP 地址進行比較。 AWS 建議您新增相關的 aws:SourceVPC 條件鍵。這會檢查請求是否來自您在政策中指定的 VPC 和您指定的 IP 地址。

相關用語

- aws:VpcSourceIp 全域條件鍵
- aws:SourceVPC 全域條件鍵
- 全域條件鍵
- 條件元素
- JSON 政策概觀

安全性警告 – 拒絕使用不支援的服務標籤條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Deny with unsupported tag condition key for service: Using the effect Deny with the
 tag condition key {{conditionKeyName}} and actions for services with the following
 prefixes can be overly permissive: {{serviceNames}}. Actions for the listed services
 are not denied by this statement. We recommend that you move these actions to a
 different statement without this condition key.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the effect Deny with the tag condition key
 {{conditionKeyName}} and actions for services with the following prefixes can be
 overly permissive: {{serviceNames}}. Actions for the listed services are not denied
 by this statement. We recommend that you move these actions to a different statement
 without this condition key."
```

解決安全警告

在原則的Condition元素中使用不支援的標籤條件索引鍵"Effect": "Deny"可能會過於寬鬆，因為該服務的條件會遭到忽略。 AWS 建議您移除不支援條件索引鍵的服務動作，並建立另一個陳述式以拒絕存取這些動作的特定資源。

如果您使用 aws:ResourceTag 條件鍵，且其不受服務動作支援，則該鍵不會包含在要求內容中。在此案例中，Deny 陳述式中的條件一律會傳回 false，且絕不會拒絕該動作。即使已正確標記資源，也會發生這種情況。

當服務支援 aws:ResourceTag 條件鍵時，您可以使用標籤來控制對該服務之資源的存取。這稱為以屬性為基礎的存取控制 (ABAC)。不支援這些鍵的服務需要您使用以資源為基礎的存取控制 (RBAC) 來控制對資源的存取權。

> **Note**
>
> 某些服務允許支援其資源和動作子集的 aws:ResourceTag 條件鍵。IAM Access Analyzer 會傳回不支援服務動作的問題清單。例如，Amazon S3 支援 aws:ResourceTag 以取得其資源的子集。若要檢視 Amazon S3 中可用且支援 aws:ResourceTag 條件鍵的所有資源類型，請參閱服務授權參考中的 Amazon S3 定義的資源類型。

例如，假設您想要拒絕存取取消標記刪除標記為鍵值組 status=Confidential 的特定資源。還假設 AWS Lambda 允許您標記和取消標記資源，但不支持aws:ResourceTag條件鍵。若要拒絕的刪除動作， AWS App Mesh 且 AWS Backup 如果存在此標籤，請使用aws:ResourceTag條件鍵。對於

Lambda，請使用資源命名慣例，其中包含 "Confidential" 字首。然後包含一個單獨的陳述式，以防止刪除具有該命名慣例的資源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyDeleteSupported",
            "Effect": "Deny",
            "Action": [
                "appmesh:DeleteMesh",
                "backup:DeleteBackupPlan"
                ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/status": "Confidential"
                }
            }
        },
        {
            "Sid": "DenyDeleteUnsupported",
            "Effect": "Deny",
            "Action": "lambda:DeleteFunction",
            "Resource": "arn:aws:lambda:*:123456789012:function:status-Confidential*"
        }
    ]
}
```

> **Warning**
>
> 請勿使用條件運算子的... IfExists版本作為此發現項目的因應措施。這表示「如果鍵存在於請求內容中，且值相符，則拒絕動作。否則，請拒絕動作。」 在上述範例中，在 DenyDeleteSupported 陳述式中包含 lambda:DeleteFunction 動作與 StringEqualsIfExists 運算子一律會拒絕該動作。對於該動作，鍵不存在於內容中，且每次嘗試刪除該資源類型都會遭到拒絕，無論該資源是否已加上標籤。

相關用語

- 全域條件鍵

- 比較 ABAC 與 RBAC
- IAM JSON 政策元素：條件運算子
- 條件元素
- JSON 政策概觀

安全性警告 — 拒絕服務 NotAction 的不支援標籤條件金鑰

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Deny NotAction with unsupported tag condition key for service: Using the effect Deny
 with NotAction and the tag condition key {{conditionKeyName}} can be overly permissive
 because some service actions are not denied by this statement. This is because the
 condition key doesn't apply to some service actions. We recommend that you use Action
 instead of NotAction.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the effect Deny with NotAction and the tag condition key
 {{conditionKeyName}} can be overly permissive because some service actions are not
 denied by this statement. This is because the condition key doesn't apply to some
 service actions. We recommend that you use Action instead of NotAction."
```

解決安全警告

在政策的 Condition 元素中使用標籤條件鍵與元素 NotAction 和 "Effect": "Deny" 可能太過寬鬆。對於不支援條件索引鍵的服務動作，則會忽略此條件。 AWS 建議您重新撰寫邏輯以拒絕動作清單。

如果您使用 aws:ResourceTag 條件鍵與 NotAction，則不會拒絕任何不支援鍵的新或現有服務動作。 AWS 建議您明確列出您要拒絕的動作。IAM Access Analyzer 會針對所列出不支援 aws:ResourceTag 條件鍵的動作傳回個別的問題清單。如需詳細資訊，請參閱 安全性警告 – 拒絕使用不支援的服務標籤條件鍵。

當服務支援 aws:ResourceTag 條件鍵時，您可以使用標籤來控制對該服務之資源的存取。這稱為以屬性為基礎的存取控制 (ABAC)。不支援這些鍵的服務需要您使用以資源為基礎的存取控制 (RBAC) 來控制對資源的存取權。

相關用語

- 全域條件鍵
- 比較 ABAC 與 RBAC
- IAM JSON 政策元素：條件運算子
- 條件元素
- JSON 政策概觀

安全性警告 - 限制存取服務主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Restrict access to service principal: Granting access to a service principal without
 specifying a source is overly permissive. Use aws:SourceArn, aws:SourceAccount,
 aws:SourceOrgID, or aws:SourceOrgPaths condition key to grant fine-grained access.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Granting access to a service principal without specifying a source
 is overly permissive. Use aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or
 aws:SourceOrgPaths condition key to grant fine-grained access."
```

解決安全警告

您可以使用服務主體 (服務的識別碼) AWS 服務 在以資源為基礎的政策`Principal`項目中指定。授予服務主體代表您採取行動的存取權時，會限制存取。您可以使用、、或`aws:SourceOrgPaths`條件鍵來限制對特定來源的存取 `aws:SourceArn` `aws:SourceAccount`aws:SourceOrgID，例如特定資源 ARN、組織 ID 或組織路徑， AWS 帳戶以防止過於寬鬆的策略。限制存取可協助您避免名為混淆代理人問題的安全問題。

相關用語

- AWS 服務 校長
- AWS 全局條件密鑰：aws：SourceAccount
- AWS 全局條件密鑰：aws：SourceArn
- AWS 全局條件鍵：aws：SourceOrgId
- AWS 全局條件鍵：aws：SourceOrgPaths
- 混淆代理人問題

安全性警告 – 缺少 OIDC 主體的條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing condition key for oidc principal: Using an Open ID Connect principal without a
 condition can be overly permissive. Add condition keys with a prefix that matches your
 federated OIDC principals to ensure that only the intended identity provider assumes
 the role.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using an Open ID Connect principal without a condition can be
 overly permissive. Add condition keys with a prefix that matches your federated OIDC
 principals to ensure that only the intended identity provider assumes the role."
```

解決安全警告

在沒有條件的情況下使用 Open ID Connect 主體可能會過於寬鬆。新增其字首與您的聯合 OIDC 主體相符的條件鍵，以確保只有預定的身分提供者擔任該角色。

相關用語

- 建立 Web 身分的角色或 OpenID Connect 聯合身分 (主控台)

安全性警告 – 缺少 github 儲存庫條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Missing github repo condition key: Granting a federated GitHub principal permissions
 without a condition key can allow more sources to assume the role than you intended.
 Add the token.actions.githubusercontent.com:sub condition key and specify the branch
 and repository name in the value.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Granting a federated GitHub principal permissions without a
 condition key can allow more sources to assume the role than you intended. Add the
 token.actions.githubusercontent.com:sub condition key and specify the branch and
 repository name in the value."
```

解決安全警告

如果您用 GitHub 作 OIDC IdP，最佳做法是限制可擔任與 IAM IdP 相關聯角色的實體。當您在角色信任原則中包含Condition陳述式時，可以將角色限制為特定 GitHub 組織、存放庫或分支。您可以使用條件鍵 token.actions.githubusercontent.com:sub 來限制存取。建議您將條件限制為一組特定的儲存庫或分支。如果您未包含此條件，則來自您控制範圍以外的組織或儲存庫的 GitHub 動作可以在您的 AWS 帳戶中扮演與 GitHub IAM IdP 相關聯的角色。

相關用語

- 設定 GitHub OIDC 身分識別提供者的角色

建議 – 空陣列動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array action: This statement includes no actions and does not affect the policy.
 Specify actions.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "This statement includes no actions and does not affect the policy.
 Specify actions."
```

解決建議

陳述式必須包含一個 Action 或 NotAction 元素，其中包括一組動作。當元素為空時，政策陳述式不會提供任何許可。指定 Action 元素中的動作。

- IAM JSON 政策元素：動作

建議 – 空陣列條件

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array condition: There are no values for the condition key {{key}} and it does
 not affect the policy. Specify conditions.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "There are no values for the condition key {{key}} and it does not
 affect the policy. Specify conditions."
```

解決建議

選擇性的 Condition 元素結構會要求您使用條件運算子和鍵值組。當條件值為空時，條件會傳回 true，且政策陳述式不會提供任何許可。指定條件值。

- IAM JSON 政策元素：條件

建議 — 空陣列條件 ForAllValues

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array condition ForAllValues: The ForAllValues prefix with an empty condition key
 matches only if the key {{key}} is missing from the request context. To determine if
 the request context is empty, we recommend that you use the Null condition operator
 with the value of true instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The ForAllValues prefix with an empty condition key matches only
 if the key {{key}} is missing from the request context. To determine if the request
 context is empty, we recommend that you use the Null condition operator with the value
 of true instead."
```

解決建議

Condition 元素結構會要求您使用條件運算子和鍵值組。ForAllValues 集合運算子會測試請求集每個成員的值是否為條件鍵集的子集。

當您使用 ForAllValues 與空條件鍵時，只有在請求中沒有鍵時才會符合條件。 AWS 建議如果您想要測試請求內容是否為空，請改用 Null 條件運算子。

- 多值內容鍵
- Null 條件運算子
- IAM JSON 政策元素：條件

建議 — 空陣列條件 ForAnyValue

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array condition ForAnyValue: The ForAnyValue prefix with an empty condition key
 {{key}} never matches the request context and it does not affect the policy. Specify
 conditions.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The ForAnyValue prefix with an empty condition key {{key}} never
 matches the request context and it does not affect the policy. Specify conditions."
```

解決建議

`Condition` 元素結構會要求您使用條件運算子和鍵值組。`ForAnyValues` 集合運算子會測試這組請求值是否至少有一個成員符合這組條件鍵值的至少一個成員。

當您使用 `ForAnyValues` 與空條件鍵時，條件一律不會相符。這意味著該聲明對政策沒有影響。AWS 建議您重新撰寫條件。

- 多值內容鍵
- IAM JSON 政策元素：條件

建議 — 空陣列條件 IfExists

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array condition IfExists: The IfExists suffix with an empty condition key matches
 only if the key {{key}} is missing from the request context. To determine if the
 request context is empty, we recommend that you use the Null condition operator with
 the value of true instead.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The IfExists suffix with an empty condition key matches only if the
 key {{key}} is missing from the request context. To determine if the request context
 is empty, we recommend that you use the Null condition operator with the value of true
 instead."
```

解決建議

`...IfExists` 後置詞會編輯條件運算子。這表示如果政策鍵中存在於請求的內容中，則依照政策所述來處理鍵。如果該鍵不存在，則評估條件元素為 true。

當您使用 `...IfExists` 與空條件鍵時，只有在請求中沒有鍵時才會符合條件。 AWS 建議如果您想要測試請求內容是否為空，請改用 `Null` 條件運算子。

- ... IfExists 條件運算子
- IAM JSON 政策元素：條件

建議 – 空陣列主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array principal: This statement includes no principals and does not affect the
 policy. Specify principals.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "This statement includes no principals and does not affect the
 policy. Specify principals."
```

解決建議

您必須在 IAM 角色和以資源為基礎的政策中使用 `Principal` 或 `NotPrincipal` 元素。資源型政策是您直接內嵌在資源中的政策。

當您在陳述式`Principal`元素中提供空白陣列時，陳述式對原則沒有任何影響。 AWS 建議您指定應該具有資源存取權的主參與者。

- IAM JSON 政策元素：主體
- IAM 政策元素： NotPrincipal

建議 – 空陣列資源

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty array resource: This statement includes no resources and does not affect the
 policy. Specify resources.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "This statement includes no resources and does not affect the policy.
 Specify resources."
```

解決建議

陳述式必須包含 `Resource` 或 `NotResource` 元素。

當您在陳述式的資源元素中提供空白陣列時，陳述式對原則沒有任何影響。 AWS 建議您為資源指定 Amazon 資源名稱 (ARN)。

- IAM JSON 政策元素：資源
- IAM 政策元素： NotResource

建議 – 空物件條件

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty object condition: This condition block is empty and it does not affect the
 policy. Specify conditions.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "This condition block is empty and it does not affect the policy.
 Specify conditions."
```

解決建議

`Condition` 元素結構會要求您使用條件運算子和鍵值組。

當您在陳述式的條件元素中提供空物件時，陳述式對政策沒有任何影響。移除可選元素或指定條件。

- IAM JSON 政策元素：條件

建議 – 空物件主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty object principal: This statement includes no principals and does not affect the
 policy. Specify principals.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "This statement includes no principals and does not affect the
 policy. Specify principals."
```

解決建議

您必須在 IAM 角色和以資源為基礎的政策中使用 `Principal` 或 `NotPrincipal` 元素。資源型政策是您直接內嵌在資源中的政策。

當您在陳述式的`Principal`項目中提供空白物件時，陳述式對原則沒有任何影響。 AWS 建議您指定應該具有資源存取權的主參與者。

- IAM JSON 政策元素：主體
- IAM 政策元素： NotPrincipal

建議 – 空 Sid 值

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Empty Sid value: Add a value to the empty string in the Sid element.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Add a value to the empty string in the Sid element."
```

解決建議

選擇性的 `Sid` (陳述式 ID) 元素允許您輸入您為政策陳述式提供的識別碼。您可以將 `Sid` 值指派給陳述式陣列中的每個陳述式。如果您選擇使用 `Sid` 元素，您必須提供字串值。

相關用語

- IAM JSON 政策元素：Sid

建議 – 改善 IP 範圍

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Improve IP range: The non-zero bits in the IP address after the masked bits are
 ignored. Replace address with {{addr}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The non-zero bits in the IP address after the masked bits are
 ignored. Replace address with {{addr}}."
```

解決建議

IP 地址條件必須採用標準 CIDR 格式，例如 203.0.113.0/24 或 2001:DB8:1234:5678::/64。當您在遮罩位元之後加入非零位元時，條件不會考慮這些位元。 AWS 建議您使用郵件中包含的新地址。

- IP 地址條件運算子
- IAM JSON 政策元素：條件

建議 – Null 具有限定詞

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Null with qualifier: Avoid using the Null condition operator with the ForAllValues or
 ForAnyValue qualifiers because they always return a true or false respectively.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Avoid using the Null condition operator with the ForAllValues or
 ForAnyValue qualifiers because they always return a true or false respectively."
```

解決建議

在 Condition 元素中，您所建置的表達式使用條件運算子 (例如等於或小於) 來比較政策中的條件鍵與值，以及請求內容中的鍵與值。針對包含多個值的單一條件鍵請求，您必須使用 ForAllValues 或 ForAnyValue 集合運算子。

當您使用 Null 條件運算子與 ForAllValues 時，陳述式一律會傳回 true。當您將Null條件運算子搭配使用時ForAnyValue，陳述式一律會傳回false。 AWS 建議您搭配這些集合運算子使用StringLike條件運算子。

相關用語

- 多值內容鍵
- Null 條件運算子
- 條件元素

建議 – 私有 IP 地址子集

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Private IP address subset: The values for condition key aws:SourceIp include a mix
 of private and public IP addresses. The private addresses will not have the desired
 effect. aws:SourceIp works only for public IP address ranges. To define permissions
 for private IP ranges, use aws:VpcSourceIp.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private
 and public IP addresses. The private addresses will not have the desired effect.
 aws:SourceIp works only for public IP address ranges. To define permissions for
 private IP ranges, use aws:VpcSourceIp."
```

解決建議

全域條件鍵 `aws:SourceIp` 僅適用於公有 IP 地址範圍。

當您的 `Condition` 元素包含私有和公有 IP 地址的混合時，陳述式可能沒有預期的效果。您可以使用 `aws:VpcSourceIP` 來指定私有 IP 地址。

> **Note**
>
> 只有在請求來自指定 IP 地址並且透過 VPC 端點時，全域條件鍵 `aws:VpcSourceIP` 才會相符。

- aws：SourceIp 全局條件鍵
- aws：VpcSourceIp 全局條件鍵
- IP 地址條件運算子
- IAM JSON 政策元素：條件

建議-私人 NotIpAddress 子集

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Private NotIpAddress subset: The values for condition key aws:SourceIp include a mix of
 private and public IP addresses. The private addresses have no effect. aws:SourceIp
 works only for public IP address ranges. To define permissions for private IP ranges,
 use aws:VpcSourceIp.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private
 and public IP addresses. The private addresses have no effect. aws:SourceIp works
 only for public IP address ranges. To define permissions for private IP ranges, use
 aws:VpcSourceIp."
```

解決建議

全域條件鍵 `aws:SourceIp` 僅適用於公有 IP 地址範圍。

當您的 `Condition` 元素包含 `NotIpAddress` 條件運算子與私有和公有 IP 地址的混合時，陳述式可能沒有預期的效果。每個未在政策中指定的公有 IP 地址都會相符。沒有任何私有 IP 地址會相符。為了達到這個效果，您可以使用 `NotIpAddress` 與 `aws:VpcSourceIP`，然後指定不應相符的私有 IP 地址。

- aws：SourceIp 全局條件鍵
- aws：VpcSourceIp 全局條件鍵
- IP 地址條件運算子
- IAM JSON 政策元素：條件

建議 – 冗餘動作

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Redundant action: The {{redundantActionCount}} action(s) are redundant because they
 provide similar permissions. Update the policy to remove the redundant action such as:
 {{redundantAction}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{redundantActionCount}} action(s) are redundant because they
 provide similar permissions. Update the policy to remove the redundant action such as:
 {{redundantAction}}."
```

解決建議

當您在元素中使用萬用字Action元 (*) 時，您可以包含多餘的權限。 AWS 建議您檢閱原則，並僅包含您需要的權限。這可協助您移除冗餘動作。

例如，下列動作包含 iam:GetCredentialReport 動作兩次。

```
"Action": [
        "iam:Get*",
        "iam:List*",
        "iam:GetCredentialReport"
    ],
```

在此範例中，會針對以 Get 或 List 為開頭的每個 IAM 動作定義許可。當 IAM 新增其他取得或清單操作時，此政策將允許這些操作。您可能想要允許這全部的唯讀動作。iam:GetCredentialReport 動作已包含在內做為 iam:Get*。若要移除重複的許可，您可以移除 iam:GetCredentialReport。

當動作的所有內容都是冗餘時，您會收到此政策檢查的問題清單。在此範例中，如果元素包含 iam:*CredentialReport，其不被認為是冗餘。其中包括 iam:GetCredentialReport (此為冗餘) 以及 iam:GenerateCredentialReport (此非為冗餘)。移除 iam:Get* 或 iam:*CredentialReport 會變更政策的許可。

- IAM JSON 政策元素：動作

AWS 具有此建議的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

冗餘動作並不會影響政策所授與的許可。使用 AWS 受管政策作為建立客戶管理政策的參考時，建 AWS 議您從政策中移除多餘的動作。

建議 – 冗餘條件值編號

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Redundant condition value num: Multiple values in {{operator}} are redundant. Replace
 with the {{greatest/least}} single value for {{key}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Multiple values in {{operator}} are redundant. Replace with the
 {{greatest/least}} single value for {{key}}."
```

解決建議

當您針對條件鍵中的類似值使用數值條件運算子時，您可以建立導致冗餘許可的重疊。

例如，下列 Condition 元素包含多個 aws:MultiFactorAuthAge 條件，具有 1200 秒的年齡重疊。

```
"Condition": {
        "NumericLessThan": {
          "aws:MultiFactorAuthAge": [
            "2700",
            "3600"
          ]
        }
      }
```

在此範例中，如果多重要素驗證 (MFA) 完成時少於 3600 秒 (1 小時) 以前，就會定義許可。您可以移除冗餘 2700 值。

- 數位條件運算子
- IAM JSON 政策元素：條件

建議 – 冗餘資源

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Redundant resource: The {{redundantResourceCount}} resource ARN(s) are redundant
 because they reference the same resource. Review the use of wildcards (*)
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The {{redundantResourceCount}} resource ARN(s) are redundant because
 they reference the same resource. Review the use of wildcards (*)"
```

解決建議

在 Amazon 資源名稱 (ARN) 中使用萬用字元 (*) 時，您可以建立冗餘資源許可。

例如，下列 Resource 元素包含多個 ARN 與冗餘許可。

```
"Resource": [
            "arn:aws:iam::111122223333:role/jane-admin",
            "arn:aws:iam::111122223333:role/jane-s3only",
            "arn:aws:iam::111122223333:role/jane*"
        ],
```

在此範例中，會針對任何名稱開頭為 jane 的角色定義許可。您可以移除冗餘 jane-admin 和 jane-s3only ARN 而不變更結果許可。這確實會使政策為動態。其將定義任何未來角色開頭為 jane 的許可。如果政策的目的是允許存取靜態數量的角色，則移除最後一個 ARN，並僅列出應定義的 ARN。

- IAM JSON 政策元素：資源

AWS 具有此建議的受管理策略

AWS 受管理的原則可讓您根據一般 AWS 使用案例指派權限來開始使用。 AWS

冗餘資源並不會影響政策所授與的許可。使用 AWS 受管政策作為建立客戶管理政策的參考時，建 AWS 議您從政策中移除多餘的資源。

建議 – 冗餘陳述式

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Redundant statement: The statements are redundant because they provide identical
 permissions. Update the policy to remove the redundant statement.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The statements are redundant because they provide identical
 permissions. Update the policy to remove the redundant statement."
```

解決建議

Statement 元素是政策的主要元素。此元素為必要。Statement 元素可包含單一陳述式，或是個別陳述式的陣列。

當您在長政策中多次包含相同的陳述式時，陳述式為冗餘。您可以移除其中一個陳述式，而不會影響政策授與的許可。當有人編輯政策時，他們可能會變更其中一個陳述式，而不會更新複本。這可能會導致超過預期的許可。

- IAM JSON 政策元素：陳述式

建議 – 服務名稱中的萬用字元

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Wildcard in service name: Avoid using wildcards (*, ?) in the service name because it
 might grant unintended access to other AWS services with similar names.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Avoid using wildcards (*, ?) in the service name because it might
 grant unintended access to other AWS services with similar names."
```

解決建議

當您在原則中包含 AWS 服務名稱時， AWS 建議您不要包含萬用字元 (*、?)。這可能會為您不想要的未來服務新增許可。例如，有十幾個 AWS 服務與他們的名字*code*中的單詞。

```
"Resource": "arn:aws:*code*::111122223333:*"
```

- IAM JSON 政策元素：資源

建議 - 使用服務不支援的標籤條件鍵允許

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Allow with unsupported tag condition key for service: Using the effect Allow with the
 tag condition key {{conditionKeyName}} and actions for services with the following
 prefixes does not affect the policy: {{serviceNames}}. Actions for the listed service
 are not allowed by this statement. We recommend that you move these actions to a
 different statement without this condition key.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the effect Allow with the tag condition key
 {{conditionKeyName}} and actions for services with the following prefixes does not
 affect the policy: {{serviceNames}}. Actions for the listed service are not allowed
 by this statement. We recommend that you move these actions to a different statement
 without this condition key."
```

解決建議

在原則的Condition元素中使用不支援的標籤條件索引鍵"Effect": "Allow"不會影響原則授與的權限，因為該服務動作會忽略該條件。 AWS 建議您移除不支援條件索引鍵之服務的動作，並建立另一個陳述式以允許存取該服務中的特定資源。

如果您使用 aws:ResourceTag 條件鍵，且其不受服務動作支援，則該鍵不會包含在要求內容中。在此案例中，Allow 陳述式中的條件一律會傳回 false，且絕不會允許該動作。即使已正確標記資源，也會發生這種情況。

當服務支援 aws:ResourceTag 條件鍵時，您可以使用標籤來控制對該服務之資源的存取。這稱為以屬性為基礎的存取控制 (ABAC)。不支援這些鍵的服務需要您使用以資源為基礎的存取控制 (RBAC) 來控制對資源的存取權。

> **Note**
>
> 某些服務允許支援其資源和動作子集的 aws:ResourceTag 條件鍵。IAM Access Analyzer 會傳回不支援服務動作的問題清單。例如，Amazon S3 支援 aws:ResourceTag 以取得其資源的子集。若要檢視 Amazon S3 中可用且支援 aws:ResourceTag 條件鍵的所有資源類型，請參閱服務授權參考中的 Amazon S3 定義的資源類型。

例如，假設您想要允許團隊成員檢視標記為鍵值組 team=BumbleBee 的特定資源詳細資訊。還假設 AWS Lambda 允許您標記資源，但不支持aws:ResourceTag條件鍵。若要允許檢視動作， AWS App Mesh 且 AWS Backup 如果此標籤存在，請使用aws:ResourceTag條件鍵。對於 Lambda，請使用資源命名慣例，其中包含團隊名稱做為字首。接著納入一個單獨的陳述式，以允許檢視採用該命名慣例的資源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowViewSupported",
            "Effect": "Allow",
            "Action": [
                "appmesh:DescribeMesh",
                "backup:GetBackupPlan"
                ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/team": "BumbleBee"
                }
            }
        },
        {
            "Sid": "AllowViewUnsupported",
            "Effect": "Allow",
            "Action": "lambda:GetFunction",
            "Resource": "arn:aws:lambda:*:123456789012:function:team-BumbleBee*"
        }
    ]
}
```

Warning

請勿使用 Not 版的條件運算子搭配 "Effect": "Allow" 做為此問題清單的解決方法。這些條件運算子提供否定相符。這表示評估條件後，結果受到否定。在上述範例中，在 AllowViewSupported 陳述式中包含 lambda:GetFunction 動作與 StringNotEquals 運算子一律允許動作，無論該資源是否已加上標籤。

請勿使用條件運算子的... IfExists版本作為此發現項目的因應措施。這表示「如果鍵存在於請求內容中，且值相符，則允許動作。否則，允許動作。」 在上述範例中，在 AllowViewSupported 陳述式中包含 lambda:GetFunction 動作與 StringEqualsIfExists 運算子一律會允許該動作。對於該動作，鍵不存在於內容中，且每次嘗試檢視該資源類型都會允許，無論該資源是否已加上標籤。

相關用語

- 全域條件鍵
- IAM JSON 政策元素：條件運算子
- 條件元素

- JSON 政策概觀

建議 — 允許服務 NotAction 使用不支援的標籤條件索引鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Allow NotAction with unsupported tag condition key for service: Using the effect Allow
 with NotAction and the tag condition key {{conditionKeyName}} allows only service
 actions that support the condition key. The condition key doesn't apply to some
 service actions. We recommend that you use Action instead of NotAction.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "Using the effect Allow with NotAction and the tag condition key
 {{conditionKeyName}} allows only service actions that support the condition key. The
 condition key doesn't apply to some service actions. We recommend that you use Action
 instead of NotAction."
```

解決建議

在政策的 `Condition` 元素中使用不受支援的標籤條件鍵與元素 `NotAction` 和 `"Effect": "Allow"` 並不會影響政策所授與的許可。對於不支援條件索引鍵的服務動作，則會忽略此條件。AWS 建議您重新撰寫邏輯以允許動作清單。

如果您使用 `aws:ResourceTag` 條件鍵與 `NotAction`，則不會允許任何不支援鍵的新或現有服務動作。 AWS 建議您明確列出您要允許的動作。IAM Access Analyzer 會針對所列出不支援 `aws:ResourceTag` 條件鍵的動作傳回個別的問題清單。如需詳細資訊，請參閱 建議 - 使用服務不支援的標籤條件鍵允許。

當服務支援 `aws:ResourceTag` 條件鍵時，您可以使用標籤來控制對該服務之資源的存取。這稱為以屬性為基礎的存取控制 (ABAC)。不支援這些鍵的服務需要您使用以資源為基礎的存取控制 (RBAC) 來控制對資源的存取權。

相關用語

- 全域條件鍵
- 比較 ABAC 與 RBAC
- IAM JSON 政策元素：條件運算子
- 條件元素
- JSON 政策概觀

建議 – 服務主體的建議條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Recommended condition key for service principal: To restrict access to the service
 principal {{servicePrincipalPrefix}} operating on your behalf, we recommend
 aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of
 {{key}}.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "To restrict access to the service principal
 {{servicePrincipalPrefix}} operating on your behalf, we recommend aws:SourceArn,
 aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of {{key}}."
```

解決建議

您可以使用服務主體 (服務的識別碼) AWS 服務 在以資源為基礎的政策`Principal`項目中指定。在授予對服務主體的存取權時，您應該使用 `aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID` 或 `aws:SourceOrgPaths` 條件鍵，而不是其他條件鍵 (例如 `aws:Referer`)。這可協助您避免名為混淆代理人問題的安全問題。

相關用語

- AWS 服務 校長
- AWS 全局條件鍵：aws：SourceAccount
- AWS 全局條件鍵：aws：SourceArn
- AWS 全局條件鍵：aws：SourceOrgId
- AWS 全局條件鍵：aws：SourceOrgPaths
- 混淆代理人問題

建議 - 政策中的不相關條件鍵

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Irrelevant condition key in policy: The condition key {{condition-key}} is not relevant
 for the {{resource-type}} policy.  Use this key in an identity-based policy to govern
 access to this resource.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The condition key {{condition-key}} is not relevant for the
 {{resource-type}} policy.  Use this key in an identity-based policy to govern access
 to this resource."
```

解決建議

部分條件鍵與資源型政策無關。例如，s3:ResourceAccount 條件鍵與連接至 Amazon S3 儲存貯體或 Amazon S3 存取點資源類型的資源型政策無關。

您應在身分型政策中使用條件鍵，以控制存取資源。

相關用語

- 以身分為基礎的政策和以資源為基礎的政策

建議 – 角色信任政策中的冗餘主體

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Redundant principal in role trust policy: The assumed-role principal
 {{redundant_principal}} is redundant with its parent role {{parent_role}}. Remove the
 assumed-role principal.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The assumed-role principal {{redundant_principal}} is redundant with
 its parent role {{parent_role}}. Remove the assumed-role principal."
```

解決建議

如果您在政策的 Principal 元素中同時指定擔任角色的主體及其父角色，則它會不允許或拒絕任何不同的許可。例如，如果您使用下列格式指定 Principal 元素，則其為冗餘主體：

```
"Principal": {
            "AWS": [
            "arn:aws:iam::AWS-account-ID:role/rolename",
            "arn:aws:iam::AWS-account-ID:assumed-role/rolename/rolesessionname"
        ]
```

我們建議移除擔任角色的主體。

相關用語

- 角色工作階段主體

建議 – 確認對象要求類型

在中 AWS Management Console，此檢查的發現項目包括下列訊息：

```
Confirm audience claim type: The 'aud' (audience) claim key identifies the recipients
 that the JSON web token is intended for. Audience claims can be multivalued or single-
valued. If the claim is multivalued, use a ForAllValues or ForAnyValue qualifier. If
 the claim is single-valued, do not use a qualifier.
```

在程式設計呼叫 AWS CLI 或 AWS API 中，此檢查的發現項目包含下列訊息：

```
"findingDetails": "The 'aud' (audience) claim key identifies the recipients that the
 JSON web token is intended for. Audience claims can be multivalued or single-valued.
 If the claim is multivalued, use a ForAllValues or ForAnyValue qualifier. If the claim
 is single-valued, do not use a qualifier."
```

解決建議

aud (對象) 要求鍵是您在向 IdP 註冊應用程式時向您發佈的應用程式的唯一識別符，可識別 JSON Web 符記適用的收件人。對象要求可以是多重值或單一值。如果要求是多重值，請使用 `ForAllValues` 或 `ForAnyValue` 條件集運算子。如果要求是單一值，請勿使用條件集運算子。

相關用語

- 建立 Web 身分的角色或 OpenID Connect 聯合身分 (主控台)
- 多值內容鍵
- 單一值與多重值的條件鍵

IAM Access Analyzer 自訂政策檢查

您可以使用 AWS Identity and Access Management Access Analyzer 自訂政策檢查，根據您指定的安全標準來驗證 IAM 政策。您可以執行下列類型的自訂原則檢查：

- 對照參考政策進行檢查：編輯政策時，您可以檢查更新版政策是否與參考政策 (例如現有政策版本) 相比，是否會授予新的存取權。當您使用 IAM 主控台中的 AWS Command Line Interface (AWS CLI)、IAM 存取分析器 API (API) 或 JSON 政策編輯器編輯政策時，可以執行此檢查。
- 檢查 IAM 動作或資源清單：您可以檢查以確保政策不允許特定的 IAM 動作或資源。如果僅指定動作，IAM Access Analyzer 會檢查政策中所有資源的動作存取權。如果只指定資源，則 IAM Access Analyzer 會檢查哪些動作可以存取指定的資源。如果同時指定了動作和資源，則 IAM Access Analyzer 會檢查哪些指定的動作可以存取指定的資源。使用 AWS CLI 或 API 建立或編輯政策時，您可以執行此檢查。
- 檢查公用存取：您可以檢查資源策略是否可以授與指定資源類型的公用存取權。您可以在使用或 API 建立或編輯政策時執行 AWS CLI 此檢查。此類型的自訂原則檢查與預覽存取不同，因為檢查不需要任何帳戶或外部存取分析器內容。存取預覽可讓您在部署資源許可之前預覽 IAM Access Analyzer 發現項目，而自訂檢查則會決定政策是否可以授與公用存取權。

每次自訂政策檢查都會收取費用。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

自訂政策檢查的運作方式

您可以對身分和資源型政策執行自訂政策檢查。自訂政策檢查不會依賴模式比對技術或檢查存取日誌，來判斷政策是否允許新存取權或指定存取權。與外部存取權調查結果類似，自訂政策檢查是以 Zelkova 為基礎。會將 IAM 政策轉換為等效的邏輯陳述式，並針對問題執行一套通用和專用的邏輯求解器 (可滿足性模數理論)。為了檢查是否有新的存取權或指定存取權，IAM Access Analyzer 會將 Zelkova 重複套用至政策。根據政策內容，查詢範圍會越來越限縮，以符合政策允許的行為類別特徵。如需可滿足性模數理論的詳細資訊，請參閱可滿足性模數理論。

在罕見情況下，IAM Access Analyzer 無法完全判斷政策陳述式是否會授予新的存取權或指定存取權。若發生這種情況，會錯誤地因為未通過自訂政策檢查而宣告誤判。IAM Access Analyzer 旨在提供全方位政策評估，並致力將誤判降到最低。此做法表示 IAM Access Analyzer 可充分保證，通過檢查就表示政策不會授予存取權。您可以檢視 IAM Access Analyzer 回應中報告的政策陳述式，以手動方式審視失敗的檢查。

檢查新存取權的參考政策範例

您可以找到參考政策的範例，並了解如何在 IAM Access Analyzer 自訂政策檢查範例儲存庫中針對新存取設定和執行自訂政策檢查 GitHub。

> **使用這些範例之前**
>
> 使用這些參考政策範例前，請執行下列操作：

- 詳閱參考政策並根據您的獨特需求自訂內容。
- 在您的環境中搭配您使用的 AWS 服務 徹底測試參考政策。

  參考政策示範如何實作和使用自訂政策檢查。他們並非闡述為完全如圖所示實作的官方 AWS 建議或最佳實務。您有責任仔細測試任何參考政策是否適合解決您環境的安全需求。
- 自訂政策檢查分析與環境無關。分析僅考慮輸入政策中包含的資訊。例如，自訂原則檢查無法檢查帳戶是否為特定 AWS 組織的成員。因此，自訂政策檢查無法根據 `aws:PrincipalOrgId` 和 `aws:PrincipalAccount` 條件索引鍵的條件索引鍵值來比較新存取權。

審視失敗的自訂政策檢查

自訂政策檢查失敗時，IAM Access Analyzer 的回應中會包含造成檢查失敗的政策陳述式之 陳述式 ID (Sid)。雖然陳述式 ID 是選用的政策元素，但建議您為每個政策陳述式都新增陳述式 ID。自訂政策檢查也會傳回陳述式索引，以協助識別檢查失敗的原因。陳述式索引遵循以零為基準的編號方式，第一個陳述式會參考為 0。有多個陳述式造成檢查失敗時，檢查一次只會傳回一個陳述式 ID。建議您修正原因中反白的陳述式，然後重新執行檢查，直到通過檢查為止。

使用自訂政策檢查驗證政策 (主控台)

在 IAM 主控台中使用 JSON 政策編輯器編輯政策時，您可以執行自訂政策檢查，此為選用步驟。您可以檢查與現有版本相比，更新版政策是否授予新的存取權。

在編輯 IAM JSON 政策時檢查是否有新的存取權

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側的導覽窗格中，選擇 Policies (政策)。
3. 在政策清單中，選擇您要編輯的政策之名稱。您可以使用搜尋方塊來篩選政策清單。
4. 選擇許可索引標籤，然後選擇編輯。
5. 選擇 JSON 選項並更新政策。
6. 在政策下方的政策驗證窗格中，選擇檢查新的存取權索引標籤，然後選擇檢查政策。如果修改後的許可會授予新存取權，該陳述式會在政策驗證窗格中反白顯示。
7. 如果您不打算授予新的存取權，請更新政策陳述式並選擇檢查政策，直到沒有偵測到新的存取權為止。

> **Note**
>
> 每次檢查新存取權都會收取費用。如需定價的詳細資訊，請參閱 IAM Access Analyzer 定價。

8. 選擇下一步。
9. 在檢視與儲存頁面上，檢視此政策中定義的許可，然後選擇儲存變更。

使用自訂原則檢查 (AWS CLI 或 API) 驗證政策

您可以從 AWS CLI 或 IAM 存取分析器 API 執行 IAM 存取分析器自訂政策檢查。

執行 IAM Access Analyzer 自訂政策檢查 (AWS CLI)

- 若要在與現有政策進行比較時，檢查更新版政策是否允許新存取權，請執行下列命令：check-no-new-access
- 若要檢查政策是否不允許指定的存取權，請執行下列命令：check-access-not-granted
- 若要檢查資源策略是否可以授與指定資源類型的公用存取權，請執行下列命令：check-no-public-access

執行 IAM Access Analyzer 自訂政策檢查 (API)

- 若要在與現有政策進行比較時，檢查更新版政策是否允許新存取權，請使用 CheckNoNewAccess API 操作。
- 若要檢查政策是否不允許指定的存取權，請使用 CheckAccessNotGranted API 操作。
- 若要檢查資源策略是否可以授與指定資源類型的公用存取權，請使用 CheckNoPublicAccessAPI 作業。

產生 IAM Access Analyzer 政策

身為管理員或開發人員，您可能會將超出其所需範圍的許可授予 IAM 實體 (使用者或角色)。IAM 提供數個選項，協助您精簡您授予的許可。其中一種選擇是產生以實體存取活動基礎的 IAM 政策。IAM Access Analyzer 會檢閱您的 AWS CloudTrail 記錄並產生政策範本，其中包含實體在指定日期範圍內使用的許可。您可以使用範本來建立具有精細許可的政策，這些許可只會授予支援特定使用案例所需的許可。

主題

政策產生的運作方式

IAM 存取分析器會分析您的 CloudTrail 事件，以識別 IAM 實體 (使用者或角色) 已使用的動作和服務。然後它會產生以該活動為基礎的 IAM 政策。當您使用產生的政策取代連接至實體的廣泛許可政策時，您可以精簡實體的許可。以下是政策產生程序的高階概觀。

- 設定產生政策範本 — 您可以指定最多 90 天的時間範圍，讓 IAM Access Analyzer 分析您的歷史 AWS CloudTrail 事件分析。您必須指定現有的服務角色或建立新的服務角色。服務角色可讓 IAM Access Analyzer 存取您上次存取的 CloudTrail 追蹤和服務資訊，以識別使用的服務和動作。您必須指定記錄帳戶事件的 CloudTrail 追蹤，才能產生策略。如需 IAM 存取分析器資 CloudTrail 料配額的詳細資訊，請參閱 IAM 存取分析器配額。
- 產生政策 — IAM 存取分析器會根據您 CloudTrail 事件中的存取活動產生政策。
- 檢閱和自訂政策 – 產生政策之後，您可以檢閱實體在指定日期範圍內所使用的服務和動作。您可以透過新增或移除許可、指定資源，以及將條件新增至政策範本，進一步自訂政策。
- 建立並連接政策 – 您可以選擇透過建立受管政策來儲存產生的政策。您可以將建立的政策連接至使用其活動來產生政策的使用者或角色。

服務與動作層級資訊

IAM Access Analyzer 產生 IAM 政策時，系統會傳回資訊以協助您進一步自訂政策。產生政策時，可以傳回兩種類別的資訊：

- 具有動作層級資訊的政策 — 對於某些 AWS 服務 (例如 Amazon EC2)，IAM Access Analyzer 可以識別 CloudTrail 事件中發現的動作，並列出其產生的政策中使用的動作。如需支援服務的清單，請參閱 IAM Access Analyzer 政策產生服務。對於某些服務，IAM Access Analyzer 會提示您將服務的動作新增至產生的政策。
- 具有服務層級資訊 – 的政策 IAM Access Analyzer 會使用最近存取的資訊來建立包含最近使用之所有服務的政策範本。使用時 AWS Management Console，我們會提示您檢閱服務並新增動作以完成政策。

如需每個服務中的動作清單，請參閱服務授權參考中的AWS 服務的動作、資源和條件金鑰。

產生政策的注意事項

產生政策之前，請先檢閱下列重要詳細資訊。

- 啟用 CloudTrail 追蹤 — 您必須為帳戶啟用 CloudTrail 追蹤，才能根據存取活動產生策略。建立 CloudTrail 追蹤時， CloudTrail 會將與追蹤相關的事件傳送到您指定的 Amazon S3 儲存貯體。若要瞭解如何建立 CloudTrail 追蹤，請參閱「AWS CloudTrail 使用者指南」中的「為您的 AWS 帳戶建立追蹤」。
- 資料事件無法使用 – IAM Access Analyzer 不會在產生的政策中識別資料事件 (例如 Amazon S3 資料事件) 的動作層級活動。
- PassRole— `iam:PassRole` 動作不會由所產生的策略追蹤 CloudTrail ，也不會包含在產生的策略中。
- 縮短政策產生時間 – 若要更快產生政策，請減少您在設定政策產生期間指定的日期範圍。
- 用 CloudTrail 於稽核 — 請勿將原則產生用於稽核目的；請 CloudTrail 改為使用。如需使用的詳細資訊 CloudTrail，請參閱使用 AWS CloudTrail. AWS STS
- 拒絕的操作 — 策略生成會審核所有 CloudTrail 事件，包括拒絕的操作。
- 一個政策 IAM 主控台 – 您可以在 IAM 主控台中一次產生一個政策。
- 在 IAM 主控台所產生政策的可用性 – 您可以在政策產生後最多 7 天內檢閱 IAM 主控台中產生的政策。7 天後，您必須產生新政策。
- 政策產生配額 – 如需有關 IAM Access Analyzer 政策產生配額的其他資訊，請參閱 IAM Access Analyzer 配額。
- Amazon S3 標準費率適用 — 當您使用政策產生功能時，IAM 存取分析器會檢閱 S3 儲存貯體中的 CloudTrail 日誌。存取 CloudTrail 記錄檔以產生原則時不會產生額外的儲存費用。 AWS 針對存放在 S3 儲存貯體中的 CloudTrail 日誌的請求和資料傳輸收取標準 Amazon S3 費率的費用。
- AWS Control Tower 支援 — 原則產生不支援 AWS Control Tower 產生原則。

產生政策所需的許可

您第一次產生政策所需的許可與您需要產生政策以供後續使用的許可不同。

首次設定

第一次產生政策時，您必須在帳戶中選擇合適的現有服務角色，或建立新的服務角色。服務角色可讓 IAM 存取分析器存取您帳戶中上次存取的資訊， CloudTrail 並提供服務。只有管理員應具有建立和設定角色所需的許可。因此，我們建議管理員在第一次設定期間建立服務角色。若要深入瞭解建立服務角色所需的權限，請參閱建立角色以將權限委派給 AWS 服務。

服務角色所需的許可

當您建立服務角色時，您可以設定該角色的兩個政策。您可以將 IAM 許可政策連接至指定角色可執行內容的角色。您也可以將角色信任政策連接至指定可以使用該角色之主體的角色。

第一個範例政策顯示產生政策所需之服務角色的許可政策。第二個範例政策顯示服務角色所需的角色信任政策。您可以使用這些原則來協助您在使用 AWS API 或 AWS CLI 產生政策時建立服務角色。當您使用 IAM 主控台建立服務角色作為政策產生程序的一部分時，我們會為您產生這些政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudtrail:GetTrail",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetServiceLastAccessedDetails",
                "iam:GenerateServiceLastAccessedDetails"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
            ]
        }
    ]
}
```

下列範例政策顯示具有允許 IAM Access Analyzer 擔任角色之許可的角色信任政策。

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

後續使用

若要在中產生政策 AWS Management Console，IAM 使用者必須具有許可政策，允許他們將用於產生政策的服務角色傳遞至 IAM Access Analyzer。 `iam:PassRole`通常伴隨著以`iam:GetRole`便用戶可以獲得要傳遞的角色的詳細信息。在這個範例中，使用者可以僅傳遞存在於指定的帳戶，名稱開頭為 `AccessAnalyzerMonitorServiceRole*` 的角色。若要進一步了解如何將 IAM 角色傳遞給 AWS 服務，請參閱授與使用者權限以將角色傳遞給 AWS 服務。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/service-role/
AccessAnalyzerMonitorServiceRole*"
```

```
    }
  ]
}
```

您還必須具有下列 IAM 存取分析器許可，才能在 AWS Management Console、 AWS API 中產生政策，或 AWS CLI 如下列政策聲明所示。

```
{
  "Sid": "AllowUserToGeneratePolicy",
  "Effect": "Allow",
  "Action": [
    "access-analyzer:CancelPolicyGeneration",
    "access-analyzer:GetGeneratedPolicy",
    "access-analyzer:ListPolicyGenerations",
    "access-analyzer:StartPolicyGeneration"
  ],
  "Resource": "*"
}
```

適用於首次及後續使用

當您使用產生策略時，您必須具有列出帳戶中 CloudTrail 追蹤的`cloudtrail:ListTrails`權限，如下列政策聲明所示。 AWS Management Console

```
{
  "Sid": "AllowUserToListTrails",
  "Effect": "Allow",
  "Action": [
    "CloudTrail:ListTrails"
  ],
  "Resource": "*"
}
```

根據 CloudTrail 活動 (主控台) 產生策略

您可以為 IAM 使用者或角色產生政策。

步驟 1：根據 CloudTrail 活動產生策略

下列程序說明如何使用 AWS Management Console產生角色的政策。

產生 IAM 角色的政策

1. 登入 AWS Management Console 並開啟 IAM 主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在左側導覽窗格中，選擇 Roles (角色)。

> **Note**
>
> 根據 IAM 使用者之活動產生政策的步驟幾乎相同。若要這樣做，請選擇 Users (使用者) 而非 Roles (角色)。

3. 在帳戶中的角色清單中，選擇您要用來產生政策之活動的角色名稱。
4. 在 [權限] 索引標籤的 [根據 CloudTrail 事件產生原則] 區段中，選擇 [產生策略]。
5. 在 [產生政策] 頁面上，指定您希望 IAM Access Analyzer 分析事件以針對角色採取的動作來分析 CloudTrail 事件的期間。您可以選擇最多 90 天的範圍。我們建議您選擇最短的時間間隔，以減少政策產生時間。
6. 在 [存CloudTrail 取] 區段中，選擇適當的現有角色或建立新角色 (如果不存在適當的角色)。該角色授予 IAM Access Analyzer 許可，以代表您存取您的 CloudTrail 資料，以檢閱存取活動以識別已使用的服務和動作。若要進一步了解此角色所需的許可，請參閱 產生政策所需的許可。
7. 在「要分析的CloudTrail 追蹤」區段中，指定記錄帳戶事件的 CloudTrail 追蹤。

   如果您選擇將記錄儲存在其他帳戶中的 CloudTrail 追蹤，則會顯示有關跨帳戶存取的資訊方塊。跨帳戶存取需要額外的設定。如需進一步了解，請參閱本主題稍後的 Choose a role for cross-account access。
8. 選擇 Generate policy (產生政策)。
9. 當正在產生政策時，您會返回 Permissions (許可) 索引標籤上的 Roles Summary (角色摘要) 頁面。等待 Policy request details (政策請求詳細資訊) 區段中的狀態顯示 Success (成功)，然後選擇 View generated policy (檢視產生的政策)。您可以在最多七天內檢視產生的政策。如果您產生另一個政策，則現有政策會被您產生的新政策取代。

步驟 2：檢閱許可並為所使用之服務新增動作

檢閱 IAM Access Analyzer 識別出之角色所使用的服務和動作。您可以針對所產生的政策範本中使用的任何服務新增動作。

1. 請檢閱下列各節：

- 在 Review permissions (檢閱許可) 頁面上，檢閱 Actions included in the generated policy (產生政策中所包含的動作) 清單。清單會顯示 IAM Access Analyzer 識別的在指定日期範圍內由角色使用的服務和動作。
- Services used (所使用的服務) 區段會顯示 IAM Access Analyzer 識別的在指定日期範圍中角色所使用的其他服務。使用哪些動作的相關資訊可能不適用於本節所列的服務。使用列出的每個服務的功能表，手動選擇您要包含在政策中的動作。

2. 完成新增動作後，請選擇 Next (下一步)。

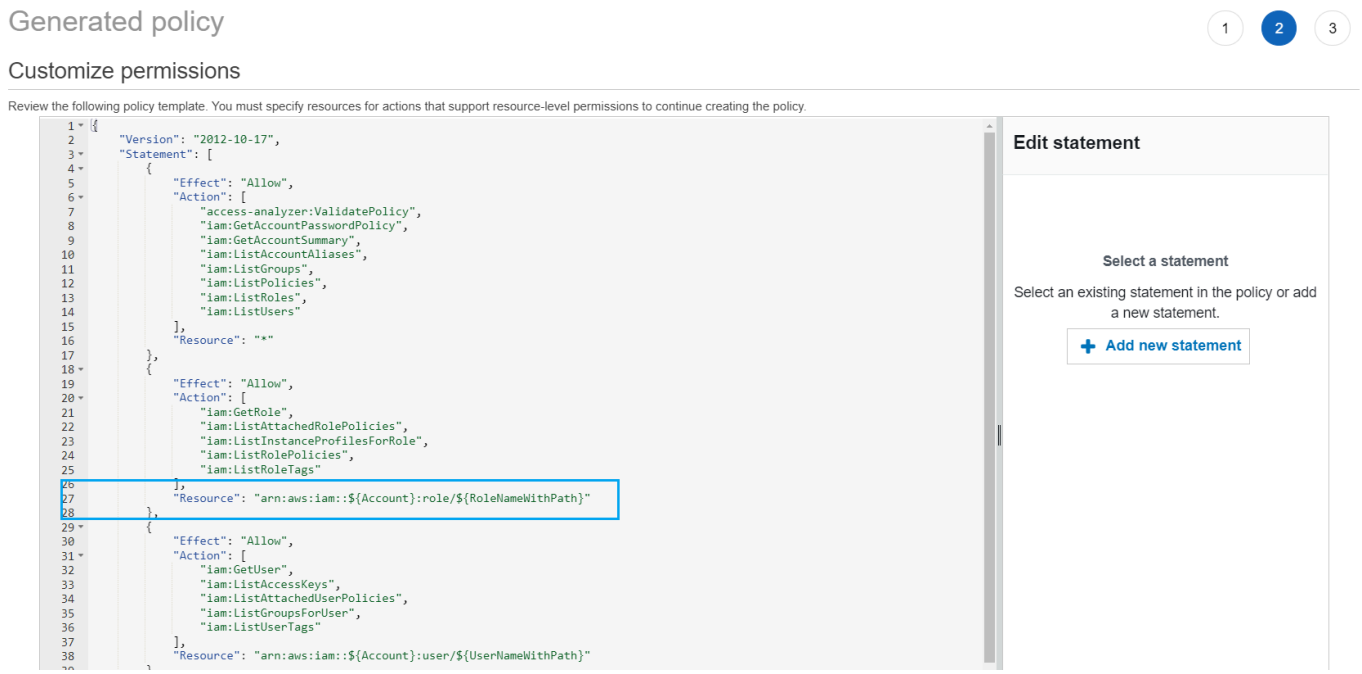
步驟 3：進一步自訂產生的政策

您可以透過新增或移除許可或指定資源，進一步自訂政策。

自訂產生的政策

1. 更新政策範本。政策範本包含支援資源層級許可之動作的資源 ARN 預留位置，如下圖所示。資源層級許可能夠讓您指定使用者可執行動作的資源。建議您使用 ARN 在政策中為支援資源層級許可的動作指定個別資源。您可以將預留位置資源 ARN 取代為使用案例的有效資源 ARN。

   如果動作不支援資源層級許可，您必須使用萬用字元 (*) 來指定該動作可以影響所有資源。若要了解哪些 AWS 服務支援資源層級許可，請參閱使用 IAM 的AWS 服務。如需每項服務中的動作清單，以及了解哪些動作支援資源層級許可，請參閱 AWS 服務的動作、資源和條件索引鍵。



2. (選用) 在範本中新增、修改或移除 JSON 政策陳述式。若要進一步了解如何撰寫 JSON 政策，請參閱建立 IAM 政策 (主控台)。
3. 完成自訂政策範本後，您有下列選項：
   - (選用) 您可以複製範本中的 JSON，以便在 Generated policy (產生的政策) 頁面之外獨立使用。例如，如果您想要使用 JSON 在不同的帳戶中建立政策。如果範本中的政策超過 JSON 政策的 6,144 個字元限制，則政策會分割成多個政策。
   - 選擇 Next (下一步) 以檢閱並在同一個帳戶中建立受管政策。

步驟 4：檢閱並建立受管政策

如果您有建立和連接 IAM 政策的許可，則可以從產生的政策中建立受管政策。然後，您可以將政策連接至帳戶中的使用者或角色。

檢閱和建立政策

1. 在 Review and create managed policy (檢閱並建立受管政策) 頁面上，為您正在建立的政策輸入 Name (名稱) 與 Description (描述) (選用)。
2. (選用) 在 Summary (摘要) 區段中，您可以檢閱將包含在政策中的許可。
3. (選用) 藉由連接標籤作為鍵值組，將中繼資料新增至政策。如需在 IAM 中使用標籤的詳細資訊，請參閱標記 IAM 資源。
4. 完成後，請執行下列其中一項動作：
   - 您可以將新政策直接連接至用來產生政策的角色。若要這麼做，請在頁面底部附近選取 [將原則附加至***YourRole##***] 旁邊的核取方塊。然後選擇 Create and attach policy (建立並連接政策)。
   - 否則，請選擇 Create policy (建立政策)。您可以在 IAM 主控台的 Policies (政策) 導覽窗格的政策清單中找到您建立的政策。
5. 您可以將您建立的政策連接至帳戶中的實體。連接政策之後，您可以移除可能連接至實體的任何其他過於廣泛的政策。若要了解如何連接受管政策，請參閱新增 IAM 身分許可 (主控台)。

使用其他帳戶中的 AWS CloudTrail 資料產生策略

您可以創建將數據存儲在中央帳戶中的 CloudTrail 跟踪以簡化管理活動。例如，您可以使 AWS Organizations 用建立追蹤，以記錄該組織中所有事件 AWS 帳戶 的所有事件。這條追蹤屬於一個中央帳戶。如果您想要為帳戶中的使用者或角色產生政策，而該策略與儲存 CloudTrail 記錄資料的帳戶不

同，則必須授與跨帳戶存取權限。為此，您需要一個角色和值區政策，以授予 IAM Access Analyzer 權限給您的 CloudTrail 日誌。如需建立 Organizations 追蹤的詳細資訊，請參閱為組織建立追蹤。

在此範例中，假設您想要為帳戶 A 中的使用者或角色產生策略。帳戶 A 中的 CloudTrail 追蹤會將 CloudTrail 記錄儲存在帳戶 B 中的值區中儲存在值區中，才能產生策略，您必須先進行下列更新：

1. 選擇現有角色，或建立新的服務角色，以授予 IAM Access Analyzer 存取帳戶 B (儲存 CloudTrail 記錄的位置) 中儲存貯體的存取權。
2. 在帳戶 B 中確認 Amazon S3 儲存貯體物件擁有權和儲存貯體許可政策，以便 IAM Access Analyzer 可存取儲存貯體中的物件。

步驟 1：選擇或建立跨帳戶存取的角色

- 在產生政策畫面中，如果您的帳戶中存在具有必要許可的角色，則會為您預先選取使用現有的角色選項。否則，請選擇建立並使用新的服務角色。新角色用於授與 IAM 存取分析器存取權限，存取帳戶 B 中的 CloudTrail 記錄。

步驟 2：在帳戶 B 中確認或更新您的 Amazon S3 儲存貯體組態

1. 登入 AWS Management Console 並開啟 Amazon S3 主控台，網址為 https://console.aws.amazon.com/s3/。
2. 在「值區」清單中，選擇儲存 CloudTrail 追蹤記錄的值區名稱。
3. 選擇 Permissions (許可) 索引標籤，然後轉至 Object Ownership (物件擁有權) 區段。

   使用 Amazon S3 物件擁有權儲存貯體設定，可控制您上傳至儲存貯體的物件的擁有權。根據預設，當其他 AWS 帳戶 上傳物件至您的值區時，上傳帳戶會擁有這些物件。若要產生政策，儲存貯體擁有者必須擁有儲存貯體中的所有物件。根據您的 ACL 使用案例，您可能需要變更儲存貯體的 Object Ownership (物件擁有權) 設定。將 Object Ownership (物件擁有權) 設定為下列其中一個選項。

   - Bucket owner enforced (儲存貯體擁有者強制執行) (建議)
   - Bucket owner preferred (儲存貯體擁有者偏好)

> **Important**
>
> 若要成功產生政策，儲存貯體中的物件必須由儲存貯體擁有者擁有。如果您選擇使用儲存貯體擁有者偏好，您只能在物件擁有權變更後的期間內產生政策。

如需有關 Amazon S3 中物件擁有權的詳細資訊，請參閱《Amazon S3 使用者指南》中的控制物件的擁有權並停用儲存貯體的 ACL。

4. 將許可新增到帳戶 B 中的 Amazon S3 儲存貯體政策，以允許存取帳戶 A 中的角色。

下列範例政策允許名為 DOC-EXAMPLE-BUCKET 之儲存貯體的 ListBucket 和 GetObject。如果存取儲存貯體的角色屬於貴組織中的帳戶，且名稱開頭為 AccessAnalyzerMonitorServiceRole，則允許存取。在Resource元素Condition中使用aws:PrincipalArn作為可確保角色只DOC-EXAMPLE-BUCKET有在帳戶屬於帳戶 A 時，才能存取帳戶的活動。您可以使optional-prefix用值區名稱、值區的選擇性字首以及organization-id組織 ID 來取代。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyGenerationBucketPolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/optional-prefix/AWSLogs/organization-id/
${aws:PrincipalAccount}/*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "organization-id"
        },
```

```
        "StringLike": {
          "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-
role/AccessAnalyzerMonitorServiceRole*"
        }
      }
    }
  ]
}
```

5. 如果您使用加密日誌 AWS KMS，請在存放日 CloudTrail 誌的帳戶中更新 AWS KMS 金鑰政策，以授予 IAM Access Analyzer 存取權以使用您的金鑰，如以下政策範例所示。將 CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN 取代為您追蹤的 ARN 以及將 organization-id 取代為貴組織 ID。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "kms:Decrypt",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:cloudtrail:arn":
 "CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN",
          "aws:PrincipalOrgID": "organization-id"
        },
        "StringLike": {
          "kms:ViaService": [
            "access-analyzer.*.amazonaws.com",
            "s3.*.amazonaws.com"
          ]
          "aws:PrincipalArn": "arn:aws:iam::${aws:PrincipalAccount}:role/service-
role/AccessAnalyzerMonitorServiceRole*"
        }
      }
    }
  ]
}
```

根據 CloudTrail活動 (AWS CLI) 產生政策

您可以使用以下列命令藉由 AWS CLI產生政策。

產生政策

- aws 訪問分析器 start-policy-generation

檢視產生的政策

- aws 訪問分析器 get-generated-policy

取消政策產生請求

- aws 訪問分析器 cancel-policy-generation

檢視政策產生請求的清單

- aws 訪問分析器 list-policy-generations

根據 CloudTrail活動 (AWS API) 產生政策

您可以使用下列作業，使用 AWS API 產生政策。

產生政策

- StartPolicy世代

檢視產生的政策

- GetGenerated政策

取消政策產生請求

- CancelPolicy世代

檢視政策產生請求的清單

- ListPolicy世代

IAM Access Analyzer 政策產生服務

下表列出 IAM 存取分析器產生具有動作層級資訊之政策的 AWS 服務。如需每個服務中的動作清單，請參閱服務授權參考中AWS 服務的動作、資源和條件金鑰。

服務	服務前綴
AWS Identity and Access Management Access Analyzer	access-analyzer
AWS Account Management	帳戶
AWS Certificate Manager	acm
Amazon Managed Workflows for Apache Airflow	airflow
AWS Amplify	mplify
AWS Amplify UI 生成器	amplifyuibuilder
Amazon AppIntegrations	app-integrations
AWS AppConfig	appconfig
Amazon AppFlow	appflow
AWS 應用程式成本分析工具	application-cost-profiler
Amazon CloudWatch 應用洞察	applicationinsights
AWS App Mesh	appmesh
Amazon AppStream 2.0	appstream
AWS AppSync	appsync
Amazon Managed Service for Prometheus	aps

服務	服務前綴
Amazon Athena	athena
AWS Audit Manager	auditmanager
AWS Auto Scaling	自動擴展
AWS Marketplace	aws-marketplace
AWS Backup	備份
AWS Batch	批次
Amazon Braket	braket
AWS Budgets	預算
AWS Cloud9	Cloud9
AWS CloudFormation	cloudformation
Amazon CloudFront	cloudfront
AWS CloudHSM	cloudhsm
Amazon CloudSearch	cloudsearch
AWS CloudTrail	cloudtrail
Amazon CloudWatch	cloudwatch
AWS CodeArtifact	codeartifact
AWS CodeDeploy	codedeploy
Amazon CodeGuru 分析器	codeguru-profiler
Amazon 評論 CodeGuru 家	codeguru-reviewer
AWS CodePipeline	codepipeline

服務	服務前綴
AWS CodeStar	codestar
AWS CodeStar 通知	codestar-notifications
Amazon Cognito 身分	cognito-identity
Amazon Cognito 使用者集區	cognito-idp
Amazon Cognito Sync	cognito-sync
Amazon Comprehend Medical	comprehendmedical
AWS Compute Optimizer	compute-optimizer
AWS Config	config
Amazon Connect	connect
AWS Cost and Usage Report	cur
AWS Glue DataBrew	databrew
AWS Data Exchange	dataexchange
AWS Data Pipeline	datapipeline
DynamoDB Accelerator	dax
AWS Device Farm	devicefarm
Amazon DevOps 大師	devops-guru
AWS Direct Connect	directconnect
Amazon Data Lifecycle Manager	dlm
AWS Database Migration Service	dms

服務	服務前綴
Amazon DocumentDB Elastic Clusters	docdb-elastic
AWS Directory Service	ds
Amazon DynamoDB	dynamodb
Amazon Elastic Block Store	ebs
Amazon Elastic Compute Cloud	ec2
Amazon Elastic Container Registry	ecr
Amazon Elastic Container Registry Public	ecr-public
Amazon Elastic Container Service	ecs
Amazon Elastic Kubernetes Service	eks
Amazon Elastic Inference	elastic-inference
Amazon ElastiCache	elasticache
AWS Elastic Beanstalk	elasticbeanstalk
Amazon Elastic File System	elasticfilesystem
Elastic Load Balancing	elasticloadbalancing
Amazon Elastic Transcoder	elastictranscoder
Amazon EMR on EKS (EMR 容器)	emr-containers
Amazon EMR Serverless	emr-serverless
Amazon OpenSearch 服務	es
Amazon EventBridge	事件
Amazon CloudWatch 顯然	evidently

服務	服務前綴
Amazon FinSpace	finspace
Amazon 數據 Firehose	firehose
AWS Fault Injection Service	fis
AWS Firewall Manager	fms
Amazon Fraud Detector	frauddetector
Amazon FSx	fsx
Amazon GameLift	gamelift
Amazon Location Service	geo
Amazon S3 Glacier	glacier
Amazon Managed Grafana	grafana
AWS IoT Greengrass	greengrass
AWS Ground Station	groundstation
Amazon GuardDuty	guardduty
AWS HealthLake	healthlake
Amazon Honeycode	honeycode
AWS Identity and Access Management	iam
AWS 身分存放區	identitystore
EC2 Image Builder	imagebuilder
Amazon Inspector Classic	inspector
Amazon Inspector	inspector2

服務	服務前綴
AWS IoT	iot
AWS IoT Analytics	iotanalytics
AWS IoT Core Device Advisor	iotdeviceadvisor
AWS IoT Events	iotevents
AWS IoT Fleet Hub	iotfleethub
AWS IoT SiteWise	iotsitewise
AWS IoT TwinMaker	iottwinmaker
AWS IoT Wireless	iotwireless
Amazon Interactive Video Service	ivs
Amazon Interactive Video Service Chat	ivschat
Amazon Managed Streaming for Apache Kafka	kafka
Amazon Managed Streaming for Kafka Connect	kafkaconnect
Amazon Kendra	kendra
Amazon Kinesis	kinesis
Amazon Kinesis Analytics V2	kinesisanalytics
AWS Key Management Service	kms
AWS Lambda	lambda
Amazon Lex	lex
AWS License Manager 訂閱管理員	license-manager-linux-subscriptions
Amazon Lightsail	lightsail

服務	服務前綴
Amazon CloudWatch 日誌	日誌
Amazon Lookout for Equipment	lookoutequipment
Amazon Lookout for Metrics	lookoutmetrics
Amazon Lookout for Vision	lookoutvision
AWS Mainframe Modernization	m2
Amazon Managed Blockchain	managedblockchain
AWS Elemental MediaConnect	mediaconnect
AWS Elemental MediaConvert	mediaconvert
AWS Elemental MediaLive	medialive
AWS Elemental MediaStore	mediastore
AWS Elemental MediaTailor	mediatailor
Amazon MemoryDB for Redis	memorydb
AWS Application Migration Service	mgn
AWS Migration Hub	mgh
AWS Migration Hub 策略建議	migrationhub-strategy
Amazon Pinpoint	mobiletargeting
Amazon MQ	mq
AWS Network Manager	networkmanager
Amazon Nimble Studio	nimble

服務	服務前綴
AWS HealthOmics	omics
AWS OpsWorks	opsworks
AWS OpsWorks CM	opsworks-cm
AWS Outposts	outposts
AWS Organizations	組織
AWS Panorama	panorama
AWS Performance Insights (績效詳情)	pi
Amazon EventBridge 管道	pipes
Amazon Polly	polly
Amazon Connect Customer Profiles	profile
Amazon QLDB	qldb
AWS Resource Access Manager	ram
AWS 資源回收筒	rbin
Amazon Relational Database Service	rds
Amazon Redshift	redshift
Amazon Redshift 資料 API	redshift-data
AWS Migration Hub Refactor Spaces	refactor-spaces
Amazon Rekognition	rekognition
AWS Resilience Hub	resiliencehub
AWS 資源總管	resource-explorer-2

服務	服務前綴
AWS Resource Groups	resource-groups
AWS RoboMaker	robomaker
AWS Identity and Access Management 任何角色	rolesanywhere
Amazon Route 53	route53
Amazon Route 53 Recovery Controls	route53-recovery-control-config
Amazon Route 53 Recovery Readiness	route53-recovery-readiness
Amazon Route 53 Resolver	route53resolver
AWS CloudWatch 朗姆酒	rum
Amazon Simple Storage Service	s3
Amazon S3 on Outposts	s3-outposts
Amazon SageMaker 地理空間	sagemaker-geospatial
Savings Plans	savingsplans
Amazon EventBridge 模式	schemas
Amazon SimpleDB	sdb
AWS Secrets Manager	secretsmanager
AWS Security Hub	securityhub
Amazon Security Lake	securitylake
AWS Serverless Application Repository	serverlessrepo
AWS Service Catalog	servicecatalog

服務	服務前綴
AWS Cloud Map	servicediscovery
Service Quotas	servicequotas
Amazon Simple Email Service	ses
AWS Shield	shield
AWS Signer	signer
AWS SimSpace Weaver	simspaceweaver
AWS Server Migration Service	sms
Amazon Pinpoint 簡訊和語音服務	sms-voice
AWS Snowball	snowball
Amazon Simple Queue Service	sqs
AWS Systems Manager	ssm
AWS Systems Manager Incident Manager	ssm-incidents
適用於 SAP 的 AWS Systems Manager	ssm-sap
AWS Step Functions	states
AWS Security Token Service	sts
Amazon Simple Workflow Service	swf
Amazon CloudWatch Synthetics	synthetics
AWS Resource Groups Tagging API	標籤
Amazon Textract	textract
Amazon Timestream	timestream

服務	服務前綴
AWS 電信網絡生成器	tnb
Amazon Transcribe	transcribe
AWS Transfer Family	傳輸
Amazon Translate	translate
Amazon Connect Voice ID	voiceid
Amazon VPC Lattice	vpc-lattice
AWS WAFV2	wafv2
AWS Well-Architected Tool	wellarchitected
Amazon Connect Wisdom	wisdom
Amazon WorkLink	worklink
Amazon WorkSpaces	工作區
AWS X-Ray	xray

IAM Access Analyzer 配額

IAM Access Analyzer 具有以下配額：

資源	預設配額	最大配額
每個區域每個 AWS 帳戶 每個分析器類型的帳戶層級分析器上限	1	1
每個區域每個 AWS 帳戶 每個分析器類型的組織層級分析器上限	5	20[1]

資源	預設配額	最大配額
每個分析器的存檔規則數量上限	100 每個封存規則的每個條件最多可擁有 20 個值。	1,000[1]
每小時每個分析器的存取預覽數目上限	1,000	1,000
AWS CloudTrail 每個策略世代處理的記錄檔	100,000	100,000
並行政策產生	1	1
原則產生 AWS CloudTrail 資料大小	25 GB	25 GB
政策產生 AWS CloudTrail 時間範圍	90 天	90 天
每天政策產生	非洲 (開普敦)：5 亞太區域 (香港)：5 歐洲 (米蘭)：5 中東 (巴林)：5 所有其他支援的區域：50 **Note** 取消的政策產生要求會套用至每日配額。	非洲 (開普敦)：5 亞太區域 (香港)：5 歐洲 (米蘭)：5 中東 (巴林)：5 所有其他支援的區域：50

[1]客戶可使用 Service Quotas 設定某些配額。

疑難排解 IAM

如果您在使用 AWS Identity and Access Management (IAM) 時遇到拒絕存取問題或類似困難，請參閱本節中的主題。

主題

- 故障診斷一般 IAM 問題
- 排查拒絕存取錯誤訊息問題
- 故障排除 IAM 政策
- 對 FIDO 安全性金鑰進行故障診斷
- IAM 角色故障診斷
- 對 IAM 和 Amazon EC2 進行故障診斷
- 對 Amazon S3 和 IAM 進行故障診斷
- 對 SAML 2.0 聯合進行疑難排解 AWS

故障診斷一般 IAM 問題

使用此處的資訊，來協助您針對使用 AWS Identity and Access Management (IAM) 時的常見問題進行診斷與修正。

問題

- 我無法登入到我的 AWS 帳戶
- 如果遺失存取金鑰
- 政策變數無法運作
- 我所做的變更不一定都會立刻生效
- 我沒有授權執行：IAM：DeleteVirtualMFA 設備
- 如何安全地建立 IAM 使用者？
- 其他資源

我無法登入到我的 AWS 帳戶

請驗證您有正確的憑證，並且正在使用正確的方法登入。如需詳細資訊，請參閱《AWS 登入 使用者指南》 中的 登入問題故障診斷。

如果遺失存取金鑰

存取金鑰包含兩個部分：

- 存取金鑰識別符。識別符是公開的，您可以在列出存取金鑰的任意 IAM 主控台中進行查看，例如使用者摘要頁面。
- 私密存取金鑰。該金鑰會在您最初建立存取金鑰對時提供。它與密碼一樣，之後無法再擷取。如果您遺失了私密存取金鑰，則必須建立新的存取金鑰對。如果您已擁有最大數量的存取金鑰，則必須先刪除現有的金鑰對，才能建立另一個。

如需詳細資訊，請參閱 重設遺失或忘記的密碼或存取金鑰 AWS。

政策變數無法運作

- 請確認包含變數的所有政策是否在政策中包含以下版本編號：`"Version": "2012-10-17"`。若沒有正確的版本編號，在評估期間不會替換這些變數。反之，只會從字面上評估這些變數。如果您包含最新的版本編號，則不包含變數的任何政策仍將具有效用。

  `Version` 政策元素與政策版本不同。`Version` 政策元素是在政策內使用，並定義政策語言的版本。另一方面，政策版本會在您在 IAM 中變更客戶受管政策時建立。變更的政策不會覆寫現有的政策。IAM 反而會建立新版本的受管政策。若要進一步了解 `Version` 政策元素，請參閱 IAM JSON 政策元素：Version。若要進一步了解政策版本，請參閱 the section called “版本控制 IAM 政策”。
- 確認您的政策變數為正確的大小寫。如需詳細資訊，請參閱 IAM 政策元素：變數與標籤。

我所做的變更不一定都會立刻生效

作為可透過全球各地資料中心的電腦存取的服務，IAM 採用了稱為最終一致性的分散式運算模式。您在 IAM (或其他 AWS 服務) 中所做的任何變更，包括在屬性型存取控制 (ABAC) 中使用的標籤，都需要一些時間才能從所有可能的端點看見。部分延遲是由在伺服器之間、複寫區域之間和全球不同地區之間傳送資料所花費的時間造成。IAM 也會使用快取以提升效能，但在某些情況下，這可能會增加時間。直到先前快取的資料逾時後，才能看到變更。

您設計的全域應用程式必須能夠處理這些可能的延遲問題。確保它們即使在某個位置所做的變更不會立即顯示在另一個位置時，仍能如預期般運作。此類變更包括建立或更新使用者、群組、角色或政策。在應用程式的關鍵、高可用性代碼路徑中，我們不建議進行此類 IAM 變更。而應在不常運作的、單獨的初始化或設定常式中進行 IAM 變更。另外，在生產工作流程套用這些變更之前，請務必確認變更已傳播完畢。

如需有關其他 AWS 服務如何受此影響的詳細資訊，請參閱下列資源：

- Amazon DynamoDB：DynamoDB 常見問答集中的 Amazon DynamoDB 的一致性模式是什麼？以及位於 Amazon DynamoDB 開發人員指南中讀取一致性。
- Amazon EC2：Amazon EC2 API 參考中的 EC2 最終一致性。
- Amazon EMR：確保在 AWS 大數據博客中 MapReduce 為 ETL 工作流程使用 Amazon S3 和 Amazon 彈性時的一致性
- Amazon Redshift： Amazon Redshift 資料庫開發人員指南中的管理資料一致性
- Amazon S3：Amazon Simple Storage Service 使用者指南中的 Amazon S3 資料一致性模式

我沒有授權執行：IAM：DeleteVirtualMFA 設備

當您嘗試為您自己或其他人指派或移除虛擬 MFA 裝置時，您可能會接收到下列錯誤：

```
User: arn:aws:iam::123456789012:user/Diego is not authorized to perform:
 iam:DeleteVirtualMFADevice on resource: arn:aws:iam::123456789012:mfa/Diego with an
 explicit deny
```

如果有人先前在 IAM 主控台中開始將虛擬 MFA 裝置指派給使用者的程序又將其取消，就可能發生此錯誤。這會在 IAM 中為該使用者建立虛擬 MFA 裝置，但從未將此裝置指派給該使用者。您必須先刪除現有的虛擬 MFA 裝置，才能使用相同的裝置名稱建立新的虛擬 MFA 裝置。

若要修正此問題，管理員不應編輯政策許可。而是，系統管理員必須使用 AWS CLI 或 AWS API 來刪除現有但未指派的虛擬 MFA 裝置。

刪除現有但未指派的虛擬 MFA 裝置

1. 檢視您帳戶中的虛擬 MFA 裝置。
   - AWS CLI: `aws iam list-virtual-mfa-devices`
   - AWS API：`ListVirtualMFADevices`
2. 在回應中，找到您嘗試修正的使用者虛擬 MFA 裝置 ARN。
3. 刪除虛擬 MFA 裝置。
   - AWS CLI: `aws iam delete-virtual-mfa-device`
   - AWS API：`DeleteVirtualMFADevice`

如何安全地建立 IAM 使用者？

如果您的員工需要存取權限 AWS，您可以選擇建立 IAM 使用者或使用 IAM 身分中心進行身份驗證。如果您使用 IAM，建 AWS 議您建立 IAM 使用者，並將登入資料安全地傳達給員工。如果您不在員工旁邊，請使用安全的工作流程將憑證傳達給員工。

使用下列工作流程可在 IAM 中安全地建立新使用者：

1. 使用 AWS Management Console建立新使用者。選擇使用自動產生的密碼授予 AWS Management Console 存取權。如有必要，請選取 Users must create a new password at next sign-in (使用者必須在下次登入時建立新密碼) 核取方塊。在使用者變更密碼之前，請勿將許可政策新增至使用者。
2. 新增使用者之後，複製新使用者的登入 URL、使用者名稱和密碼。若要檢視密碼，請選擇 Show (顯示)。
3. 使用公司內的安全通訊方法，例如電子郵件、聊天或票證系統，將密碼傳送給您的員工。另外，為使用者提供 IAM 使用者主控台連結及其使用者名稱。請員工確認他們可以成功登入，然後再授與他們許可。
4. 員工確認之後，新增他們所需的許可。安全最佳實務是新增要求使用者使用 MFA 進行身分驗證以管理其憑證的政策。如需政策範例，請參閱 AWS：允許 MFA 驗證的 IAM 使用者在安全登入資料頁面上管理自己的登入資料。

其他資源

下列資源可協助您在使用時進行疑難排解 AWS。

- AWS CloudTrail 用戶指南 — 用於 AWS CloudTrail 跟踪對該信息進行的 API 調用的歷史記錄 AWS 並將其存儲在日誌文件中。這有助於您判斷哪些使用者和帳戶存取帳戶中的資源，何時進行呼叫，請求了哪些動作等等。如需詳細資訊，請參閱 使用以下方式記錄 IAM 和 AWS STS API 呼叫 AWS CloudTrail。
- AWS 知識中心 — 尋找常見問題集和其他資源的連結，以協助您疑難排解問題。
- AWS S@@ upport 中心 — 取得技術支援。
- AWS 高級 Support 中心-獲得高級技術支持。

排查拒絕存取錯誤訊息問題

當 AWS 明確或隱含拒絕授權要求時，會出現拒絕存取錯誤。當政策包含特定 AWS 動作的Deny陳述式時，就會發生明確拒絕。如果沒有適用的 Deny 陳述式，也沒有適用的 Allow 陳述式，則會發生隱含拒絕。由於 IAM 政策預設會拒絕 IAM 主體，因此該政策必須明確允許主體執行動作。否則，政策會隱含拒絕存取。如需詳細資訊，請參閱 明確和隱含拒絕之間的差異。

如果相同政策類型的多個政策拒絕授權請求，則 AWS 不會在拒絕存取錯誤訊息中指定政策數目。如果有多個原則類型拒絕授權要求，則只會在錯誤訊息中 AWS 包含其中一種原則類型。

> **Important**
>
> 登入時遇到問題 AWS嗎？ 請確定您位在使用者類型的正確 AWS 登入頁面。如果您是 AWS 帳戶根使用者 (帳戶擁有者)，則可以 AWS 使用建立 AWS 帳戶. 如果您是 IAM 使用者，您的帳戶管理員可以為您提供可用來登入 AWS的憑證。如果您需要請求支持，請不要使用此頁面上的反饋鏈接，因為表單是由 AWS 文檔團隊收到的，而不是 AWS Support。請在 [聯絡我們] 頁面上選擇 [仍無法登入您的 AWS 帳戶]，然後選擇其中一個可用的支援選項。

當我向 AWS 服務提出請求時，我收到「訪問被拒絕」

- 檢查錯誤訊息是否包含發出拒絕存取的政策類型。例如，如果錯誤提到由於服務控制政策 (SCP) 而拒絕存取，則您可以專注於排除 SCP 問題。當您知道政策類型時，也可以針對該政策類型的政策中的特定動作，檢查是否有拒絕陳述式或缺少允許。如果錯誤訊息未提及發出拒絕存取的政策類型，請使用本節中的其餘指導方針進一步疑難排解。
- 確任您擁有呼叫所請求之動作和資源的以身分為基礎的政策許可。如果設定了任何條件，還必須在送出請求時滿足這些條件。有關查看或修改用於 IAM 使用者、群組或角色的政策的資訊，請參閱 管理 IAM 政策。
- 如果 AWS Management Console 傳回一則訊息，指出您未獲得執行動作的授權，則您必須聯絡系統管理員以尋求協助。您的管理員提供您的登入憑證或登入連結。

  以下範例錯誤的發生情境是 `mateojackson` IAM 使用者嘗試使用主控台檢視虛構 *`my-example-widget`* 資源的詳細資訊，但卻沒有虛構 `widgets:`*`GetWidget`* 許可。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 widgets:GetWidget on resource: my-example-widget
```

在此情況下，Mateo 必須請求管理員更新他的政策，允許他使用 `widgets:GetWidget` 動作存取 *`my-example-widget`* 資源。

- 您是否試著存取支援以資源為基礎的政策的服務，例如 Amazon S3，Amazon SNS 或 Amazon SQS？ 若是如此，確認政策將您指定為主體，並授與您存取權。如果您要對在您的帳戶中的服務發出請求，則您的以身分為基礎的政策或以資源為基礎的政策中之一可為您授與許可。如果您要對不同帳戶中的服務發出請求，則您的以身分為基礎的政策或以資源為基礎的政策都必須為您授與許可。若要檢視支援以資源為基礎的政策之服務，請參閱 AWS 與 IAM 搭配使用的服務。
- 如果您的政策包含搭配金鑰值組的條件，請仔細地檢閱該條件。範例包括多個服務支援的`aws:RequestTag/tag-key`全域`ResourceTag/tag-key`條件索引鍵 AWS KMS `kms:EncryptionContext:encryption_context_key`、和條件索引鍵。確定金鑰名稱不符合多個結果。由於條件金鑰名稱不區分大小寫，條件會檢查符合 `foo`、`Foo` 或 `F00` 且名為 `foo` 的金鑰。如果您的請求包含多個鍵值組，且其中有鍵名稱只有大小寫不同，則您的存取可能會意外遭拒。如需更多詳細資訊，請參閱 IAM JSON 政策元素：Condition。
- 如果您具有許可界限，請確認允許您的請求時之許可界限所使用的政策。如果您的以身分為基礎的政策允許該請求，但您的許可界限不允許，則請求會遭拒絕。許可界限會控制 IAM 主體 (使用者或角色) 可以擁有的許可上限。以資源為基礎的政策不會受到許可界限所限制。許可界限不常見。如需有關如何 AWS 評估策略的詳細資訊，請參閱政策評估邏輯。
- 如果您是手動簽署請求 (而未使用 AWS 開發套件)，請確認您已正確地簽署請求。

當我使用臨時安全憑證來發出請求時，出現「存取遭拒」

- 首先，確定您不是因為與臨時憑證無關的原因而遭拒絕存取。如需更多詳細資訊，請參閱 當我向 AWS 服務提出請求時，我收到「訪問被拒絕」。
- 要確認服務是否接受臨時安全憑證，請參閱 AWS 與 IAM 搭配使用的服務。
- 確認您的請求正確簽署且請求的格式也正確。如需詳細資訊，請參閱您的工具組文件或 搭配使用暫時憑證與 AWS 資源。
- 確認您的臨時安全憑證並未過期。如需更多詳細資訊，請參閱 IAM 中的暫時安全憑證。
- 確認 IAM 使用者或角色擁有正確許可。臨時安全性憑證的許可衍生自 IAM 使用者或角色。因此，這些許可僅限於您所擔任臨時憑證之角色所獲得授與的許可。有關如何確定臨時安全憑證的許可的更多資訊，請參閱 控制臨時安全安全憑證的許可。
- 如果您擔任角色，您的角色工作階段可能受工作階段政策限制。當您使用程式設計方式要求臨時安全登入資料時 AWS STS，您可以選擇性地傳遞內嵌或受管理的工 工作階段政策是一種進階政策，且您會在以程式設計方式建立角色的暫時憑證工作階段時，以參數方式傳遞。您可以使用 `Policy` 參

數傳遞單一 JSON 內嵌工作階段政策文件。您可以使用 PolicyArns 參數，指定多達 10 個受管工作階段政策。所產生工作階段的許可會是角色的以身分為基礎的政策和工作階段政策的交集。或者，如果您的管理員或自訂程式為您提供臨時憑證，他們可能已包含限制您存取的工作階段政策。

- 如果您是聯合身分使用者，您的工作階段可能受工作階段政策限制。您可以透過以 IAM 使用者身分登入，然後要求聯合權杖，成 AWS 為聯合身分使用者。如需有關聯合身分使用者的詳細資訊，請參閱 GetFederationToken—透過自訂身分經紀人聯合。如果您或身分經紀人在請求聯合權杖時傳遞工作階段政策，則您的工作階段會受這些政策限制。所產生工作階段的許可會是 IAM 使用者的以身分為基礎的政策和工作階段政策的交集。如需有關工作階段政策的詳細資訊，請參閱 工作階段政策。
- 如果您使用角色存取具有以資源為基礎之政策的資源，則請確認政策已授與該角色許可。例如，以下政策允許 MyRole 從帳戶 111122223333 存取 MyBucket。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "S3BucketPolicy",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},
    "Action": ["s3:PutObject"],
    "Resource": ["arn:aws:s3:::MyBucket/*"]
  }]
}
```

拒絕存取錯誤訊息範例

大多數拒絕存取錯誤訊息的格式為 User *user* is not authorized to perform *action* on *resource* because *context*。在此範例中，*###*是未接收存取的 Amazon Resource Name (ARN)，*##*是政策拒絕的服務動作，而*##*是政策對其執行動作的資源的 ARN。*##*欄位代表政策類型的其他內容，其中說明政策拒絕存取的原因。

當原則因為原則 AWS 包含Deny陳述式而明確拒絕存取時，請在拒絕存取錯誤訊息with an explicit deny in a *type* policy中加入該片語。當原則隱含拒絕存取時，請在「拒絕存取」錯誤訊息because no *type* policy allows the *action* action中加入該字句。 AWS

> **Note**
>
> 某些 AWS 服務不支援此存取遭拒的錯誤訊息格式。存取遭拒錯誤訊息的內容可能會因提出授權請求的服務而有所不同。

以下範例顯示不同類型的拒絕存取錯誤訊息的格式。

因服務控制政策而拒絕存取 – 隱含拒絕

1. 在服務控制政策 (SCP) 中檢查該動作是否有缺少的 `Allow` 陳述式。對於下列範例，動作是 `codecommit:ListRepositories`。
2. 透過新增 `Allow` 陳述式來更新您的 SCP。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的更新 SCP。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no service control policy allows the
 codecommit:ListRespositories action
```

因服務控制政策而拒絕存取 – 明確拒絕

1. 在服務控制政策 (SCP) 中檢查該動作是否有 `Deny` 陳述式。對於下列範例，動作是 `codecommit:ListRepositories`。
2. 透過移除 `Deny` 陳述式來更新您的 SCP。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的更新 SCP。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories with an explicit deny in a service control policy
```

因 VPC 端點政策而拒絕存取 – 隱含拒絕

1. 在您的虛擬私有雲端 (VPC) 端點政策中檢查該動作是否有遺失的 `Allow` 陳述式。對於下列範例，動作是 `codecommit:ListRepositories`。
2. 透過新增 `Allow` 陳述式來更新 VPC 端點政策。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的更新 VPC 端點政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no VPC endpoint policy allows the
 codecommit:ListRepositories action
```

因 VPC 端點政策而拒絕存取 – 明確拒絕

1. 在您的虛擬私有雲端 (VPC) 端點政策中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是 codedeploy:ListDeployments。
2. 透過移除 Deny 陳述式來更新 VPC 端點政策。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的更新 VPC 端點政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a VPC endpoint policy
```

因許可界限而拒絕存取 – 隱含拒絕

1. 在許可界限中檢查該動作是否有遺失的 Allow 陳述式。對於下列範例，動作是 codedeploy:ListDeployments。
2. 透過將 Allow 陳述式新增至您的 IAM 政策來更新您的許可界限。如需詳細資訊，請參閱 IAM 實體的許可界限 及 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* because no permissions boundary allows the
 codedeploy:ListDeployments action
```

因許可界限而拒絕存取 – 明確拒絕

1. 在許可界限中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是 sagemaker:ListModels。
2. 透過從您的 IAM 政策中移除 Deny 陳述式來更新您的許可界限。如需詳細資訊，請參閱 IAM 實體的許可界限 及 編輯 IAM 政策。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sagemaker:ListModels with an explicit deny in a permissions boundary
```

因工作階段政策而拒絕存取 – 隱含拒絕

1. 在工作階段政策中檢查該動作是否有遺失的 Allow 陳述式。對於下列範例，動作是 codecommit:ListRepositories。
2. 透過新增 Allow 陳述式來更新您的工作階段政策。如需詳細資訊，請參閱工作階段政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no session policy allows the
 codecommit:ListRepositories action
```

因工作階段政策而拒絕存取 – 明確拒絕

1. 在工作階段政策中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是 codedeploy:ListDeployments。
2. 透過移除 Deny 陳述式來更新您的工作階段政策。如需詳細資訊，請參閱工作階段政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in a sessions policy
```

因以資源為基礎的政策而拒絕存取 – 隱含拒絕

1. 在以資源為基礎的政策中檢查該動作是否有遺失的 Allow 陳述式。對於下列範例，動作是 secretsmanager:GetSecretValue。
2. 透過新增 Allow 陳述式來更新您的政策。如需詳細資訊，請參閱以資源為基礎的政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue because no resource-based policy allows the
 secretsmanager:GetSecretValue action
```

因以資源為基礎的政策而拒絕存取 – 明確拒絕

1. 在以資源為基礎的政策中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是 `secretsmanager:GetSecretValue`。
2. 透過移除 Deny 陳述式來更新您的政策。如需詳細資訊，請參閱以資源為基礎的政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
secretsmanager:GetSecretValue on resource: arn:aws:secretsmanager:us-
east-1:123456789012:secret:* with an explicit deny in a resource-based policy
```

因角色信任政策而拒絕存取 – 隱含拒絕

1. 在角色信任政策中檢查該動作是否有遺失的 Allow 陳述式。對於下列範例，動作是 `sts:AssumeRole`。
2. 透過新增 Allow 陳述式來更新您的政策。如需詳細資訊，請參閱以資源為基礎的政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
sts:AssumeRole because no role trust policy allows the sts:AssumeRole action
```

因角色信任政策而拒絕存取 – 明確拒絕

1. 在角色信任政策中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是 `sts:AssumeRole`。
2. 透過移除 Deny 陳述式來更新您的政策。如需詳細資訊，請參閱以資源為基礎的政策和 編輯 IAM 政策。

```
User: arn:aws:iam::777788889999:user/JohnDoe is not authorized to perform:
sts:AssumeRole with an explicit deny in the role trust policy
```

因以身分為基礎的政策而拒絕存取 – 隱含拒絕

1. 在連接至身分的以身分為基礎的政策中，檢查該動作是否有遺失的 Allow 陳述式。對於下列範例，動作是連接至使用者 JohnDoe 的 `codecommit:ListRepositories`。

2. 透過新增 Allow 陳述式來更新您的政策。如需詳細資訊，請參閱以身分為基礎的政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codecommit:ListRepositories because no identity-based policy allows the
 codecommit:ListRepositories action
```

因以身分為基礎的政策而拒絕存取 – 明確拒絕

1. 在連接至身分的以身分為基礎的政策中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是連接至使用者 JohnDoe 的 codedeploy:ListDeployments。
2. 透過移除 Deny 陳述式來更新您的政策。如需詳細資訊，請參閱以身分為基礎的政策和 編輯 IAM 政策。

```
User: arn:aws:iam::123456789012:user/JohnDoe is not authorized to perform:
codedeploy:ListDeployments on resource: arn:aws:codedeploy:us-
east-1:123456789012:deploymentgroup:* with an explicit deny in an identity-based policy
```

當 VPC 請求因其他政策而失敗時拒絕存取

1. 在服務控制政策 (SCP) 中檢查該動作是否有明確的 Deny 陳述式。對於下列範例，動作是 SNS:Publish。
2. 透過移除 Deny 陳述式來更新您的 SCP。如需詳細資訊，請參閱《AWS IAM Identity Center 使用者指南》中的更新 SCP。

```
User: arn:aws:sts::111122223333:assumed-role/role-name/role-session-name is not
 authorized to perform:
SNS:Publish on resource: arn:aws:sns:us-east-1:444455556666:role-name-2
with an explicit deny in a VPC endpoint policy transitively through a service control
 policy
```

故障排除 IAM 政策

原則是一個實體 AWS ，當附加至身分識別或資源時，會定義其權限。 AWS 當主參與者 (例如使用者) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。原則會以 JSON 文件的形

式儲 AWS 存在中，這些文件會以身分識別為基礎的原則附加至主體，或以資源為基礎的原則附加至資源。您可以連接以身分為基礎的政策到主體 (或身分)，例如 IAM 群組、使用者或角色。以身分為基礎的政策包括 AWS 受管政策、客戶受管政策以及內嵌政策。您可以在使用 Visual 和 JSON 編輯器選項中 AWS Management Console 建立和編輯客戶管理的政策。當您檢視中的原則時 AWS Management Console，您可以看到該原則授與的權限摘要。您可以使用視覺化編輯器和政策摘要幫助診斷並修正在管理 IAM 政策時遇到的常見錯誤。

請記住，所有 IAM 政策都是使用以JavaScript 物件標記法 (JSON) 規則開頭的語法來儲存。您不需要了解該語法即可建立或管理您政策。您可以在 AWS Management Console中使用視覺化編輯器來建立和編輯政策。若要進一步了解 IAM 中 JSON 語法的詳細資訊，請參閱 IAM JSON 政策語言的文法 。

針對 IAM 政策主題進行故障診斷

- 使用視覺化編輯器進行故障排除
  - 政策結構調整
  - 在視覺化編輯器中選擇資源 ARN
  - 在視覺化編輯器中拒絕許可
  - 在視覺化編輯器中指定多個服務
  - 在視覺化編輯器中縮減政策大小
  - 在視覺化編輯器中修正無法識別的服務、動作或資源類型
- 使用政策摘要進行故障排除
  - 缺少政策摘要
  - 政策摘要包含無法識別的服務、動作或資源類型
  - 服務不支援 IAM 政策摘要
  - 我的政策未授與預期的許可
- 故障排除政策管理
  - 在 IAM 帳戶中連接或分開政策
  - 根據 IAM 身分的活動來變更其政策
- JSON 政策文件故障排除
  - 驗證您的政策
  - 我在 JSON 編輯器中沒有政策驗證的許可
  - 多個 JSON 政策物件
  - 多個 JSON 陳述式元素
  - 在 JSON 陳述式元素中具有多個效果、動作或資源元素

- 缺少 JSON 版本元素

使用視覺化編輯器進行故障排除

在建立或編輯客戶管理政策時，您可以使用視覺化編輯器中的資訊，來協助您針對政策中的錯誤進行故障排除。若要檢視使用視覺化編輯器建立政策的範例，請參閱 the section called “控制對身分的存取”。

政策結構調整

建立原則時，請先 AWS 驗證、處理和轉換原則，然後再儲存原則。當 AWS 傳回原則以回應使用者查詢或將其顯示在主控台時，會將原則 AWS 轉換回人類可讀的格式，而不會變更原則授與的權限。這可能會導致與政策視覺化編輯器或 JSON 標籤中顯示的內容有所不同：視覺化編輯器許可區塊可以新增、移除或重新排序，並且區塊中的內容可以進行最佳化。在 JSON 標籤中，可以移除較不重要的空格，並且可以將 JSON 映射中的元素重新排序。此外，主元素內的 AWS 帳戶 ID 也可以由的 ARN 取 AWS 帳戶根使用者代。由於可能發生這些更改，不應以字串形式來比較 JSON 政策文件。

當您在中建立客戶管理政策時 AWS Management Console，您可以選擇完全在 JSON 編輯器中運作。如果從未在視覺化編輯器中進行任何變更並從 JSON 編輯器中選擇下一步，則不太可能會調整政策結構。不過，如果您建立一個政策並使用視覺化編輯器進行任何修改，或者從視覺化編輯器選項中選擇下一步，IAM 就可能會調整政策結構，以便在視覺化編輯器中最佳化其外觀。

此結構調整僅在您的編輯工作階段中進行，不會自動儲存。

如果您的政策在編輯工作階段中進行結構調整，則 IAM 會根據以下情況確定是否儲存結構調整：

使用此編輯器選項	若您編輯政策	然後從該索引標籤選擇下一步	當您選擇 Save changes (儲存變更) 時
視覺化	已編輯	視覺化	對政策進行結構調整
視覺化	已編輯	JSON	對政策進行結構調整
視覺化	未編輯	視覺化	對政策進行結構調整
JSON	已編輯	視覺化	對政策進行結構調整
JSON	已編輯	JSON	此政策結構未變更

使用此編輯器選項	若您編輯政策	然後從該索引標籤選擇下一步	當您選擇 Save changes (儲存變更) 時
JSON	未編輯	JSON	此政策結構未變更

IAM 可能會對複雜政策或具有允許多個服務、資源類型或條件金鑰的許可區塊或陳述式之政策進行結構調整。

在視覺化編輯器中選擇資源 ARN

在使用視覺化編輯器建立或編輯政策時，您必須先選擇一個服務，然後從該服務中選擇動作。如果選取的服務和動作支援選擇的特定資源，視覺化編輯器將列出支援的資源類型。然後您可以選擇 Add ARN (新增 ARN) 來提供關於資源的詳細資訊。您可以從以下選項中進行選擇以新增資源類型的 ARN。

- 使用 ARN 產生器：根據資源類型，您可能會看到用於產生 ARN 的不同欄位。您也可以選擇 Any (任意) 來為指定設定的任何值提供許可。例如，若您選取了 Amazon EC2 Read (讀取) 存取等級群組，那麼您政策中的動作便支援 `instance` 資源類型。您必須提供資源的「區域」、「帳戶」和InstanceId值。如果提供您的帳戶 ID，但為區域和執行個體 ID 選擇 Any (任意)，則政策會為您帳戶中的任何執行個體授與許可。
- 輸入或貼上 ARN：您可以根據資源的 Amazon Resource Name (ARN) 來指定資源。您也可以在 ARN 的任何欄位中包含萬用字元 (*) (在每組冒號之間)。如需詳細資訊，請參閱 IAM JSON 政策元素：Resource。

在視覺化編輯器中拒絕許可

預設情況下，您使用視覺化編輯器建立的政策允許執行您選擇的動作。若要拒絕選擇的動作，請選擇 Switch to deny permissions (切換為拒絕許可)。由於請求係預設被拒絕，做為安全最佳實務，我們建議您僅允許使用者所需動作和資源的許可。只有在要覆蓋其他語句或政策單獨允許的許可時，才應建立陳述式來拒絕許可。我們建議您將拒絕許可數限制為最低，因為它們可能會增加解決許可問題的難度。如需 IAM 評估政策邏輯之方法的詳細資訊，請參閱 政策評估邏輯。

> **Note**
>
> 默認情況下，只 AWS 帳戶根使用者 有該帳戶中的所有資源可以訪問。因此，如果未以根使用者登入，您必須具有政策授與的許可。

在視覺化編輯器中指定多個服務

在使用視覺化編輯器建構政策時，您每次只能選擇一個服務。這是一項最佳實務，因為視覺化編輯器允許您從該服務的動作中進行選擇。然後，您可以從該服務和選定的動作支援的資源中進行選擇。如此一來就可以更輕鬆地建立政策和進行故障排除。

如果您熟悉 JSON 語法，還可以使用萬用字元 (*) 手動指定多個服務。例如，輸入 **Code***，以便為以 Code 開頭的所有服務 (如 CodeBuild 與 CodeCommit) 提供許可。不過，您必須接著輸入動作和資源 ARN 以完成您的政策。此外，在儲存您的政策時，可能會進行結構調整，以在單獨的許可區塊中包含每個服務。

或者，若要在服務中使用 JSON 語法 (如萬用字元)，請使用 JSON 編輯器選項來建立、編輯和儲存政策。

在視覺化編輯器中縮減政策大小

在使用視覺化編輯器建立政策時，IAM 將建立一個 JSON 文件來存放您的政策。您可以切換到 JSON 編輯器選項來檢視該文件。如果該 JSON 文件超過政策的大小限制，視覺化編輯器將顯示一條錯誤訊息，並禁止檢閱並儲存您的政策。若要查看 IAM 對於受管政策的大小限制，請參閱 IAM 和 STS 字元限制。

若要在視覺化編輯器中縮減您的政策大小，請編輯您的政策或將許可區塊移到另一個政策。錯誤訊息包括政策文件中包含的字元數，您可以使用該資訊說明來縮減您的政策大小。

在視覺化編輯器中修正無法識別的服務、動作或資源類型

在視覺化編輯器中建立或編輯政策時，您可能會看到一條警告，指出您的政策包含無法識別的服務、動作或資源類型。

> **Note**
>
> IAM 對支援政策摘要的服務檢閱服務名稱、動作和資源類型。不過，您的政策摘要可能包含不存在資源值或條件。請一律使用政策模擬器來測試政策。

如果您的政策包含無法識別的服務、動作或資源類型，則存在以下錯誤之一：

- 預覽服務：處於預覽狀態的服務不支援視覺化編輯器。如果您參與預覽，可以忽略該警告並繼續，但必須手動輸入動作和資源 ARN 以完成您的政策。或者，您可以選擇 JSON 編輯器選項，來輸入或貼上 JSON 政策文件。

- 自訂服務：自訂服務不支援視覺化編輯器。如果您使用自訂服務，可以忽略該警告並繼續，但必須手動輸入動作和資源 ARN 以完成您的政策。或者，您可以選擇 JSON 編輯器選項，來輸入或貼上 JSON 政策文件。
- 服務不支援視覺化編輯器：如果您的政策包含不支援視覺化編輯器的公開提供 (GA) 服務，您可以忽略該警告並繼續，但必須手動輸入動作和資源 ARN 以完成您的政策。或者，您可以選擇 JSON 編輯器選項，來輸入或貼上 JSON 政策文件。

  公開提供服務是公開發佈的服務，不是預覽或自訂服務。如果無法識別的服務是公開提供的，並且名稱拼寫正確，則該服務不支援視覺化編輯器。若要了解如何請求 GA 服務的視覺化編輯器或政策摘要支援，請參閱 服務不支援 IAM 政策摘要。
- 動作不支援視覺化編輯器：如果您政策包含的受支援服務具有不支援的動作，您可以忽略該警告並繼續，但必須手動輸入動作和資源 ARN 以完成您的政策。或者，您可以選擇 JSON 編輯器選項，來輸入或貼上 JSON 政策文件。

  如果您的政策包含的受支援服務具有不支援的動作，則該服務不完全支援視覺化編輯器。若要了解如何請求 GA 服務的視覺化編輯器或政策摘要支援，請參閱 服務不支援 IAM 政策摘要。
- 資源類型不支援視覺化編輯器：如果您的政策包含的受支援動作具有不支援的資源類型，您可以忽略該警告並繼續。不過，IAM 無法確認是否包含所有選定動作的資源，且您可能會看到額外的警告。
- 拼寫錯誤：在視覺化編輯器中手動輸入服務、動作或資源時，您建立的政策可能會包含拼寫錯誤。為了避免發生此情況，建議您使用視覺化編輯器從服務和動作清單中進行選擇，然後根據提示完成資源部分。不過，如果服務不完全支援視覺化編輯器，您可能需要手動輸入某些政策部分。

  如果您確定自己的政策不包含上述任何錯誤，那麼您的政策可能包含拼寫錯誤。檢查服務、動作和資源類型名稱有無拼寫錯誤。例如，您可能誤用了 `s2` 而不是 `s3`，或誤用了 `ListMyBuckets` 而非 `ListAllMyBuckets`。另一個常見的動作拼寫錯誤是 ARN 中包含不必要的文字 (如 `arn:aws:s3: : :*`) 或動作中缺少冒號 (如 `iam.CreateUser`)。您可以透過選擇下一步，來檢視政策摘要，並確認政策是否提供所需的許可，從而對可能包含拼寫錯誤的政策進行評估。

使用政策摘要進行故障排除

您可以診斷並解決與政策摘要相關的問題。

缺少政策摘要

IAM 主控台中提供了政策摘要表，這些表總結政策中對每個服務允許或拒絕的存取級別、資源和條件。政策摘要列於三個表中：政策摘要、服務摘要以及動作摘要。政策摘要表包括由所選政策定義的服務清單和許可摘要。您可以在該政策的政策詳細資訊頁面上，檢視連接至實體之任何政策的政策摘要。

您可以在 Policies (政策) 頁面上檢視受管政策的政策摘要。如果 AWS 無法呈現原則的摘要，則您會看到 JSON 政策文件而非摘要，並收到下列錯誤：

A summary for this policy cannot be generated. (無法產生此政策的摘要。) You can still view or edit the JSON policy document. (您仍可以檢視或編輯 JSON 政策文件。)

如果您的政策不包含摘要，則已發生下列錯誤之一：

- 不支援的政策元素：IAM 不支援為包含以下其中一項政策元素的政策產生政策摘要：
  - `Principal`
  - `NotPrincipal`
  - `NotResource`
- 無政策許可：如果政策不提供任何有效許可，則無法產生政策摘要。例如，如果政策包含元素 `"NotAction": "*"` 的單一陳述式，則它將授與對於除「所有動作」(*) 之外的所有動作的存取權。這代表它未授與對任何內容的 `Deny` 或 `Allow` 的存取權。

> **Note**
>
> 請謹慎使用這些政策元素 (如 `NotPrincipal`、`NotAction`、`NotResource`)。如需使用政策元素的資訊，請參閱 IAM JSON 政策元素參考。

如果您提供不相符的服務和資源，則可能會產生一個不提供有效許可的政策。如果您指定一個服務中的動作和另一個服務中的資源，則可能會發生這種情況。在這種情況下，仍然會出現政策摘要。唯一顯示出有問題的跡象是摘要中的資源列可能包含來自不同服務的資源。如果此欄位包含不相符的資源，則您應該檢閱您的政策錯誤。若要更全面了解您的政策，請一律以政策模擬器來測試。

政策摘要包含無法識別的服務、動作或資源類型

在 IAM 主控台中，若政策摘要包含警告符號 (⚠)，則政策可能包含無法識別的服務、動作或資源類型。若要了解政策摘要中的警示，請參閱 政策摘要 (服務清單)。

> **Note**
>
> IAM 會對支援政策摘要的服務檢閱服務名稱、動作和資源類型。不過，您的政策摘要可能包含不存在資源值或條件。請一律使用政策模擬器來測試政策。

如果您的政策包含無法識別的服務、動作或資源類型，則存在以下錯誤之一：

- 預覽服務：處於預覽狀態的服務不支援政策摘要。
- 自訂服務：自訂服務不支援政策摘要。
- 服務不支援摘要：如果您的政策包括不支援政策摘要的公開提供 (GA) 服務，則該服務將包含在政策摘要表的 Unrecognized services (無法識別的服務) 部分。公開提供服務是公開發佈的服務，不是預覽或自訂服務。如果無法識別的服務是公開提供的，並且名稱拼寫正確，則該服務不支援 IAM 政策摘要。若要了解如何請求 GA 服務的政策摘要支援，請參閱 服務不支援 IAM 政策摘要。
- 動作不支援摘要：如果您的政策包含的受支援服務具有不受支援的動作，則該動作將包含在服務摘要表的 Unrecognized actions (無法識別的動作) 部分。若要了解服務摘要中的警示，請參閱 服務摘要 (動作清單)。
- 資源類型不支援摘要：如果您政策包含的受支援動作具有不受支援的資源類型，則該資源將包含在服務摘要表的 Unrecognized resource types (無法識別的資源類型) 部分。若要了解服務摘要中的警示，請參閱 服務摘要 (動作清單)。
- 錯字 — AWS 檢查 JSON 在語法上是否正確，而且原則不包含錯別字或其他錯誤做為原則驗證的一部分。

> **Note**
>
> 作為最佳實務，我們會建議使用您 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作。我們建議您開啟現有政策並檢閱並解決任何政策驗證建議。

服務不支援 IAM 政策摘要

如果 IAM 政策摘要或視覺化編輯器無法辨識公開提供 (GA) 服務或動作，則該服務可能不支援這些功能。公開提供服務是公開發佈的服務而非預覽或自訂服務。如果無法識別的服務是公開提供的，並且名稱拼寫正確，則該服務不支援這些功能。如果您的政策包含的受支援服務具有不支援的動作，則該服務不完全支援 IAM 政策摘要。

請求服務新增 IAM 政策摘要或視覺化編輯器支援

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 找到包含不受支援服務的政策：
   - 若政策為受管政策，請在導覽窗格中選擇 Policies (政策)。在政策清單中，選擇您要檢視的政策名稱。
   - 如果該政策是連接到使用者的內嵌政策，請在導覽窗格中選擇 Users (使用者)。在使用者清單中，選擇要檢視其政策的使用者的名稱。在使用者的政策表中，展開您要查看政策摘要的標題。
3. 在 AWS Management Console 頁尾左側，選擇 [意見反應]。在 IAM 的意見反應方塊中，輸入 **I request that the <ServiceName> service add support for IAM policy summaries and the visual editor**。若您希望一個以上的服務支援摘要，請輸入 **I request that the <ServiceName1>, <ServiceName2>, and <ServiceName3> services add support for IAM policy summaries and the visual editor**。

請求服務新增對缺漏動作的 IAM 政策摘要支援

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 找到包含不受支援服務的政策：
   - 若政策為受管政策，請在導覽窗格中選擇 Policies (政策)。在政策清單中，選擇您要檢視的政策名稱。
   - 如果該政策是連接到使用者的內嵌政策，請在導覽窗格中選擇 Users (使用者)。在使用者清單中，選擇要檢視其政策的使用者的名稱。在使用者的政策表中，選擇您要檢視的政策名稱來展開政策摘要。
3. 在政策摘要中，選擇包含不支援動作的服務的名稱。
4. 在 AWS Management Console 頁尾左側，選擇 [意見反應]。在 IAM 的意見反應方塊中，輸入 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName> action**。若您希望報告超過一個未受支援的動作，請輸入 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName1>, <ActionName2>, and <ActionName3> actions**。

若要請求包含缺少之動作的不同服務，請重複最後三個步驟。

我的政策未授與預期的許可

若要指派許可給使用者、群組、角色或資源，您必須建立一個政策，其為一個定義許可的文件。政策文件包含下列元素：

- 效果：政策是否允許或拒絕存取
- 動作：政策允許或拒絕的動作清單
- 資源：動作可在其中發生的資源清單
- 條件：(選擇性) 政策授與許可的條件

若要了解這些內容與其他政策元素，請參閱 IAM JSON 政策元素參考。

若要授與存取，您的政策必須定義含有支援的資源之動作。如果您的政策也包含條件，則該條件必須包含一個全域條件鍵或者必須套用到動作。若要了解受動作支援的資源，請參閱您服務的 AWS 文件。若要瞭解動作支援哪些條件，請參閱AWS 服務的動作、資源和條件索引鍵。

若要了解您的政策是否定義不授與許可的動作、資源或條件，您可以使用位於https://console.aws.amazon.com/iam/ 的 IAM 主控台檢閱政策的政策摘要。您可以使用政策摘要來辨識並修正政策中的問題。

對於元素不依照 IAM 政策中的定義授與許可的情況，原因可能有：

- 定義了動作，但是沒有適用的資源
- 定義了資源，但是沒有適用的動作
- 定義了條件，但是沒有適用的動作

若要檢視包含警告的政策摘要的範例，請參閱 the section called “政策摘要 (服務清單)”。

已定義動作，但是沒有適用的資源

下方的政策定義了所有 `ec2:Describe*` 動作和一個特定資源。這些動作都不支援資源級許可，因而未授與任何 `ec2:Describe` 動作。資源級許可表示動作使用政策的 `Resource` 元素中的 ARN 來支援資源。如果動作不支援資源層級許可，則政策中的陳述式必須在 `Resource` 元素中使用萬用字元 (*)。若要了解哪些服務支援資源層級的許可，請參閱 AWS 與 IAM 搭配使用的服務。

```
{
    "Version": "2012-10-17",
    "Statement": [{
```

```
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "arn:aws:ec2:us-east-2:ACCOUNT-ID:instance/*"
    }]
}
```

此政策不提供任何許可，而政策摘要包含下列錯誤：

`This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.`

若要修正此政策，您必須使用 `Resource` 元素中的 `*`。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
    }]
}
```

已定義資源，但是沒有適用的動作

下面的政策定義了 Amazon S3 儲存貯體資源，但不包含可在該資源上執行的 S3 動作。此政策還授予對所有 Amazon CloudFront 操作的完全訪問權限。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "cloudfront:*",
        "Resource": [
            "arn:aws:cloudfront:*",
            "arn:aws:s3:::examplebucket"
        ]
    }]
}
```

此原則提供所有 CloudFront 動作的權限。但是，因為政策定義 S3 `examplebucket` 資源卻未定義任何 S3 動作，政策摘要包含下列警告：

```
This policy defines some actions, resources, or conditions that do not
provide permissions. To grant access, policies must have an action that has
an applicable resource or condition.
```

若要修正此政策以提供 S3 儲存貯體許可，您必須定義可在儲存貯體資源上執行的 S3 動作。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "cloudfront:*",
            "s3:CreateBucket",
            "s3:ListBucket*",
            "s3:PutBucket*",
            "s3:GetBucket*"
        ],
        "Resource": [
            "arn:aws:cloudfront:*",
            "arn:aws:s3:::examplebucket"
        ]
    }]
}
```

或者，若要修正此政策以僅提供 CloudFront 許可，請移除 S3 資源。

已定義條件，但是沒有適用的動作

如果 S3 字首等同於 `custom` 而版本 ID 等同於 `1234`，下列政策會為所有 S3 資源定義兩項 Amazon S3 動作。但是，`s3:VersionId` 條件金鑰用於物件版本標記，不受所定義的儲存貯體動作的支援。若要瞭解動作支援哪些條件，請參閱AWS 服務的動作、資源和條件索引鍵，並遵循條件索引鍵的服務文件連結。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
                }
            }
        }
    ]
}
```

如果儲存貯體名稱包含 s3:ListBucketVersions 字首，此政策為 s3:ListBucket 動作和 custom 動作提供許可。但是，由於 s3:VersionId 條件不受任何定義的動作支援，政策摘要包含下列錯誤：

```
This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.
```

若要修正此政策來使用 S3 物件版本標記，您必須定義支援 s3:VersionId 條件金鑰的 S3 動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
```

```
                }
            }
        }
    ]
}
```

此政策為政策中的每個動作和條件提供許可。但由於不存在單一動作符合兩個條件的情況，政策仍不提供任何許可。為解決這一問題，您必須建立兩個單獨的陳述式，每個陳述式只包含具有套用條件的動作。

若要修正此政策，請建立兩個陳述式。第一個陳述式包含支援 s3:prefix 條件的動作，第二個陳述式包含支援 s3:VersionId 條件的動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": "custom"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:VersionId": "1234"
                }
            }
        }
    ]
}
```

故障排除政策管理

您可以診斷並解決與政策管理相關的問題。

在 IAM 帳戶中連接或分開政策

某些 AWS 受管理的策略會連結至服務。這些政策僅用於該服務的某個服務連結角色。在 IAM 主控台中，當您查看政策的政策詳細資訊頁面時，該頁面會包含一個橫幅，顯示該政策已與服務建立關聯。您不可將此政策連接到 IAM 中的使用者、群組或角色。當您為服務建立與服務相關的角色時，該政策會自動連接到您的新角色。由於政策是必要的您無法將政策與服務相關角色分開。

根據 IAM 身分的活動來變更其政策

您可以為您的 IAM 身分 (使用者、群組和角色) 根據其活動更新政策。要做到這一點，請在 CloudTrail 事件歷史記錄中查看您帳戶的事件。 CloudTrail 事件記錄檔包含詳細的事件資訊，您可以用來變更策略的權限。您可能會發現使用者或角色正在嘗試執行中的動作， AWS 且該要求遭到拒絕。在這種情況下，您可以考量使用者或角色是否應該有執行該動作的許可。若是如此，您可以在其政策中新增動作，甚至其所嘗試存取資源的 ARN。或者，如果使用者或角色具備其未使用的許可，則您可以考慮從其政策中移除這類許可。請確定您的政策僅授與執行必要動作所需的最低權限。如需有關使用的詳細資訊 CloudTrail，請參閱《使用指南》中的〈在 CloudTrail主控台中檢視 CloudTrail 事件AWS CloudTrail〉。

JSON 政策文件故障排除

您可以診斷並解決與 JSON 政策文件相關的問題。

驗證您的政策

當您建立或編輯 JSON 政策時，IAM 可以執行政策驗證以協助您建立有效的政策。IAM 會識別 JSON 語法錯誤，而 IAM Access Analyzer 會提供額外的政策檢查及建議，協助您進一步改良政策。若要進一步了解政策驗證的資訊，請參閱 驗證 IAM 政策。若要進一步了解 IAM Access Analyzer 政策檢查和可動作的建議，請參閱 IAM Access Analyzer 政策驗證。

我在 JSON 編輯器中沒有政策驗證的許可

在中 AWS Management Console，如果您沒有檢視 IAM Access Analyzer 政策驗證結果的權限，您可能會收到下列錯誤：

```
You need permissions. You do not have the permissions required to perform
this operation. Ask your administrator to add permissions.
```

若要修正此錯誤，請要求管理員為您新增 access-analyzer:ValidatePolicy 許可。

多個 JSON 政策物件

一個 IAM 政策必須包含一個且只能包含一個 JSON 物件。可在兩旁放置 { } 括弧來表示物件。雖然您可以在外側括弧對中嵌入額外的 { } 括弧以在 JSON 物件中巢套其他物件，但是一個政策只能包含一個最外層的 { } 括弧對。以下範例不正確，因為它在最上層包含兩個物件 (以##標示)：

```
{
      "Version": "2012-10-17",
      "Statement":
      {
         "Effect":"Allow",
         "Action":"ec2:Describe*",
         "Resource":"*"
      }
   }
   {
     "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket/*"
     }
   }
```

不過，您可以使用正確的政策語法來達成上面範例的意圖。可以將兩個區塊合併到單個 Statement 元素中，而非包含兩個各自擁有 Statement 元素的完整政策物件。Statement 元素將有兩個物件的陣列做為其值，如以下範例所示 (以粗體標示)：

```
{
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":" *"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::my-bucket/*"
        }
```

```
      ]
    }
```

多個 JSON 陳述式元素

此錯誤乍看可能會像是前一章節中錯誤的變形。但是，它在語法上是不同類型的錯誤。在以下範例中，頂層只有一個政策物件，由單一 { } 括弧對表示。但是，該物件包含兩個 Statement 元素。

一個 IAM 政策只能包含一個 Statement 元素，包含在冒號左側的名稱 (Statement) 和跟隨其後在冒號右側的值。Statement 元素的值必須是物件，以 { } 括弧表示，其中包含一個 Effect 元素、一個 Action 元素和一個 Resource 元素。以下範例不正確，因為它在政策物件中包含兩個 Statement 元素 (以##標示)：

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
      },
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket/*"
      }
    }
```

值物件可以是多個值物件組成的陣列。若要解決此問題，可使用物件陣列將兩個 Statement 元素合併為一個元素，如以下範例所示 (以粗體標示)：

```
{
      "Version": "2012-10-17",
      "Statement": [    {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":"*"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::my-bucket/*"
```

```
        }
     ]
    }
```

Statement 元素的值是一種物件陣列。此示例中的陣列包含兩個物件，每個物件本身是 Statement 元素的正確值。陣列中的每個物件之間用逗號隔開。

在 JSON 陳述式元素中具有多個效果、動作或資源元素

在 Statement 名稱/值對的值側，物件只能包含一個 Effect 元素、一個 Action 元素和一個 Resource 元素。以下政策不正確，因為在 Effect 的值物件中有兩個 Statement 元素：

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Deny",
        "Effect": "Allow",
        "Action": "ec2:* ",
        "Resource": "*"
      }
    }
```

> **Note**
>
> 政策引擎不允許新的或再編輯的政策中出現此類錯誤。但是，政策引擎會繼續允許在引擎更新之前儲存的政策。現有包含錯誤的政策之行為如下所示：
>
> - 多個 Effect 元素：僅遵循最後一個 Effect 元素。忽略其他元素。
> - 多個 Action 元素：所有 Action 元素進行內部合併，被視為單一列表。
> - 多個 Resource 元素：所有 Resource 元素進行內部合併，被視為單一列表。
>
> 政策引擎不允許您儲存任何有語法錯誤的政策。您必須先更正政策中的錯誤，然後才能儲存。我們建議您檢閱並修正所有政策驗證對您的政策做出的建議。

在每種情況下，解決方案都是刪除不正確的多餘元素。針對 Effect 元素，這十分簡單：若您希望先前的範例拒絕針對 Amazon EC2 執行個體的許可，您必須從政策移除 "Effect": "Allow", 一行，如下所示：

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Deny",
        "Action": "ec2:* ",
        "Resource": "*"
      }
    }
```

但是，如果重複元素是 Action 或 Resource，則解決方法可能會更加複雜。您可能需要提供多個動作允許 (或拒絕) 許可，或者需要控制對多個資源的存取。例如，以下範例不正確，因為它有多個 Resource 元素 (以##標示)：

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::my-bucket",
        "Resource": "arn:aws:s3:::my-bucket/*"
      }
    }
```

Statement 元素的值物件中的每個必要元素都只能出現一次。解決方案是將每個值置於陣列中。以下範例運用將兩個單獨資源元素合併為一個以陣列為值物件的 Resource 元素，來對此進行說明 (以粗體標示)：

```
{
      "Version": "2012-10-17",
      "Statement": {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": [
          "arn:aws:s3:::my-bucket",
          "arn:aws:s3:::my-bucket/*"
        ]
      }
    }
```

缺少 JSON 版本元素

Version 政策元素與政策版本不同。Version 政策元素是在政策內使用，並定義政策語言的版本。另一方面，政策版本會在您在 IAM 中變更客戶受管政策時建立。變更的政策不會覆寫現有的政策。IAM 反而會建立新版本的受管政策。若要進一步了解 Version 政策元素，請參閱 IAM JSON 政策元素：Version。若要進一步了解政策版本，請參閱 the section called “版本控制 IAM 政策”。

隨著 AWS 功能的發展，IAM 政策會新增新功能以支援這些功能。有時，政策語法更新包括新版本編號。如果您在政策中使用政策語法的較新功能，則必須告知政策分析引擎所使用的版本。預設政策版本是「2008-10-17」。如果要使用之後推出的任何政策功能，則必須指定支援所需功能的版本編號。我們建議您一律包含最新的政策語法版本編號 (目前為 "Version": "2012-10-17")。例如，以下政策不正確，因為它在資源 ARN 中使用政策變數 ${...}。但是，它沒有指定支援政策變數的政策語法版本 (以##標示)：

```
{
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

透過在政策頂層新增值 2012-10-17 (支援政策變數的第一個 IAM API 版本) 的 Version 元素，可解決此問題 (以粗體標示)：

```
{
  "Version": "2012-10-17",
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

對 FIDO 安全性金鑰進行故障診斷

使用此處的資訊，來協助您針對在使用 FIDO2 安全性金鑰時可能遇到的常見問題進行診斷。

主題

我無法啟用 FIDO 安全性金鑰

請根據您的狀態，以 IAM 使用者或系統管理員身分參閱下列解決方案

IAM 使用者

如果您無法啟用 FIDO 安全性金鑰，請檢查下列各項：

- 您使用的是支援的組態嗎？

  如需可與和搭配使用之裝置和瀏覽器的相關資訊 WebAuthn AWS，請參閱使用金鑰和安全金鑰的支援組態。

- 您正在使用 Mozilla Firefox 嗎？

  目前的 Firefox 版本預 WebAuthn 設支援。若要 WebAuthn在 Firefox 中啟用支援，請執行下列動作：

  1. 從 Firefox 地址列中，輸入 **about:config**。
  2. 在所開啟畫面的搜尋列中，輸入 **webauthn**。
  3. 選擇 security.webauth.webauthn，並將它的值變更為 true。

- 您正在使用任何瀏覽器外掛程式嗎？

  AWS 不支持使用插件添加 WebAuthn 瀏覽器支持。而是使用提供 WebAuthn 標準原生支援的瀏覽器。

  即使您使用的是受支援的瀏覽器，您也可能有與之不相容的外掛程式 WebAuthn。不相容的外掛程式可能會讓您無法啟用和使用 FIDO 相容安全金鑰。您應該停用任何可能不相容的外掛程式，並重新啟動瀏覽器。然後重試啟用 FIDO 安全性金鑰。

- 您具有適當的許可嗎？

  如果您沒有上述任何相容性問題，則可能沒有適當的許可。請聯絡系統管理員。

系統管理員

如果您是管理員，而 IAM 使用者在使用支援的組態下還是無法啟用其 FIDO 安全性金鑰，則請確保他們具有適當的許可。如需詳細範例，請參閱 IAM 教學課程：允許使用者管理其憑證和 MFA 設定。

我無法使用 FIDO 安全性金鑰登入

如果您是 IAM 使用者且無法 AWS Management Console 使用 FIDO 安全金鑰登入，請先參閱使用金鑰和安全金鑰的支援組態。如果您使用支援的組態，但無法登入，請聯絡系統管理員以尋求協助。

我的 FIDO 安全性金鑰遺失或損壞

以目前受支援 MFA 類型任意組合的最多八台 MFA 裝置可被指派給一個使用者。對於多台 MFA 裝置，您只需要有一台 MFA 裝置登入 AWS Management Console。取代 FIDO 安全金鑰與取代硬體 TOTP 權杖類似。如需遺失或損壞任何類型之 MFA 裝置要怎麼處理的資訊，請參閱 MFA 裝置遺失或停止運作時怎麼辦？。

其他問題

如果發生這裡未涵蓋的 FIDO 安全性金鑰問題，請採取下列行動之一：

- IAM 使用者：請聯絡系統管理員。
- AWS 帳戶 根使用者：聯絡 AWS Support。

IAM 角色故障診斷

使用此處的資訊，來協助您針對在使用 IAM 角色時所可能遇到的常見問題，進行診斷與排除。

主題

- 我無法擔任角色
- 顯示在我的 AWS 帳戶中的新角色
- 我無法編輯或刪除我的 AWS 帳戶中的角色
- 我沒有授權執行：iam：PassRole
- 為何我無法擔任具 12 小時工作階段的角色？ (AWS CLI, AWS API)
- 當我嘗試在 IAM 主控台中切換角色時收到錯誤

- 我的角色具有允許我執行動作的政策，但我收到「存取遭拒」
- 服務未建立角色的預設政策版本
- 主控台中沒有服務角色的使用案例

我無法擔任角色

請檢查以下內容：

- 若要允許使用者在角色工作階段中再次擔任目前的角色，請在角色信任原則中指定角色 AWS 帳戶 ARN 或 ARN 作為主參與者。 AWS 服務 提供 Amazon EC2、Amazon ECS、亞馬遜 EKS 和 Lambda 等運算資源提供臨時登入資料並自動更新這些登入資料。此可確保您始終擁有一組有效的憑證。對於這些服務，不需要再次擔任目前的角色即可取得臨時憑證。但是，如果您想要傳遞工作階段標籤或一個工作階段政策，則需要再次擔任目前的角色。若要瞭解如何修改角色信任原則以新增主參與者角色 ARN 或 AWS 帳戶 ARN，請參閱。修改角色信任政策 (主控台)
- 當您假設使用的角色時 AWS Management Console，請務必使用角色的確切名稱。在您擔任角色時，角色名稱會區分大小寫。
- 當您使用 AWS STS API 擔任角色時 AWS CLI，或者確保在 ARN 中使用角色的確切名稱。在您擔任角色時，角色名稱會區分大小寫。
- 確認您的 IAM 政策對於您要擔任的角色呼叫 `sts:AssumeRole` 授與了許可。您的 IAM 政策的 `Action` 元素必須允許您呼叫 `AssumeRole` 動作。此外，您的 IAM 政策的 `Resource` 元素必須指定您要擔任的角色。例如，`Resource` 元素可以透過其 Amazon Resource Name (ARN) 或萬用字元 (*) 來指定角色。例如，您必須至少有一個適用政策授與類似如下的許可：

```
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
```

- 請確認您的 IAM 身分已用 IAM 政策要求的任何標籤加上標籤。例如，在以下政策許可中，`Condition` 元素要求作為請求擔任該角色的主體，也就是您，必須有特定標籤。您必須以 `department = HR` 或 `department = CS` 加上標籤。否則，您無法擔任該角色。如需有關標記 IAM 使用者和角色的詳細資訊，請參閱 the section called “標記 IAM 資源”。

```
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "*",
    "Condition": {"StringEquals": {"aws:PrincipalTag/department": [
            "HR",
```

```
            "CS"
        ]}}
```

- 確定您符合角色的信任政策中指定的所有條件。Condition 可以指定過期日期、外部 ID、或請求必須只來自特定的 IP 地址。考慮下列範例，如果目前的日期為指定的日期之後的任何時間，則政策永遠不會相符，且無法授與您擔任該角色的許可。

```
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
    "Condition": {
        "DateLessThan" : {
            "aws:CurrentTime" : "2016-05-01T12:00:00Z"
        }
    }
```

- 確認您呼叫的 AWS 帳戶 來源AssumeRole是您假設之角色的受信任實體。受信任實體被定義為角色的信任政策中的 Principal。以下範例是連接到您要擔任之角色的信任政策。在此範例中，您登入所使用的帳戶 ID 和 IAM 使用者必須是 123456789012。如果您的帳戶號碼沒有列在角色的信任政策的 Principal 元素中，則您無法擔任該角色。無論存取政策中授與您什麼許可均是如此。請注意，範例政策會限制發生在 2017 年 7 月 1 日到 2017 年 12 月 31 日 (包含)(UTC) 間的動作許可。如果您在這些日期之前或之後登入，則政策不會相符，而且您無法擔任角色。

```
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
    }
```

- 來源身份 — 管理員可以將角色配置為要求身份傳遞自訂字串 AWS，該字串可識別正在中執行動作的人員或應用程式 (稱為來源身份識別)。確認所擔任的角色是否要求設定來源身分。如需來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

顯示在我的 AWS 帳戶中的新角色

某些 AWS 服務會要求您使用直接連結至服務的唯一服務角色類型。此服務連結角色是由服務預先定義的，並且包含服務所需要的所有許可。這會使設定服務更為簡單，因為您不必手動新增必要的許可。如需服務連結角色的一般資訊，請參閱 使用服務連結角色。

當服務開始支援服務連結的角色時，您可能已經在使用服務。若是如此，您可能會收到一封電子郵件，通知您關於您的帳戶中的新角色。此角色包含服務代表您執行動作所需的所有許可。您不需要執行任何動作來支援此角色。不過，您不應從您的帳戶刪除該角色。如此可能會移除服務存取 AWS 資源所需的許可。您可以前往 IAM 主控台的 IAM Roles (角色) 頁面，檢視您帳戶中的服務連結角色。服務連結角色會在表格的 Trusted entities (受信任實體) 欄中以 (Service-linked role) ((服務連結角色)) 顯示。

如需哪些服務支援服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找在 Service-Linked Role (服務連結角色) 一欄中顯示 Yes (是) 的服務。如需使用服務的服務連結角色相關資訊，請選擇 Yes (是) 連結。

我無法編輯或刪除我的 AWS 帳戶中的角色

您無法在 IAM 中刪除或編輯服務連結角色 的許可。這些角色包括服務所需的預先定義的信任和許可，以代表您執行動作。您可以使用 IAM 主控 AWS CLI台或 API 僅編輯服務連結角色的說明。您可以前往主控台的 IAM Roles (角色) 頁面，檢視您帳戶中的服務連結角色。服務連結角色會在表格的 Trusted entities (受信任實體) 欄中以 (Service-linked role) ((服務連結角色)) 顯示。在 Summary (摘要) 頁面上的橫幅，也會指出角色是一個服務連結角色。您可以僅透過連結的服務來管理和刪除這些角色，如果該服務支援動作。修改或刪除服務連結的角色時請務必謹慎，因為這樣可能移除服務存取 AWS 資源所需的許可。

如需哪些服務支援服務連結角色的資訊，請參閱 AWS 與 IAM 搭配使用的服務，並尋找在 Service-Linked Role (服務連結角色) 一欄中顯示 Yes (是) 的服務。

我沒有授權執行：iam：PassRole

當您建立服務連結的角色時，您必須擁有傳遞該角色到服務的許可。有些服務會自動在您在該服務中執行動作時，在您的帳戶中建立服務連結角色。例如，Amazon EC2 Auto Scaling 會在您第一次建立 Auto Scaling 群組時，為您建立 `AWSServiceRoleForAutoScaling` 服務連結角色。如果您嘗試建立 Auto Scaling 群組而無 `PassRole` 許可，您會收到以下錯誤：

```
ClientError: An error occurred (AccessDenied) when calling the
PutLifecycleHook operation: User: arn:aws:sts::111122223333:assumed-role/
Testrole/Diego is not authorized to perform: iam:PassRole on resource:
arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
```

若要修正此錯誤，請要求管理員為您新增 `iam:PassRole` 許可。

若要了解哪些服務支援服務連結角色，請參閱 AWS 與 IAM 搭配使用的服務。若要了解服務是否為您自動建立服務連結角色，請選擇 Yes (是) 連結以檢視該服務的服務連結角色文件。

為何我無法擔任具 12 小時工作階段的角色？ (AWS CLI, AWS API)

當您使用 AWS STS `AssumeRole*` API 或 `assume-role*` CLI 作業擔任角色時，您可以指定`DurationSeconds`參數的值。您可以指定從 900 秒 (15 分鐘) 到角色的 Maximum session duration (最大工作階段持續時間) 設定的值。如果您指定高於此設定的值，操作會失敗。此設定的上限值為 12 小時。例如，如果您指定 12 小時的工作階段持續時間，但是您的管理員設定最大工作階段持續時間為 6 小時，則您的操作會失敗。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。

如果您使用角色鏈結 (使用角色來擔任第二個角色)，則您的工作階段上限為一小時。如果您隨後使用 `DurationSeconds` 參數提供大於一小時的值，則操作會失敗。

當我嘗試在 IAM 主控台中切換角色時收到錯誤

您在 Switch Role (切換角色) 頁面上輸入的資訊必須符合角色的資訊。否則，操作會失敗，並且您會收到下列錯誤：

```
Invalid information in one or more fields. Check your information or
contact your administrator.
```

如果您收到此錯誤，請確認下列資訊正確：

- 帳號 ID 或別名 — AWS 帳戶 ID 為 12 位數字。您的帳戶可能有一個別名，這是一個易記的標識符，例如您的公司名稱，可以用來代替您的 AWS 帳戶 ID。您可以在此欄位中使用帳戶 ID 或別名。
- 角色名稱：角色名稱會區分大小寫。帳戶 ID 和角色名稱必須符合為角色設定的帳戶 ID 和角色名稱。

如果您持續收到錯誤訊息，請連絡您的管理員以驗證先前的資訊。角色信任政策或 IAM 使用者政策可能會限制您的存取。您的管理員可以驗證這些政策的許可。

我的角色具有允許我執行動作的政策，但我收到「存取遭拒」

您的角色工作階段可能受工作階段政策限制。當您使用程式設計方式要求臨時安全登入資料時 AWS STS，您可以選擇性地傳遞內嵌或受管理的工 工作階段政策是一種進階政策，且您會在以程式設計方式建立角色的暫時憑證工作階段時，以參數方式傳遞。您可以使用 `Policy` 參數傳遞單一 JSON 內嵌工作階段政策文件。您可以使用 `PolicyArns` 參數，指定多達 10 個受管工作階段政策。所產生工作階段的許可會是角色的以身分為基礎的政策和工作階段政策的交集。或者，如果您的管理員或自訂程式為您提供暫時憑證，他們可能已包含限制您存取的工作階段政策。

服務未建立角色的預設政策版本

服務角色是服務擔任的角色，以代表您在您的帳戶中執行動作。當您設定某些 AWS 服務環境時，您必須定義要擔任的服務角色。在某些情況下，服務會為您在 IAM 中建立服務角色及其政策。雖然您可以從 IAM 之內修改或刪除服務角色及其政策，但 AWS 不建議這樣做。該角色和政策本應僅供該服務使用。如果您編輯政策並設定另一個環境，當服務嘗試使用相同的角色和政策時，操作可能會失敗。

例如，當您第一次使 AWS CodeBuild 用時，服務會建立名為的角色`codebuild-RWBCore-service-role`。該服務角色使用名為 `codebuild-RWBCore-managed-policy` 的政策。如果您編輯政策，它會建立新版本，並將該版本儲存為預設版本。如果您在中執行後續作業 AWS CodeBuild，服務可能會嘗試更新原則。如果這樣做，您會收到下列錯誤：

```
codebuild.amazon.com did not create the default version (V2) of the
codebuild-RWBCore-managed-policy policy that is attached to the codebuild-
RWBCore-service-role role. To continue, detach the policy from any other
identities and then delete the policy and the role.
```

如果您收到這個錯誤，您必須在 IAM 中進行變更，才能繼續進行您的服務操作。首先，將預設政策版本設定為 V1，然後重試操作。如果先前已刪除 V1，或選擇 V1 無法運作，請清理並刪除現有的政策和角色。

如需編輯受管政策的詳細資訊，請參閱 編輯客戶受管政策 (主控台)。如需政策版本的詳細資訊，請參閱 版本控制 IAM 政策。

刪除服務角色及其政策

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格上選擇 Policies (政策)。
3. 在政策清單中，選擇您要刪除的政策名稱。
4. 選擇連接的實體索引標籤，可檢視哪些 IAM 使用者、群組或角色使用此政策。如果其中任一個身分使用政策，請完成下列任務：

    a. 建立具有必要許可的新受管政策。若要確保身分在您的動作之前和之後具有相同的許可，請從現有政策複製 JSON 政策文件。然後，建立新的受管政策並貼上 JSON 文件，如使用 JSON 編輯器建立政策所述。

    b. 對於每個受影響的身分，請連接新的政策，然後分開舊的政策。如需詳細資訊，請參閱 新增和移除 IAM 身分許可。

5. 在導覽窗格中，選擇 Roles (角色)。
6. 在角色清單中，選擇您要刪除的角色名稱。
7. 選擇 Trust relationships (信任關係) 標籤，以檢視哪些實體可以擔任角色。如果列出服務以外的任何實體，請完成下列任務：
   a. 建立新角色，這些角色信任那些實體。
   b. 您在上一個步驟中建立的政策。如果您略過了該步驟，請立即建立新的受管政策。
   c. 通知任何擔任該角色的人，他們不能再這樣做。為他們提供如何擔任新角色和具有相同許可的相關資訊。
8. 刪除策略。
9. 刪除角色。

主控台中沒有服務角色的使用案例

某些服務需要您手動建立服務角色，才能授與服務許可，以代表您執行動作。如果服務未列在 IAM 主控台中，您必須手動將服務列為受信任主體。如果您使用的服務或功能的文件不包含將服務列為受信任主體的說明，請提供頁面的意見回饋。

若要手動建立服務角色，您必須知道將擔任該角色之服務的服務主體。服務主體是用來將許可授與給服務的識別符。服務主體是由服務定義。

您可以檢查下列項目，找到某些服務的服務主體：

1. 打開 AWS 與 IAM 搭配使用的服務.
2. 請檢查服務是否在 Service-linked roles (服務連結角色) 欄位中為 Yes (是)。
3. 選擇是連結，以檢視該服務的服務連結角色文件。
4. 尋找該服務的 Service-Linked Role Permissions (服務連結角色許可) 區段，以檢視服務主體。

您可以使用 AWS CLI 命令或 AWS API 操作手動建立服務角色。若要使用 IAM 主控台手動建立服務角色，請完成下列任務：

1. 使用您的帳戶 ID 建立 IAM 角色。請勿連接政策或授與任何許可。如需詳細資訊，請參閱 建立角色以委派許可給 IAM 使用者。
2. 開啟角色並編輯信任關係。角色必須信任服務，而不是信任帳戶。例如，更新下列 `Principal` 元素：

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

將主體變更為服務的值，例如 IAM。

```
"Principal": { "Service": "iam.amazonaws.com" }
```

3. 將許可政策連接至角色，以新增服務所需的許可。
4. 返回需要許可的服務，並使用文件中描述的方法來通知服務有關新的服務角色。

對 IAM 和 Amazon EC2 進行故障診斷

使用此處的資訊，來協助您針對在使用 Amazon EC2 和 IAM 時可能遇到的拒絕存取或其他問題，進行故障診斷與修正。

主題

- 嘗試啟動執行個體時，我看不到預期在 Amazon EC2 主控台 IAM 角色清單中看到的角色
- 我的執行個體上的憑證針對錯誤的角色
- 當我嘗試呼叫 AddRoleToInstanceProfile 時，出現 AccessDenied 錯誤
- Amazon EC2：當我嘗試使用角色啟動執行個體時，出現 AccessDenied 錯誤。
- 我無法存取 EC2 執行個體上的暫時安全憑證
- IAM 子目錄中的 info 文件的錯誤是什麼意思？

嘗試啟動執行個體時，我看不到預期在 Amazon EC2 主控台 IAM 角色清單中看到的角色

請檢查以下內容：

- 如果您是以 IAM 使用者身分登入，請確認您有呼叫 `ListInstanceProfiles` 的許可。如需使用角色所需許可的資訊，請參閱 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可 中的「以 Amazon EC2 使用角色所需的許可」。如需新增許可給使用者的詳細資訊，請參閱 管理 IAM 政策。

  如果您無法修改自己的許可，則必須聯絡可以操作 IAM 的管理員以更新您的許可。

- 如果已使用 IAM CLI 或 API 建立角色，請確認您已建立執行個體描述檔，並將該角色新增到該執行個體描述檔。此外，如果您以不同方式命名角色和執行個體描述檔，則在 Amazon EC2 主控台的 IAM 角色清單中將看不到正確的角色名稱。Amazon EC2 主控台中的 IAM 角色清單列出執行個體描述檔的名稱，而不是角色的名稱。您必須選擇包含所需角色的執行個體設定檔的名稱。如需執行個體設定檔的詳細資訊，請參閱 使用執行個體設定檔。

> **Note**
>
> 如果使用 IAM 主控台來建立角色，則無需使用執行個體描述檔。對於您在 IAM 主控台中建立的每個角色，將建立一個與角色同名的執行個體描述檔，該角色將自動新增到該執行個體描述檔中。執行個體描述檔只能包含一個 IAM 角色，並且無法增加該限制。

我的執行個體上的憑證針對錯誤的角色

在執行個體設定檔的角色最近可能已遭替換。若是如此，則您的應用程式將需要等待下一次自動排程的憑證輪換，您的角色憑證才可用。

若要強制變更，必須取消關聯執行個體描述檔，然後關聯執行個體描述檔，或者可以停止執行個體，然後重新啟動它。

當我嘗試呼叫 **AddRoleToInstanceProfile** 時，出現 **AccessDenied** 錯誤

如果您以 IAM 使用者身分提出請求，請確認您具有以下許可：

- `iam:AddRoleToInstanceProfile` 與資源符合執行個體設定檔 ARN (例如，`arn:aws:iam::999999999999:instance-profile/ExampleInstanceProfile`)。

如需有關處理角色所需許可的詳細資訊，請參閱「如何開始使用？」 在 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可 中。如需新增許可給使用者的詳細資訊，請參閱 管理 IAM 政策。

Amazon EC2：當我嘗試使用角色啟動執行個體時，出現 **AccessDenied** 錯誤。

請檢查以下內容：

- 啟動沒有執行個體設定檔的執行個體。這有助於確保問題僅限於 Amazon EC2 執行個體的 IAM 角色。
- 如果您以 IAM 使用者身分提出請求，請確認您具有以下許可：
  - `ec2:RunInstances` 使用萬用字元資源 (「*」)
  - `iam:PassRole` 與資源符合角色 ARN (例如，`arn:aws:iam::999999999999:role/ExampleRoleName`)
- 呼叫 IAM `GetInstanceProfile` 動作，以確保您使用的是有效的執行個體描述檔名稱或有效的執行個體描述檔 ARN。如需詳細資訊，請參閱在 Amazon EC2 執行個體中使用 IAM 角色。
- 呼叫 IAM `GetInstanceProfile` 動作，以確保執行個體描述檔具有角色。空白執行個體設定檔將失敗，並顯示 `AccessDenied` 錯誤。如需建立角色的詳細資訊，請參閱 建立 IAM 角色。

如需有關處理角色所需許可的詳細資訊，請參閱「如何開始使用？」 在 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可 中。如需新增許可給使用者的詳細資訊，請參閱 管理 IAM 政策。

我無法存取 EC2 執行個體上的暫時安全憑證

若要存取 EC2 執行個體上的暫時安全憑證，您必須先使用 IAM 主控台建立角色。然後啟動使用該角色的 EC2 執行個體，並檢查執行中的執行個體。如需詳細資訊，請參閱 使用 IAM 角色為在 Amazon EC2 執行個體上執行的應用程式授予許可 中的我該如何開始？

如果您仍然無法存取 EC2 執行個體上的暫時安全憑證，請檢查下列項目：

- 您是否可以存取 Instance Metadata Service (IMDS) 的另一部分？ 如果沒有，請檢查您沒有封鎖對 IMDS 請求存取的防火牆規則。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/
hostname; echo
```

- IMDS 的 `iam` 子目錄是否存在？ 如果不可以，請呼叫 EC2 `DescribeInstances` API 操作或使用 `aws ec2 describe-instances` CLI 命令，確認您的執行個體具有與其相關聯的 IAM 執行個體設定檔。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam;
 echo
```

- 檢查 IAM 子目錄中的 `info` 文件是否有錯誤。如果您有錯誤，請參閱 IAM 子目錄中的 `info` 文件的錯誤是什麼意思？ 以取得更多資訊。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/
info; echo
```

IAM 子目錄中的 **info** 文件的錯誤是什麼意思？

此 **iam/info** 文件表示 **"Code":"InstanceProfileNotFound"**

您的 IAM 執行個體描述檔已被刪除，Amazon EC2 無法再為您的執行個體提供憑證。您必須將有效的執行個體描述檔連接至 Amazon EC2 執行個體。

如果存在具有該名稱的執行個體描述檔，則請檢查該執行個體描述檔是否已被刪除而建立了另一個同名的執行個體描述檔。

1. 呼叫 IAM `GetInstanceProfile` 操作以取得 `InstanceProfileId`。
2. 呼叫 Amazon EC2 `DescribeInstances` 操作以取得該執行個體的 `IamInstanceProfileId`。
3. 確認來自 IAM 操作的 `InstanceProfileId` 是否符合來自 Amazon EC2 操作的 `IamInstanceProfileId`。

如果 ID 不同，則連接到執行個體的執行個體設定檔不再有效。您必須將有效的執行個體設定檔連接至執行個體。

此 **iam/info** 文件表示成功，但指出 **"Message":"Instance Profile does not contain a role..."**

IAM `RemoveRoleFromInstanceProfile` 動作已從執行個體描述檔中移除該角色。您可以使用 IAM `AddRoleToInstanceProfile` 動作將角色連接到執行個體描述檔。您的應用程式需要等到下一次排程的重新整理才能存取該角色的憑證。

若要強制變更，必須取消關聯執行個體描述檔，然後關聯執行個體描述檔，或者可以停止執行個體，然後重新啟動它。

此 **iam/security-credentials/[role-name]** 文件表示 **"Code":"AssumeRoleUnauthorizedAccess"**

Amazon EC2 沒有可以擔任該角色的許可。擔任角色的許可由連接到角色的信任政策控制，如下面的範例所示。使用 IAM `UpdateAssumeRolePolicy` API 來更新信任政策。

```
{"Version": "2012-10-17","Statement": [{"Effect": "Allow","Principal": {"Service":
 ["ec2.amazonaws.com"]},"Action": ["sts:AssumeRole"]}]}
```

您的應用程式需要等到下一次自動排程的重新整理才能存取該角色的憑證。

若要強制變更，必須取消關聯執行個體描述檔，然後關聯執行個體描述檔，或者可以停止執行個體，然後重新啟動它。

對 Amazon S3 和 IAM 進行故障診斷

使用此處的資訊，協助您對使用 Amazon S3 和 IAM 時可能遇到的問題進行故障診斷和修正。

如何授與對 Amazon S3 儲存貯體的匿名存取權？

您使用在 `principal` 元素中指定萬用字元 (*) 的 Amazon S3 儲存貯體政策，這表示任何人都可以存取儲存貯體。透過匿名存取，任何人 (包括沒有使用者 AWS 帳戶) 都可以存取值區。如需範例政策，請參閱 Amazon Simple Storage Service 使用者指南中的 Amazon S3 儲存貯體政策之範例案例。

我以 AWS 帳戶 root 使用者身分登入；為什麼我無法存取帳戶下的 Amazon S3 儲存貯體？

在某些情況下，您可能擁有對 IAM 和 Amazon S3 具有完全存取權的 IAM 使用者。如果 IAM 使用者將儲存貯體政策指派給 Amazon S3 儲存貯體，但未將儲存貯體指定 AWS 帳戶根使用者 為主體，則系統會拒絕根使用者存取該儲存貯體。但作為根使用者，您仍然可以存取儲存貯體。若要這麼做，請修改儲存貯體政策來允許從 Amazon S3 主控台或 AWS CLI進行根使用者存取。請使用下列主體，將 *123456789012* 取代為 AWS 帳戶的 ID。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

對 SAML 2.0 聯合進行疑難排解 AWS

使用此處的資訊，來協助您針對在使用 SAML 2.0 和 IAM 的聯合身分時所可能遇到的問題，進行故障診斷與排除。

主題

- 錯誤：您的請求包含無效的 SAML 回應。若要登出，請按一下這裡。
- 錯誤： RoleSessionName AuthnResponse （服務： AWSSecurityTokenService；狀態碼：400；錯誤代碼： InvalidIdentityToken）

- 錯誤：未授權執行 STS：使AssumeRole用 SAML（服務： AWSSecurityTokenService；狀態碼：403；錯誤代碼： ）AccessDenied
- 錯誤： RoleSessionName 在 AuthnResponse 必須匹配 [A-ZA-Z_0-9 + = ，.@-] {2,64}（服務：AWSSecurityTokenService; 狀態碼：400; 錯誤代碼： ）InvalidIdentityToken
- 錯誤:來源身份必須符合 [A-ZA-Z_0-9+=, .@-] {2,64}，而不是以 "aws:" (服務:AWSSecurityTokenService; 狀態碼:400; 錯誤碼:) InvalidIdentityToken
- 錯誤:回應簽章無效 (服務: AWSSecurityTokenService；狀態碼：400；錯誤碼: InvalidIdentityToken)
- 錯誤:無法假定角色:發行者不存在於指定的提供者 (服務: AWSOpenIdDiscoveryService; 狀態碼:400; 錯誤碼: AuthSamlInvalidSamlResponseException)
- 錯誤：無法剖析中繼資料。
- 錯誤：指定的供應商不存在。
- 錯誤：要求 DurationSeconds 超過此角色 MaxSessionDuration 設定的值。
- 錯誤：回應沒有包括必要的對象。
- 如何在瀏覽器中查看 SAML 回應以排除故障

錯誤：您的請求包含無效的 SAML 回應。若要登出，請按一下這裡。

當身分提供者的 SAML 回應不包含 Name 設定為 `https://aws.amazon.com/SAML/Attributes/Role` 的屬性時，可能會發生此錯誤。該屬性必須包含一個或多個 AttributeValue 元素，每個元素都包含逗號分隔的字串對：

- 使用者可以映射到角色的 ARN
- SAML 供應商的 ARN

如需更多詳細資訊，請參閱 設定驗證回應的 SAML 宣告。若要在瀏覽器中查看 SAML 回應，請按照 如何在瀏覽器中查看 SAML 回應以排除故障 中列出的步驟操作。

錯誤： RoleSessionName AuthnResponse （服務：AWSSecurityTokenService；狀態碼：400；錯誤代碼：InvalidIdentityToken ）

當身分提供者的 SAML 回應不包含 Name 設定為 `https://aws.amazon.com/SAML/Attributes/RoleSessionName` 的屬性時，可能會發生此錯誤。屬性值是使用者的識別符，通常是使用者 ID 或電子郵件地址。

如需更多詳細資訊，請參閱 設定驗證回應的 SAML 宣告。若要在瀏覽器中查看 SAML 回應，請按照 如何在瀏覽器中查看 SAML 回應以排除故障 中列出的步驟操作。

錯誤：未授權執行 STS：使AssumeRole用 SAML（服務：AWSSecurityTokenService；狀態碼：403；錯誤代碼：）AccessDenied

如果 SAML 回應中指定的 IAM 角色拼寫錯誤或不存在，則會發生此錯誤。請務必使用與您角色完全相同的名稱，因為角色名稱區分大小寫。修正 SAML 服務供應商組態中角色的名稱。

只有當您的角色信任政策包含 `sts:AssumeRoleWithSAML` 動作時，才允許您存取。如果您的 SAML 聲明設定為使用 `PrincipalTag` 屬性，則您的信任政策也必須包含 `sts:TagSession` 動作。如需有關工作階段標籤的詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。

如果您的角色信任政策中沒有 `sts:SetSourceIdentity` 許可，此錯誤可能發生。如果您的 SAML 聲明設定為使用 `SourceIdentity` 屬性，則您的信任政策也必須包含 `sts:SetSourceIdentity` 動作。如需來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

如果聯合身分使用者沒有擔任該角色的許可，也會發生此錯誤。該角色必須具有任政策，該政策將 IAM SAML 身分提供者的 ARN 指定為 `Principal`。該角色還包含控制哪些使用者可以擔任該角色的條件。確認您的使用者滿足條件的要求。

如果 SAML 回應不包括其中包含 `NameID` 的 `Subject`，也會發生此錯誤。

如需詳細資訊，請參閱在 AWS 中為聯合身分使用者建立許可和 設定驗證回應的 SAML 宣告。若要在瀏覽器中查看 SAML 回應，請按照 如何在瀏覽器中查看 SAML 回應以排除故障 中列出的步驟操作。

錯誤： RoleSessionName 在 AuthnResponse 必須匹配 [A-ZA-Z_0-9 + = ，.@-] {2,64}（服務： AWSSecurityTokenService; 狀態碼：400; 錯誤代碼： ）InvalidIdentityToken

如果 `RoleSessionName` 屬性值太長或包含無效的字元，則會發生此錯誤。最大有效長度為 64 個字元。

如需更多詳細資訊，請參閱 設定驗證回應的 SAML 宣告。若要在瀏覽器中查看 SAML 回應，請按照 如何在瀏覽器中查看 SAML 回應以排除故障 中列出的步驟操作。

錯誤:來源身份必須符合 [A-ZA-Z_0-9+=, .@-] {2,64}，而不是以 **"aws:"** (服務: AWSSecurityTokenService; 狀態碼:400; 錯誤碼:) InvalidIdentityToken

如果 `sourceIdentity` 屬性值太長或包含無效的字元，則會發生此錯誤。最大有效長度為 64 個字元。如需來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

如需建立 SAML 聲明的詳細資訊，請參閱 設定驗證回應的 SAML 宣告。若要在瀏覽器中查看 SAML 回應，請按照 如何在瀏覽器中查看 SAML 回應以排除故障 中列出的步驟操作。

錯誤:回應簽章無效 (服務: AWSSecurityTokenService；狀態碼：400；錯誤碼: InvalidIdentityToken)

當身分提供者的聯合中繼資料與 IAM 身分提供者的中繼資料不相符時，則會發生此錯誤。例如，身分服務供應商的中繼資料檔案可能已變更以更新過期的憑證。從身分服務供應商處下載更新的 SAML 中繼資料檔案。然後使用`aws iam update-saml-provider`跨平台 CLI 命令或`Update-IAMSAMLProvider` PowerShell指令程式，在 IAM 中定義的 AWS 身分識別提供者實體中更新它。

錯誤:無法假定角色:發行者不存在於指定的提供者 (服務: AWSOpenIdDiscoveryService; 狀態碼:400; 錯誤碼: AuthSamlInvalidSamlResponseException)

如果 SAML 回應中的發行者與在聯合中繼資料檔案內宣告的發行者不相符，便會發生此錯誤。當您在 IAM 中建立身分提供者 AWS 時，中繼資料檔案已上傳至。

錯誤：無法剖析中繼資料。

如果沒有採用正確格式的中繼資料檔案，則可能會發生此錯誤。

在中建立或管理 SAML 身分識別提供者時 AWS Management Console，您必須從身分識別提供者擷取 SAML 中繼資料文件。

此中繼資料檔案包括發行者名稱、過期資訊以及可用於驗證從 IdP 接收的 SAML 身分驗證回應 (聲明) 的金鑰。中繼資料檔案必須以 UTF-8 格式編碼，並且不含位元組順序記號 (BOM)。若要移除 BOM，您可以使用文字編輯工具，例如 Notepad++，將檔案編碼為 UTF-8。

包含在 SAML 中繼資料文件中的 x.509 憑證必須使用大小至少有 1024 位元的金鑰。此外，x.509 憑證也必須沒有任何重複的擴充。您可以使用擴充，但這些擴充只能在憑證中出現一次。如果 x.509 憑證不符合任一條件，IdP 建立會失敗，並傳回「無法剖析中繼資料」錯誤。

如 SAML V2.0 中繼資料互通性設定檔版本 1.0 所定義，IAM 既不會評估中繼資料文件的 X.509 憑證是否過期，也不會對其採取動作。

錯誤：指定的供應商不存在。

如果您在 SAML 聲明指定的供應商名稱與在 IAM 中設定的供應商名稱不符，就可能發生此錯誤。如需檢視供應商名稱的詳細資訊，請參閱 在 IAM 中建立 SAML 身分識別提供者。

錯誤：要求 DurationSeconds 超過此角色 MaxSessionDuration 設定的值。

如果您擔 AWS CLI 任或 API 中的角色，則可能會發生此錯誤。

當您使用具有-saml CLI 的假設角色或使用 S AssumeRoleAML API 作業來擔任角色時，您可以指定參數的值。`DurationSeconds`您可以指定從 900 秒 (15 分鐘) 到角色的最大工作階段持續時間設定的值。如果您指定高於此設定的值，操作會失敗。例如，如果您指定 12 小時的工作階段持續時間，但是您的管理員設定最大工作階段持續時間為 6 小時，則您的操作會失敗。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。

錯誤：回應沒有包括必要的對象。

如果對象 URL 與 SAML 組態中的身分提供者不相符，就可能會發生此錯誤。請確定您的身分提供者 (IdP) 依存方識別符完全符合 SAML 組態中提供的對象 URL (實體 ID)。

如何在瀏覽器中查看 SAML 回應以排除故障

以下程序介紹在排除與 SAML 2.0 相關的故障問題時，如何從您的瀏覽器中查看服務供應商的 SAML 回應。

對於所有瀏覽器，請前往可以重現問題的頁面。然後，針對適用的瀏覽器執行以下步驟：

主題

- Google Chrome
- Mozilla Firefox
- Apple Safari
- 如何處理 Base64 編碼的 SAML 回應

Google Chrome

在 Chrome 中查看 SAML 回應

這些步驟已使用 Google Chrome 版本 106.0.5249.103 (正式版本) (arm64) 進行了測試。如果您使用其他版本，則可能需要根據情況調整步驟。

1. 按下 F12 來啟動 Developer Tools (開發人員工具) 主控台。
2. 選取 Network (網路) 標籤，然後選取 Developer Tools (開發人員工具) 視窗左上角的 Preserve log (保留記錄)。
3. 重現問題。
4. (選擇性) 如果 Developer Tools (開發人員工具) 的 Network (網路) 記錄窗格中未顯示 Method (方法) 欄，請在任一欄標籤上按一下滑鼠右鍵，然後選擇 Method (方法) 以新增該欄。
5. 在 Developer Tools (開發人員工具) 的 Network (網路) 記錄窗格中尋找 SAML Post (SAML 文章)。選擇該列，然後檢視頂端的 Payload (承載) 標籤。尋找包含加密請求的 SAMLResponse 元素。關聯值為 Base64 編碼的回應。

Mozilla Firefox

在 Firefox 中查看 SAML 回應

此程序已在 Mozilla Firefox 版本 105.0.3 (64 位元) 上進行了測試。如果您使用其他版本，則可能需要根據情況調整步驟。

1. 按下 F12 來啟動 Web Developer Tools (Web 開發人員工具) 主控台。
2. 選取 Network (網路) 標籤。
3. 在 Web Developer Tools (Web 開發人員工具) 視窗右上角選擇選項 (小齒輪圖示)。選取 Persist logs (持續記錄)。
4. 重現問題。
5. (選擇性) 如果 Web Developer Tools (Web 開發人員工具) 的 Network (網路) 記錄窗格中未顯示 Method (方法) 欄，請在任一欄標籤上按一下滑鼠右鍵，然後選擇 Method (方法) 以新增該欄。
6. 尋找表格中的 POST SAML。選取該列，然後檢視 Request 標籤並尋找 SamlResponse 元素。關聯值為 Base64 編碼的回應。

Apple Safari

在 Safari 中查看 SAML 回應

這些步驟已使用 Apple Safari 版本 16.0 (17614.1.25.9.10, 17614) 進行了測試。如果您使用其他版本，則可能需要根據情況調整步驟。

1. 在 Safari 中啟用 Web Inspector。開啟 Preferences (偏好設定) 視窗，選取 Advanced (進階) 標籤，然後選取 Show Develop menu in the menu bar (在選單列顯示開發選單)。
2. 現在您可以開啟 Web Inspector。在選單列中選擇 Develop (開發)，然後選取 Show Web Inspector (顯示 Web Inspector)。
3. 選取 Network (網路) 標籤。
4. 在 Web Inspector 視窗右上角選擇選項 (包含三條水平線的小圓形圖示)。選取 Preserve logs (保留記錄)。
5. (選擇性) 如果 Web Inspector 的 Network (網路) 記錄窗格中未顯示 Method (方法) 欄，請在任一欄標籤上按一下滑鼠右鍵，然後選擇 Method (方法) 以新增該欄。
6. 重現問題。
7. 尋找表格中的 POST SAML。選擇該列，然後檢視 Headers (標題) 標籤。
8. 尋找包含加密請求的 SAMLResponse 元素。向下捲動到找到含有名稱 `SAMLResponse` 的 `Request Data`。關聯值為 Base64 編碼的回應。

如何處理 Base64 編碼的 SAML 回應

在瀏覽器中找到 Base64 編碼的 SAML 回應元素之後，複製這些元素並使用您偏好的 Base-64 解碼工具來擷取含有 XML 標籤的回應。

> **安全提示**
>
> 由於您查看的 SAML 回應資料可能包含敏感安全資料，我們建議您不要使用線上 base64 解碼器。而是使用安裝在本機電腦上，不會透過網路傳送 SAML 資料的工具。

適用於視窗系統的內建選項 (PowerShell)：

```
PS C:
\> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("base64encodedtext
```

適用於 MacOS 與 Linux 系統的內建選項：

```
$ echo "base64encodedtext" | base64 --decode
```

檢閱解碼檔案中的值

檢閱解碼的 SAML 回應檔案中的值。

- 確認 SAML: nameID 屬性的值與已驗證使用者的使用者名稱相符。
- 檢閱「https://aws.amazon.com/SAML/Attributes/Role」的值。ARN 和 SAML 提供者區分大小寫，而且 ARN 必須符合您帳戶中的資源。
- 檢閱 https://aws.amazon.com/SAML/Attributes/ RoleSession 名稱的值。此值必須符合宣告規則中的值。
- 如果您設定電子郵件地址或帳戶名稱的屬性值，請確定值正確無誤。這些值必須對應於已驗證使用者的電子郵件地址或帳戶名稱。

檢查錯誤並確認配置

檢查值是否包含錯誤，並確認下列組態是否正確。

- 宣告規則符合必要的元素，且所有 ARN 都是正確的。如需詳細資訊，請參閱 使用信賴方信任和新增宣告來設定您的 SAML 2.0 IdP。
- 您已將最新的中繼資料檔案從 IdP 上傳到 SAML 提供者 AWS 中。如需詳細資訊，請參閱 啟用 SAML 2.0 聯合身分的使用者存取 AWS Management Console。
- 您已正確設定 IAM 角色的信任政策。如需更多詳細資訊，請參閱 修改角色。

的參考資訊 AWS Identity and Access Management

使用本節中的這些主題，針對 IAM 和 AWS STS的各種面向來尋找詳細參考資料。

主題

Amazon 資源名稱 (ARN)

Amazon 資源名稱 (ARN) 可以唯一識別 AWS 資源。當您需要明確指定所有資源 (例如 IAM 政策 AWS、Amazon 關聯式資料庫服務 (Amazon RDS) 標籤和 API 呼叫中的資源時，我們需要 ARN。

ARN 格式

以下是 ARN 的一般格式。特定格式視資源而定。若要使用 ARN，請以資源特定資訊取代*##*文字。請注意，有些資源的 ARN 會省略區域、帳戶 ID 或同時省略區域和帳戶 ID。

```
arn:partition:service:region:account-id:resource-id
arn:partition:service:region:account-id:resource-type/resource-id
arn:partition:service:region:account-id:resource-type:resource-id
```

partition

資源所在的分割區。分區是一組區 AWS 域。每個 AWS 帳戶的範圍是一個分區。

以下是支援的分割區：

- `aws`- AWS 地區
- `aws-cn` - 中國區域

- aws-us-gov - AWS GovCloud (US) 區域

service

識別 AWS 產品的服務命名空間。

region

區域代碼。例如，請為美國東部 (俄亥俄) 指定 us-east-2。如需區域代碼清單，請參閱 AWS 一般參考 中的區域端點。

account-id

擁有資源的 AWS 帳號識別碼 (不含連字號)。例如 123456789012。

resource-type

資源類型。例如，虛擬私有雲端 (VPC) 的 vpc。

resource-id

資源識別碼。這是資源名稱、資源 ID 或資源路徑。某些資源識別碼包括父項資源 (sub-resource-type/父資源/子資源) 或限定元，例如版本 (資源類型:資源:資源名稱:限定符)。

範例

IAM 使用者

arn:aws:iam::*123456789012*:user/*johndoe*

SNS 主題

#####:##:##-##-1: example-sns-topic-name

VPC

arn:aws:ec2:*us-east-1*:*123456789012*:vpc/*vpc-0e9801d129EXAMPLE*

查詢資源的 ARN 格式

ARN 的確切格式取決於服務和資源類型。有些資源 ARN 可能包含路徑、變數或萬用字元。若要查詢特定 AWS 資源的 ARN 格式，請開啟「服務授權參考」，開啟服務的頁面，然後瀏覽至資源類型表格。

ARN 中的路徑

資源 ARN 可以包含路徑。例如，在 Amazon S3 中，資源識別符是物件名稱，其中包含正斜線 (/) 來形成路徑。同樣地，IAM 使用者名稱和群組名稱可以包含路徑。IAM 路徑僅允許包含英數字元和下列字元：正斜線 (/)、加號 (+)、等號 (=)、逗號 (,)、句點 (.)、at 符號 (@)、底線 (_) 和連字號 (-)。

在路徑中使用萬用字元

路徑可以包含萬用字元，也就是星號 (*)。例如，如果您要編寫 IAM 政策，您可以使用萬用字元指定路徑為 product_1234 的所有 IAM 使用者，如下所示：

```
arn:aws:iam::123456789012:user/Development/product_1234/*
```

同樣地，您可以指定 user/* 來代表所有使用者，或指定 group/* 來代表所有群組，如以下範例所示：

```
"Resource":"arn:aws:iam::123456789012:user/*"
"Resource":"arn:aws:iam::123456789012:group/*"
```

以下範例顯示的 ARN 是針對其中資源名稱包含路徑的 Amazon S3 儲存貯體：

```
arn:aws:s3:::my_corporate_bucket/*
arn:aws:s3:::my_corporate_bucket/Development/*
```

不正確的萬用字元用法

您不能使用 ARN 部分中指定資源類型的萬用字元，例如 IAM ARN 中的 user 詞彙。例如，不允許下列項目。

```
arn:aws:iam::123456789012:u*   <== not allowed
```

IAM 識別碼

IAM 對使用者、使用者群組、角色、政策和伺服器憑證使用一些不同的識別碼。本節介紹識別碼以及何時使用識別碼。

主題

- 易用名稱和路徑

- IAM ARN
- 唯一識別碼

易用名稱和路徑

當您建立使用者、角色、使用者群組或政策，或上傳伺服器憑證時，請為其提供好記的名稱。範例包括 Bob、 TestApp 1、開發人員 ManageCredentialsPermissions、或 ProdServerCert。

如果您使用 IAM API 或 AWS Command Line Interface (AWS CLI) 建立 IAM 資源，則可以新增選用路徑。您可以使用單一路徑，也可以巢狀處理多個路徑作為資料夾結構。例如，您可以使用巢狀路徑 `/division_abc/subdivision_xyz/product_1234/engineering/` 以符合您的公司的組織結構。然後，您可以建立一個政策，以允許該路徑中的所有使用者存取政策模擬器 API。若要查看此政策，請參閱 IAM：根據使用者路徑存取政策模擬器 API。如需如何指定易記名稱的資訊，請參閱使用者 API 文件。如需有關如何使用路徑的其他範例的詳細資訊，請參閱 IAM ARN。

當您使 AWS CloudFormation 用建立資源時，您可以指定使用者、使用者群組、角色以及客戶管理策略的路徑。

如果您的使用者和使用者群組位於相同路徑中，IAM 不會自動將使用者置於該使用者群組中。例如，您可以建立一個開發人員使用者群組並將其路徑指定為 `/division_abc/subdivision_xyz/product_1234/engineering/`。如果您建立一個名為 Bob 的使用者，並為他新增相同的路徑，則此操作不會自動將 Bob 置於 Developers 使用者群組。IAM 不會根據使用者或使用者群組的路徑強制執行使用者或使用者群組之間的任何邊界。如果具有不同路徑的用戶被授予對這些資源的許可，則他們可以使用相同的資源。 AWS 帳戶中 IAM 資源的數量和大小有限。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。

IAM ARN

大多數的資源都有易用名稱 (例如，使用者名稱 Bob 或使用者群組名稱 Developers)。但是，許可政策語言要求您使用下列 Amazon Resource Name (ARN) 格式指定一或多個資源。

```
arn:partition:service:region:account:resource
```

其中：

- partition 辨識資源所在的分割區。對於標準 AWS 區域，分割區為 aws。如果您有其他分割區的資源，則該分割區為 aws-*partitionname*。例如，中國 (北京) 區域的資源分割區，即為 aws-cn。您無法在不同分割區的帳戶之間委派存取權。

- `service`識別 AWS 產品。IAM 資源一律使用 `iam`。
- `region` 會識別資源的區域。對 IAM 資源而言，此項目一律空白。
- `account`指定沒有連字號的 AWS 帳戶 ID。
- `resource` 是按名稱辨識特定資源。

您可以使用下列語法指定 IAM 和 AWS STS ARN。ARN 的區域部分是空白的，因為 IAM 資源是全域。

語法：

```
arn:aws:iam::account:root
arn:aws:iam::account:user/user-name-with-path
arn:aws:iam::account:group/group-name-with-path
arn:aws:iam::account:role/role-name-with-path
arn:aws:iam::account:policy/policy-name-with-path
arn:aws:iam::account:instance-profile/instance-profile-name-with-path
arn:aws:sts::account:federated-user/user-name
arn:aws:sts::account:assumed-role/role-name/role-session-name
arn:aws:sts::account:self
arn:aws:iam::account:mfa/virtual-device-name-with-path
arn:aws:iam::account:u2f/u2f-token-id
arn:aws:iam::account:server-certificate/certificate-name-with-path
arn:aws:iam::account:saml-provider/provider-name
arn:aws:iam::account:oidc-provider/provider-name
```

以下許多範例包括 ARN 的資源部分中的路徑。無法在 AWS Management Console中建立或操控路徑。若要使用路徑，您必須使用 AWS API AWS CLI、或 Windows 適用的工具來處理資源 PowerShell。

範例：

```
arn:aws:iam::123456789012:root
arn:aws:iam::123456789012:user/JohnDoe
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
arn:aws:iam::123456789012:group/Developers
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
arn:aws:iam::123456789012:role/S3Access
arn:aws:iam::123456789012:role/application_abc/component_xyz/RDSAccess
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
```

```
arn:aws:iam::123456789012:role/service-role/QuickSightAction
arn:aws:iam::123456789012:policy/UsersManageOwnCredentials
arn:aws:iam::123456789012:policy/division_abc/subdivision_xyz/UsersManageOwnCredentials
arn:aws:iam::123456789012:instance-profile/Webserver
arn:aws:sts::123456789012:federated-user/JohnDoe
arn:aws:sts::123456789012:assumed-role/Accounting-Role/JaneDoe
arn:aws:sts::123456789012:self
arn:aws:iam::123456789012:mfa/JaneDoeMFA
arn:aws:iam::123456789012:u2f/user/JohnDoe/default (U2F security key)
arn:aws:iam::123456789012:server-certificate/ProdServerCert
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/
ProdServerCert
arn:aws:iam::123456789012:saml-provider/ADFSProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/
a1b2c3d4567890abcdefEXAMPLE11111
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

下列範例提供更多詳細資訊，協助您瞭解不同類型 IAM 和 AWS STS 資源的 ARN 格式。

- 帳戶中的 IAM 使用者：

  > **Note**
  >
  > 每個 IAM 使用者名稱都是唯一的。對於使用者，使用者名稱不區分大小寫 (例如在登入程序期間)，但在政策中或作為 ARN 的一部分使用時，該使用者名稱會區分大小寫。

  ```
  arn:aws:iam::123456789012:user/JohnDoe
  ```

- 另一個具有反映組織結構圖之路徑的使用者：

  ```
  arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/JaneDoe
  ```

- IAM 使用者群組：

  ```
  arn:aws:iam::123456789012:group/Developers
  ```

- 具有路徑的 IAM 使用者群組：

  ```
  arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
  ```

- IAM 角色：

```
arn:aws:iam::123456789012:role/S3Access
```

- 服務連結角色：

```
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
```

- 服務角色：

```
arn:aws:iam::123456789012:role/service-role/QuickSightAction
```

- 受管政策：

```
arn:aws:iam::123456789012:policy/ManageCredentialsPermissions
```

- 可以與 Amazon EC2 執行個體建立關聯的執行個體設定檔：

```
arn:aws:iam::123456789012:instance-profile/Webserver
```

- 在 IAM 中識別為 "Paulo" 的聯合身分使用者：

```
arn:aws:sts::123456789012:federated-user/Paulo
```

- 有人擔任「會計 - 角色」角色，角色工作階段名稱為 "Mary" 的活動工作階段：

```
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Mary
```

- 在呼叫工作階段上運作的 API 呼叫 (例如 API) 中用作資源時，代表呼叫者自己的工作階段： AWS STS SetContext

```
arn:aws:sts::123456789012:self
```

- 指派給名為 Jorge 之使用者的多重要素驗證裝置：

```
arn:aws:iam::123456789012:mfa/Jorge
```

- 伺服器憑證：

```
arn:aws:iam::123456789012:server-certificate/ProdServerCert
```

- 具有反映組織結構圖之路徑的伺服器憑證：

```
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/
ProdServerCert
```

- 身分提供者 (SAML 和 OIDC)：

```
arn:aws:iam::123456789012:saml-provider/ADFSProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

- OIDC 身分識別提供者，具有可反映 Amazon EKS OIDC 身分識別提供者 URL 的路徑：

```
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/
a1b2c3d4567890abcdefEXAMPLE11111
```

另一個重要的 ARN 是根使用者 ARN。雖然這不是 IAM 資源，仍建議您熟悉此 ARN 的格式。它通常會用在資源型政策的 Principal 元素。

- AWS 帳戶 會顯示下列項目：

```
arn:aws:iam::123456789012:root
```

以下範例顯示您可以指派給 Richard 的政策，以允許他管理自己的存取金鑰。請注意，該資源是 IAM 使用者 Richard。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageRichardAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/division_abc/subdivision_xyz/Richard"
        },
        {
            "Sid": "ListForConsole",
```

```
            "Effect": "Allow",
            "Action": "iam:ListUsers",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 當您使用 ARN 識別 IAM 政策中的資源時，可以包含政策變數。政策變數可以包含執行時間資訊的預留位置 (例如使用者名稱)，作為 ARN 的一部分。如需詳細資訊，請參閱IAM 政策元素：變數與標籤

在 ARN 中使用萬用字元和路徑

您可以在 ARN 的##部分中使用萬用字元來指定多個使用者、使用者群組或政策。例如，若要指定在 product_1234 上工作的所有使用者，則使用：

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

如果您有使用者的名稱是以字串 app_ 開頭，您可以使用以下 ARN 引用他們。

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

若要指定您的中的所有使用者、使用者群組或策略 AWS 帳戶，請在user/group/、或policy/部分 ARN 之後分別使用萬用字元。

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
arn:aws:iam::123456789012:policy/*
```

如果您為使用者 arn:aws:iam::111122223333:user/* 指定以下 ARN，即符合以下兩個範例。

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

但是，如果您為使用者 arn:aws:iam::111122223333:user/division_abc* 指定以下 ARN，則符合第二個範例，但不符合第一個範例。

```
arn:aws:iam::111122223333:user/JohnDoe
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

不要在 ARN 的 user/、group/ 或 policy/ 部分使用萬用字元。例如，IAM 不允許下列項目：

```
arn:aws:iam::123456789012:u*
```

Example 將路徑和 ARN 用於以專案為基礎之使用者群組的範例

無法在 AWS Management Console中建立或操控路徑。若要使用路徑，您必須使用 AWS API AWS CLI、或 Windows 工具來處理資源 PowerShell。

在這個範例中，Marketing_Admin 使用者群組中的 Jules 在 /marketing/ 路徑中建立了專案類型使用者群組。Jules 會將來自公司不同部門的使用者指派至使用者群組。此範例說明使用者的路徑與使用者所在的使用者群組無關。

這個行銷群組有一個他們將推出的新產品，因此 Jules 在 /marketing/ path 中建立一個名為 Widget_Launch 的新的使用者群組。然後，Jules 將以下政策指派給使用者群組，讓該使用者群組存取屬於指定給此特定啟動之 example_bucket 的物件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::example_bucket/marketing/newproductlaunch/widget/*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket*",
      "Resource": "arn:aws:s3:::example_bucket",
      "Condition": {"StringLike": {"s3:prefix": "marketing/newproductlaunch/widget/*"}}
    }
  ]
}
```

然後，將此 Jules 將正在進行此次啟動的使用者指派給該使用者群組。這包括來自 /marketing/ 路徑的 Patricia 和 Eli。也包括來自 /sales/ 路徑的 Chris 和 Chloe，以及 /legal/ 路徑的 Aline 和 Jim。

唯一識別碼

當 IAM 建立使用者、使用者群組、角色、政策、執行個體設定檔或伺服器憑證時，它會指派給每個資源唯一的 ID。唯一 ID 類似於如下：

AIDAJQABLZS4A3QDU576Q

在大多數情況下，當您使用 IAM 資源時，使用的都是易記名稱和 ARN。如此即不需要知道特定資源的唯一 ID。但是，當使用易用名稱不可行時，唯一 ID 有時非常有用的。

一個例子在您的 AWS 帳戶. 在您的帳戶、使用者、使用者群組、角色或政策的易用名稱必須是唯一的。例如，您可以建立名為 John 的 IAM 使用者。您的公司使用 Amazon S3，且有一個儲存貯體包含每位員工的資料夾。IAM 使用者 John 是 名為 User-S3-Access 的 IAM 使用者群組成員，其具有允許使用者僅存取儲存貯體中自己的資料夾的許可。如需如何建立身分型政策的範例，該政策允許 IAM 使用者採用使用者易記的名稱在 S3 中存取自己的儲存貯體物件，請參閱 Amazon S3：可讓 IAM 使用者以程式設計方式和在主控台存取其 S3 主目錄。

假設名為 John 的員工離開您的公司並刪除名為 John 的對應 IAM 使用者。但後來另一位名叫 John 的員工開始建立一個名為 John 的新 IAM 使用者。將名為 John 的 IAM 使用者新增至現有的 IAM 使用者群組 User-S3-Access。如果與使用者群組關聯的政策指定友好的 IAM 使用者名稱 John，則該政策會允許新的 John 存取前任 John 留下的資訊。

一般而言，我們建議您為政策中的資源指定 ARN，而非其唯一的 ID。但是，即使您建立一個重新使用之前刪除的易用名稱的新 IAM 使用者，每個 IAM 使用者都有一個唯一的 ID。在此範例中，舊 IAM 使用者 John 和新 IAM 使用者 John 具有不同的唯一 ID。您可以建立以資源為基礎的政策，依唯一 ID 授予存取權，而不僅僅依使用者名稱授予存取權。為免無意中將資訊存取權授予不該擁有此存取權的員工，這樣做可以降低這種情況發生的機率。

下列範例展示您如何在資源型政策的 Principal 元素中指定唯一的 ID。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:role/role-name",
    "AIDACKCEVSQ6C2EXAMPLE",
    "AROADBQP57FF2AEXAMPLE"
  }
```

下列範例展示您如何使用全域條件索引鍵 aws:userid 在政策的 Condition 元素中指定唯一的 ID。

```
"Condition": {
    "StringLike": {
```

```
      "aws:userId": [
        "AIDACKCEVSQ6C2EXAMPLE",
        "AROADBQP57FF2AEXAMPLE:role-session-name",
        "AROA1234567890EXAMPLE:*",
        "111122223333"
      ]
    }
  }
```

使用者 ID 可以非常有用的另一個範例是，如果您維護自己的 IAM 使用者或角色資訊資料庫 (或其他存放區)。唯一 ID 可為您建立的每個 IAM 使用者或角色提供唯一識別碼。若您有重複使用名稱的 IAM 使用者或角色，就會出現這種情況，如上例所示。

了解唯一 ID 字首

IAM 使用以下字首，指示每個唯一的 ID 適用於哪種類型的資源。前綴可能會根據創建時間而有所不同。

字首	資源類型
ABIA	AWS STS 服務承載令牌
ACCA	環境特定的憑證
AGPA	使用者群組
AIDA	IAM 使用者
AIPA	Amazon EC2 執行個體設定檔
AKIA	存取金鑰
ANPA	受管政策
ANVA	受管政策中的版本
APKA	公有金鑰
AROA	角色
ASCA	憑證

字首	資源類型
ASIA	暫時 (AWS STS) 存取金鑰 ID 使用此字首，但只有在與私密存取金鑰和工作階段權杖結合使用時才是唯一的。

獲取唯一識別碼

IAM 主控台中沒有 IAM 資源的唯一 ID。若要取得唯一 ID，您可以使用下列 AWS CLI 命令或 IAM API 呼叫。

AWS CLI:

- get-caller-identity
- get-group
- get-role
- get-user
- get-policy
- get-instance-profile
- get-server-certificate

IAM API：

- GetCallerIdentity
- GetGroup
- GetRole
- GetUser
- GetPolicy
- GetInstanceProfile
- GetServerCertificate

IAM 和 AWS STS 配額

AWS Identity and Access Management （IAM）和 AWS Security Token Service （STS）具有限制對象大小的配額。這會影響您命名物件的方式、您可以建立的物件數，以及您在傳遞物件時可使用的字元數。

> **Note**
>
> 若要取得有關 IAM 用量和配額的帳戶層級資訊，請使用 GetAccountSummaryAPI 操作或命令。get-account-summary AWS CLI

IAM 名稱需求

IAM 名稱具有下列需求與限制：

- 政策文件只能包含下列 Unicode 字元：水平定位字元 (U+0009)、換行字元 (U+000A)、歸位字元 (U+000D)，以及 U+0020 到 U+00FF 範圍內的字元。
- 使用者、群組、角色、政策、執行個體設定檔、伺服器憑證和路徑的名稱必須是英數字元，包括以下常見的字元：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、at 符號 (@)、底線 (_) 和連字號 (-)。路徑名稱必須以正斜線 (/) 開頭和結尾。
- 使用者、群組、角色和執行個體設定檔的名稱在帳戶中必須是唯一的。名稱不分大小寫。例如：您無法同時建立名為 **`ADMINS`** 和 **`admins`** 的群組。
- 第三方用於擔任角色的外部 ID 值必須最少為 2 個字元，最多為 1,224 個字元。該值必須為英數字元，且不包含空格。也可以包含下列符號：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、小老鼠 (@)、冒號 (:)、正斜線 (/) 和連字號 (-)。如需有關外部 ID 的詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。
- 內嵌政策的政策名稱必須對其內嵌的使用者、群組或角色是唯一。此名稱可以包含任何基本拉丁文 (ASCII) 字元，但下列預留字元除外：反斜線 (\)、正斜線 (/)、星號 (*)、問號 (?) 和空格。這些字元是根據 RFC 3986 第 2.2 條予以保留。
- 使用者密碼 (登入設定檔) 可以包含任何基本拉丁文 (ASCII) 字元。
- AWS 帳戶 ID 別名在各個 AWS 產品之間必須是唯一的，並且必須是遵循 DNS 命名慣例的英數字元。別名必須為小寫，且不得以連字號開始或結束，不可包含兩個連續的連字號，也不能是 12 位數的號碼。

如需基本拉丁文 (ASCII) 字元的清單，請前往國會圖書館基本拉丁文 (ASCII) 代碼表。

IAM 物件配額

配額 (在中 AWS也稱為限制) 是您的資源、動作和項目的最大值 AWS 帳戶。使用 Service Quotas 來管理您的 IAM 配額。

如需 IAM 服務端點和 Service Quotas 的清單，請參閱 AWS 一般參考 中的 AWS Identity and Access Management 端點和配額。

請求提高配額

1. 按照《AWS 登入使用者指南》如何登入 AWS 主題中所述適合您的使用者類型之登入程序操作來登入 AWS Management Console。
2. 開啟 Service Quotas 主控台。
3. 在導覽窗格中，選擇 AWS services (AWS 服務)。
4. 在導覽列上，選擇 US East (N. Virginia) (美國東部 (維吉尼亞北部)) 區域。然後搜尋 **IAM**。
5. 選擇 AWS Identity and Access Management (IAM)、選擇配額，然後依照指示請求增加配額。

如需詳細資訊，請參閱《Service Quotas 使用者指南》中的請求增加配額。

如要查看如何使用 Service Quotas 主控台要求增加 IAM 配額的範例，請觀看下列影片。

使用 Service Quotas 主控台請求增加 IAM 配額。

您可以為可調整配額申請提升預設 IAM 配額。到達 maximum quota 的請求會自動核准，並在幾分鐘內完成。

下表列出可自動核准配額增加區域的資源。

IAM 資源的可調整配額

資源	預設配額	最大配額
每個帳戶的客戶受管政策	1500	5000
每個帳戶的群組	300	500
每個帳戶的執行個體設定檔	1000	5000
每個角色的受管政策	10	20

資源	預設配額	最大配額
每個使用者的受管政策	10	20
角色信任政策長度	2,048 個字元	4096 個字元
每個帳戶的角色	1000	5000
每個帳戶的伺服器憑證	20	1000

IAM Access Analyzer 配額

如需 IAM Access Analyzer 服務端點和 Service Quotas 的清單，請參閱 AWS 一般參考 中的 IAM Access Analyzer 端點和配額。

IAM Roles Anywhere 配額

如需 IAM Roles Anywhere 服務端點和 Service Quotas 的清單，請參閱 AWS 一般參考 中的 AWS Identity and Access Management Roles Anywhere 端點和配額。

IAM 和 STS 字元限制

以下是 IAM 和 AWS STS的字元數上限和大小上限：您無法請求提高下列限制。

描述	限制
ID 的別 AWS 帳戶 名	3–63 個字元
對於內嵌政策	您可以新增任意數量的內嵌政策到 IAM 使用者、角色或群組。但是每個實體的總計彙總政策大小 (所有內嵌政策的大小總和) 不能超過下列限制： • 使用者政策大小不可超過 2,048 個字元。 • 角色政策大小不可超過 10,240 個字元。 • 群組政策大小不可超過 5,120 個字元。

描述	限制
	Note 在根據這些限制計算政策的大小時，IAM 不會計算空格數。
針對受管政策	• 每個受管政策的大小不可超過 6,144 個字元。 **Note** 在根據此限制計算政策的大小時，IAM 不會計算空格數。
Group name (群組名稱)	128 個字元
執行個體設定檔名稱	128 個字元
登入設定檔的密碼	1–128 個字元
路徑	512 個字元
政策名稱	128 個字元
角色名稱	64 個字元 **Important** 如果您打算在中使用具有「切換角色」功能的角色 AWS Management Console，則合併`Path`且不`RoleName`能超過 64 個字元。

描述	限制
角色工作階段持續時間	12 小時 當您擔 AWS CLI 任或 API 中的角色時，可以使用 `duration-seconds` CLI 參數或 `DurationSeconds` API 參數來要求較長的角色工作階段。您可以指定值從 900 秒 (15 分鐘) 到角色的最長工作階段持續時間設定，範圍為 1–12 小時。如果您不為 `DurationSeconds` 參數指定一個值，則您的安全憑證有效期為 1 小時。在主控台中切換角色的 IAM 使用者會被授予最長工作階段持續時間或使用者工作階段中的剩餘時間，以較短者為準。工作階段持續時間設定上限不會限制 AWS 服務擔任的工作階段。若要了解如何檢視角色的最大值，請參閱 查看角色的最大工作階段持續時間設定。
角色工作階段名稱	64 個字元

描述	限制
角色工作階段政策	• 所傳遞的 JSON 政策文件和所有傳遞的受管政策 ARN 字元的大小總計不可超過 2,048 個字元。 • 當您建立工作階段時，最多可傳遞 10 個受管政策 ARN。 • 您在透過編寫程式的方式為角色或聯合身分使用者建立暫時工作階段時，只能傳遞一個 JSON 政策文件。 • 此外， AWS 轉換會將傳遞的工作階段原則和工作階段標籤壓縮為具有個別限制的封裝二進位格式。`PackedPolicySize` 回應元素會按百分比指出請求的政策和標籤與大小上限的距離。 • 我們建議您使用 AWS CLI 或 AWS API 傳遞工作階段原則。 AWS Management Console 可能會將其他主控台工作階段資訊新增至封裝的政策。
角色工作階段標籤	• 工作階段標籤必須符合 128 個字元的標籤鍵限制，以及 256 個字元的標籤值限制。 • 您最多可以傳遞 50 個工作階段標籤。 • AWS 轉換會將傳遞的工作階段原則和工作階段標籤壓縮為具有單獨限制的封裝二進位格式。您可以使用 AWS CLI 或 AWS API 傳遞工作階段標籤。`PackedPolicySize` 回應元素會按百分比指出請求的政策和標籤與大小上限的距離。
SAML 驗證回應 base64 編碼	100,000 個字元 此字元限制適用於 `assume-role-with-saml` CLI 或 `AssumeRoleWithSAML` API 操作。

描述	限制
標籤鍵	128 個字元 此字元限制適用於 IAM 資源和工作階段標籤上的標籤。
標籤值	256 個字元 此字元限制適用於 IAM 資源和工作階段標籤上的標籤。 標籤值可為空，也就是說標籤值的長度為 0 個字元。
IAM 建立的唯一 ID	128 個字元。例如： • 以 AIDA 開頭的使用者 ID • 以 AGPA 開頭的群組 ID • 以 AROA 開頭的角色 ID • 以 ANPA 開頭的受管政策 ID • 以 ASCA 開頭的伺服器憑證 ID **Note** 這不是詳盡清單，也不保證特定類型的 ID 僅以指定的字母組合開頭。
使用者名稱	64 個字元

介面 VPC 端點

如果您使用 Amazon Virtual Private Cloud (Amazon VPC) 託管資 AWS 源，則可以在 VPC 和 AWS Security Token Service (AWS STS) 之間建立私有連接。您可以使用此連線 AWS STS 來啟用與 VPC 中的資源進行通訊，而無需透過公用網際網路。

Amazon VPC 是一項 AWS 服務，可用於在您定義的虛擬網路中啟動 AWS 資源。您可利用 VPC 來控制您的網路設定，例如 IP 地址範圍、子網路、路由表和網路閘道。若要將 VPC 連接到 AWS STS，您需要為的定義介面 VPC 端點。 AWS STS端點提供可靠、可擴充的連線能力， AWS STS 無需網際網路閘道、網路位址轉譯 (NAT) 執行個體或 VPN 連線。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的什麼是 Amazon VPC？。

介面 VPC 私人雲端端點採用 AWS PrivateLink 一項 AWS 技術，可使用具有私有 IP 位址的 elastic network interface，在 AWS 服務之間進行私人通訊。如需詳細資訊，請參閱「AWS PrivateLink 服 AWS 務」。

以下資訊適用於 Amazon VPC 的使用者。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的 Amazon VPC 入門。

可用性

AWS STS 目前支援下列區域中的 VPC 端點：

- 美國東部 (維吉尼亞北部)
- 美國東部 (俄亥俄)
- 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 (海德拉巴)
- 亞太區域 (雅加達)
- 亞太區域 (墨爾本)
- 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)

- 加拿大西部 (卡加利)
- 中國 (北京)
- 中國 (寧夏)
- 歐洲 (法蘭克福)
- 歐洲 (愛爾蘭)
- 歐洲 (倫敦)
- 歐洲 (米蘭)
- Europe (Paris)
- 歐洲 (西班牙)
- 歐洲 (斯德哥爾摩)
- 歐洲 (蘇黎世)
- 以色列 (特拉維夫)
- Middle East (Bahrain)
- 中東 (阿拉伯聯合大公國)
- 南美洲 (聖保羅)
- AWS GovCloud (美國東部)
- AWS GovCloud (美國西部)

為以下項目建立 VPC 端點 AWS STS

若要開 AWS STS 始使用您的 VPC，請為. AWS STS如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用介面 VPC 端點存取 AWS 服務。

建立 VPC 端點後，您必須使用相符的地區端點來傳送請 AWS STS 求。 AWS STS 建議您同時使用`setRegion`和`setEndpoint`方法來呼叫地區端點。您可以單獨將 `setRegion` 方法用在手動啟用的區域，例如亞太區域 (香港)。在這種情況下，系統會將呼叫導向 STS 區域端點。如要了解如何手動啟用區域，請參閱 AWS 一般參考 中的管理 AWS 區域。如果您單獨將 `setRegion` 方法用在預設啟用的區域，系統會將呼叫導向 `https://sts.amazonaws.com` 的全域端點。

當您使用地區端點時，請使用公用端點或私有介面 VPC 端點 AWS STS 呼叫其他 AWS 服務 (以使用中為準)。例如，假設您已為其建立介面 VPC 端點， AWS STS 且已經 AWS STS 從位於 VPC 中的資源請求臨時登入資料。在這種情況下，依預設這些憑證會開始流過界面 VPC 端點。如需使用提出區域要求的詳細資訊 AWS STS，請參閱AWS STS 在一個管理 AWS 區域。

AWS 與 IAM 搭配使用的服務

以下列出的 AWS 服務按字母順序分組，並包含其支援哪些 IAM 功能的相關資訊：

- 服務 — 您可以選擇服務名稱，以檢視有關該服務的 IAM 授權和存取權的 AWS 文件。
- 動作 – 您可以在政策中指定個別動作。如果服務不支援此功能，則會選取視覺化編輯器中的 All actions (所有動作)。在 JSON 政策文件中，您必須在 * 元素中使用 `Action`。如需每個服務中的動作清單，請參閱服務的動作、資源和條件 AWS 金鑰。
- 資源層級許可 – 您可以使用 ARN 在政策中指定個別的資源。如果服務不支援此功能，則會選擇政策視覺化編輯器中的 所有資源。在 JSON 政策文件中，您必須在 * 元素中使用 `Resource`。有些動作，例如 `List*` 動作，不支援指定 ARN，因為它們是專為傳回多個資源而設計。如果服務因為某些資源但非其他而支援此功能，在表格中會以 Partial 表示它。如需詳細資訊，請參閱該服務的文件。
- 資源型政策 – 您可以將資源型政策連接到服務內的資源。資源型政策包括 `Principal` 元素以指定哪些 IAM 身分可以存取該資源。如需詳細資訊，請參閱 以身分為基礎和以資源為基礎的政策。
- ABAC (依據標籤授權) – 若要依據標籤控制存取，應使用 `aws:ResourceTag/`*`key-name`*、`aws:RequestTag/`*`key-name`* 或 `aws:TagKeys` 條件鍵，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鍵，則值為 Partial。如需有關根據屬性 (例如標籤) 定義許可的詳細資訊，請參閱 ABAC 是做什麼用的 AWS？。若要查看含有設定 ABAC 步驟的教學課程，請參閱使用屬性型存取控制 (ABAC)。
- 臨時登入資料 — 您可以使用使用 IAM 身分中心登入、在主控台中切換角色或使用 AWS CLI 或 AWS API 產生的短期 AWS STS 登入資料。僅在使用長期 IAM 使用者憑證時，您才可以使用 No 值來存取服務。這包括使用者名稱和密碼或您的使用者存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時安全憑證。
- 服務連結角色 – 服務連結角色是一種特殊類型的服務角色，提供服務代表您存取其他服務中資源的許可。選擇是或部分連結，參閱文件以取得支援這些角色的服務。此欄不會指出服務是否使用標準服務角色。如需詳細資訊，請參閱 使用服務連結角色。
- 詳細資訊 – 如果服務不完全支援某功能，您可以檢閱項目的註腳以查看限制和相關資訊的連結。

可搭配 IAM 運作的服務

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Account Management	 是	 是	 否	 否	 是	 否
AWS Activate Console	 是	 否	 否	 否	 是	 否
AWS Amplify 管理員	 是	 是	 否	 否	 是	 否
AWS Amplify	 是	 是	 否	 部分	 是	 否
AWS Amplify UI 生成器	 是	 是	 否	 是	 是	 否
適用於 Amazon MSK 叢集的 Apache Kafka API	 是	 是	 否	 否	 是	 否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon API Gateway	是	是	是	否	是	是
Amazon API Gateway Management	是	是	否	是	是	否
Amazon API Gateway Management V2	是	是	否	是	是	否
AWS App2Container	是	否	否	否	是	否
AWS AppConfig	是	是	否	是	是	否
AWS AppFabric	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon AppFlow	是	是	否	是	是	否
Amazon AppIntegrations	是	是	否	是	是	是
Application Auto Scaling	是	是	否	是	是	是
AWS 應用程式成本分析工具	是	否	否	否	是	否
AWS 應用發現阿森納	是	否	否	否	是	否
AWS Application Discovery Service	是	否	否	否	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Application Migration Service	是	是	否	是	是	是
AWS 應用轉型服務	是	否	否	否	是	否
AWS App Mesh	是	是	否	是	是	是
AWS App Mesh 預覽版	是	是	否	否	是	是
AWS 應用亞軍	是	是	否	是	是	是
Amazon AppStream 2.0	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS AppSync	是	是	否	是	是	否
AWS Artifact	是	是	否	否	是	否
Amazon Athena	是	是	否	是	是	否
AWS Audit Manager	是	是	否	是	是	是
AWS Auto Scaling	是	否	否	否	是	是
AWS B2B 資料交換	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Backup	是	是	是	是	是	是
AWS Backup 閘道器	是	是	否	是	是	否
AWS Backup 儲存	是	否	否	否	是	否
AWS Batch	是	部分	否	是	是	是
Amazon Bedrock	是	是	否	是	是	否
AWS Billing and Cost Management	是	否	否	否	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Billing and Cost Management 資料匯出	是	是	否	是	是	否
AWS Billing Conductor	是	是	否	是	是	否
Amazon Braket	是	是	否	是	是	是
AWS 預算服務	是	是	否	否	否	否
AWS BugBust	是	是	否	是	是	是
AWS Certificate Manager (ACM)	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Chatbot	是	是	否	否	是	是
Amazon Chime	是	是	否	是	是	是
AWS Clean Rooms	是	是	否	是	是	否
AWS Clean Rooms 毫升	是	是	否	是	是	否
AWS Client VPN	是	是	否	否	是	是
AWS Cloud9	是	是	是	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS 雲端 控制 API	是	否	否	否	是	否
Amazon 雲端目錄	是	是	否	否	是	否
AWS CloudFormation	是	是	否	是	是	否
Amazon CloudFront	是	是	否	部分	是	部分 (資訊)
Amazon CloudFront KeyValueStore	是	是	否	否	是	否
AWS CloudHSM	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Cloud Map	是	是	否	是	是	否
Amazon CloudSearch	是	是	否	否	是	否
AWS CloudShell	是	是	否	否	是	否
AWS CloudTrail	是	是	部分 (資訊)	部分 (資訊)	是	是
AWS CloudTrail 資料	是	是	否	是	是	否
Amazon CloudWatch	是	是	否	是	是	部分 (資訊)

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon CloudWatch 應用洞察	是	否	否	否	是	否
Amazon CloudWatch 應用信號	是	是	否	是	是	否
Amazon CloudWatch 顯然	是	是	否	是	是	否
Amazon CloudWatch 網絡監控	是	是	否	是	是	否
Amazon CloudWatch 日誌	是	是	是	部分	是	是
Amazon CloudWatch 網絡監控	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon CloudWatch 觀測訪問管理器	是	是	否	是	是	否
Amazon CloudWatch 朗姆	是	是	否	是	是	否
Amazon CloudWatch Synthetics	是	是	否	是	是	否
AWS CodeArtifact	是	是	是	是	是	否
AWS CodeBuild	是	是	是 (資訊)	部分 (資訊)	是	否
Amazon CodeCatalyst	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS CodeCommit	是	是	否	是	是	否
AWS CodeConnections	是	是	否	是	是	否
AWS CodeDeploy	是	是	否	是	是	否
AWS CodeDeploy 安全主機命令服務	是	否	否	否	是	否
Amazon CodeGuru 分析器	是	是	否	是	是	是
Amazon 評論 CodeGuru 家	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon CodeGuru 安全	是	是	否	是	是	否
AWS CodePipeline	是	部分	否	是	是	否
AWS CodeStar	是	部分	否	是	是	否
AWS CodeStar 連線	是	是	否	是	是	是
AWS CodeStar 通知	是	是	否	是	是	是
Amazon CodeWhisperer	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Cognito	是	是	否	是	是	是
Amazon Cognito Sync	是	是	否	否	是	是
Amazon Cognito 使用者集區	是	是	否	是	是	是
Amazon Comprehend	是	是	否	是	是	否
Amazon Comprehend Medical	是	否	否	否	是	否
AWS Compute Optimizer	是	否	否	否	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Config	是	部分 (資訊)	否	是	是	是
Amazon Connect	是	是	否	是	是	是
Amazon Connect Cases	是	是	否	是	是	否
Amazon Connect Customer Profiles	是	是	否	是	是	是
Amazon Connect 大量對外通訊	是	是	否	是	是	否
Amazon Connect Voice ID	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Console Mobile Application	是	是	否	否	是	否
AWS Consolidated Billing (合併帳單)	是	否	否	否	是	否
AWS 控制目錄	是	是	否	否	是	否
AWS Control Tower	是	是	否	否	是	否
AWS Cost and Usage Report	是	是	否	否	是	否
AWS Cost Explorer	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS 成本最佳化中心	是	否	否	否	是	否
AWS 客戶驗證服務	是	否	否	否	是	否
AWS Database Migration Service	是	是	無 (資訊)	是	是	是
Database Query Metadata Service	是	否	否	否	是	否
AWS Data Exchange	是	是	否	是	是	否
Amazon Data Lifecycle Manager	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Data Pipeline	是	是	否	部分	是	否
AWS DataSync	是	是	否	是	是	是
Amazon DataZone	是	否	否	否	是	否
AWS 截止日期雲	是	是	否	是	是	否
AWS DeepComposer	是	是	否	是	是	否
AWS DeepRacer	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Detective	是	是	否	是	是	否
AWS Device Farm	是	是	否	是	是	是
Amazon DevOps 大師	是	是	否	否	是	是
AWS 診斷工具	是	是	否	是	是	否
AWS Direct Connect	是	是	否	是	是	是
AWS Directory Service	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon DocumentDB Elastic Clusters	是	是	否	是	是	是
Amazon DynamoDB Accelerator (DAX)	是	是	否	否	是	是
Amazon DynamoDB	是	是	是	否	是	否
Amazon Elastic Compute Cloud (Amazon EC2)	是	部分	否	是	是	部分 (資訊)
Amazon EC2 Auto Scaling	是	是	否	是	是	是
EC2 Image Builder	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon EC2 執行個體連線	是	是	否	否	是	是
Amazon ElastiCache	是	是	否	是	是	是
AWS Elastic Beanstalk	是	部分	否	是	是	是
Amazon Elastic Block Store (Amazon EBS)	是	部分	否	是	是	否
Amazon Elastic Container Registry (Amazon ECR)	是	是	是	是	是	是
Amazon Elastic Container Registry 公有 (Amazon ECR 公有)	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Elastic Container Service (Amazon ECS)	是	部分 (資訊)	否	是	是	是
AWS Elastic Disaster Recovery	是	是	否	是	是	是
Amazon Elastic File System (Amazon EFS)	是	是	是	部分	是	是
Amazon Elastic Inference	是	是	否	否	是	否
Amazon Elastic Kubernetes Service (Amazon EKS)	是	是	否	是	是	是
Amazon Elastic Kubernetes Service (Amazon EKS) 授權	是	是	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Elastic Load Balancing	是	部分	否	部分	是	是
Amazon Elastic Transcoder	是	是	否	否	是	否
AWS 元素設備和軟件激活服務	是	是	否	是	是	否
AWS 元素設備和軟件	是	是	否	是	是	否
AWS Elemental MediaConnect	是	是	否	否	是	是
AWS Elemental MediaConvert	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Elemental MediaLive	是	是	否	是	是	否
AWS Elemental MediaPackage	是	是	否	是	是	部分 (資訊)
AWS Elemental MediaPackage V2	是	是	否	是	是	否
AWS Elemental MediaPackage 視頻點播	是	是	否	是	是	部分 (資訊)
AWS Elemental MediaStore	是	是	是	是	是	否
AWS Elemental MediaTailor	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS 元素輔 Support 案件	是	否	否	否	是	否
AWS 元素輔 Support 內容	是	否	否	否	是	否
Amazon EMR	是	是	否	是	是	是
Amazon EMR on EKS	是	是	否	是	是	是
Amazon EMR Serverless	是	是	否	是	是	是
AWS 實體解析度	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon EventBridge	是	是	是	是	是	否
Amazon EventBridge 管道	是	是	否	是	是	否
Amazon EventBridge 排程	是	是	否	是	是	否
Amazon EventBridge 模式	是	是	是	是	是	否
AWS 故障注入服務	是	是	否	是	是	是
Amazon FinSpace	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon FinSpace API	是	是	否	否	是	否
AWS Firewall Manager	是	是	否	是	是	部分
Fleet Hub for AWS IoT Device Management	是	是	否	是	是	否
Amazon Forecast	是	是	否	是	是	否
Amazon Fraud Detector	是	是	否	是	是	否
FreeRTOS	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS 免費方案	是	否	否	否	是	否
Amazon FSx	是	是	否	是	是	是
Amazon GameLift	是	是	否	是	是	否
AWS Global Accelerator	是	是	否	是	是	是
AWS Glue	是	是	是	部分	是	否
AWS Glue DataBrew	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Ground Station	是	是	否	是	是	是
Amazon Ground Truth 標記	是	否	否	否	是	否
Amazon GuardDuty	是	是	否	是	是	是
AWS Health API 和通知	是	是	否	否	是	否
AWS HealthImaging	是	是	否	是	是	否
AWS HealthLake	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS HealthOmics	是	是	否	是	是	否
Amazon Honeycode	是	是	否	否	是	否
AWS IAM Identity Center	是	是	否	部分	是	是
IAM Identity Center 目錄	是	否	否	否	是	否
IAM Identity Center 身分存放區	是	是	否	否	是	否
IAM Identity Center OIDC 服務	是	是	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Identity and Access Management (IAM)	是	是	部分 (資訊)	部分 (資訊)	部分 (資訊)	否
AWS Identity and Access Management 存取分析器	是	是	否	是	是	部分
AWS Identity and Access Management 任何角色	是	是	否	是	是	是
AWS 身分存放區驗證	是	否	否	否	是	否
AWS 身份同步	是	是	否	否	是	否
AWS Import/Export	是	否	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Inspector	是	是	否	是	是	是
Amazon Inspector Classic	是	否	否	否	是	是
Amazon InspectorScan	是	否	否	否	是	否
Amazon Interactive Video Service	是	是	否	是	是	是
Amazon Interactive Video Service Chat	是	是	否	是	是	否
AWS Invoicing	是	否	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS IoT 1-Click	是	是	否	是	是	否
AWS IoT Analytics	是	是	否	是	是	否
AWS IoT	是	是	部分 (資訊)	是	是	否
AWS IoT Core 裝置顧問	是	是	否	是	是	否
AWS IoT 設備測試儀	是	否	否	否	是	否
AWS IoT Events	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS IoT FleetWise	是	是	否	是	是	否
AWS IoT Greengrass	是	是	否	是	是	否
AWS IoT Greengrass V2	是	是	否	部分	是	否
AWS IoT 工作 DataPlane	是	是	否	否	是	否
AWS IoT RoboRunner	是	是	否	否	是	否
AWS IoT SiteWise	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS IoT TwinMaker	是	是	否	是	是	是
AWS IoT Wireless	是	是	否	是	是	否
AWS IQ	是	是	否	否	是	是
AWS IQ 權限	是	是	否	否	是	否
Amazon Kendra	是	是	否	是	是	否
Amazon Kendra Intelligent Ranking	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Key Management Service (AWS KMS)	是	是	是	是	是	是
Amazon Keyspaces (適用於 Apache Cassandra)	是	是	否	是	是	是
Amazon Managed Service for Apache Flink	是	是	否	是	是	否
Amazon Managed Service for Apache Flink V2	是	是	否	是	是	否
Amazon 數據 Firehose	是	是	否	是	是	否
Amazon Kinesis Data Streams	是	是	是	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Kinesis Video Streams	是	是	否	是	是	否
AWS Lake Formation	是	否	否	否	是	是
AWS Lambda	是	是	是	部分 (資訊)	是	部分 (資訊)
AWS Launch Wizard	是	否	否	否	是	否
Amazon Lex	是	是	否	是	是	是
Amazon Lex V2	是	是	是	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS License Manager	是	是	否	是	是	是
AWS License Manager 訂閱管理員	是	否	否	否	是	否
AWS License Manager 使用者訂閱	是	否	否	否	是	是
Amazon Lightsail	是	部分 (資訊)	否	部分 (資訊)	是	是
Amazon Location Service	是	是	否	是	是	否
Amazon Lookout for Equipment	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Lookout for Metrics	是	是	否	是	是	否
Amazon Lookout for Vision	是	是	否	是	是	否
Amazon Machine Learning	是	是	否	否	是	否
Amazon Macie	是	是	否	是	是	是
AWS Mainframe Modernization	是	是	否	是	是	是
AWS Mainframe Modernization 應用測試	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Managed Blockchain	是	是	否	是	是	否
Amazon Managed Blockchain Query	是	否	否	否	是	否
Amazon Managed Grafana	是	是	否	是	是	是
Amazon Managed Service for Prometheus	是	是	否	是	是	否
Amazon Managed Streaming for Apache Kafka (MSK)	是	是	部分 (資訊)	是	是	是
Amazon Managed Streaming for Kafka Connect	是	是	否	否	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Managed Workflows for Apache Airflow	是	是	否	是	是	否
AWS Marketplace	是	否	否	否	是	是
AWS Marketplace 目錄	是	是	否	是	是	否
AWS Marketplace Commerce Analytics	是	否	否	否	否	否
AWS Marketplace 服務部署	是	是	否	是	是	否
AWS Marketplace 探索	是	否	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Marketplace Entitlement Service	是	否	否	否	是	否
AWS Marketplace 形象建立服務	是	否	否	否	是	否
AWS Marketplace 管理入口網站	是	否	否	否	是	否
AWS Marketplace Metering Service	是	否	否	否	是	否
AWS Marketplace 私人 Marketplace	是	否	否	否	是	否
AWS Marketplace 採購系統整合	是	否	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Marketplace 賣家報告	是	是	否	否	是	否
AWS Marketplace 供應商洞察	是	是	否	是	是	否
Amazon Mechanical Turk	是	否	否	否	是	否
Amazon MediaImport	是	否	否	否	否	否
Amazon MemoryDB for Redis	是	是	否	是	是	是
Amazon Message Delivery Service	是	否	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon 消息閘道服務	是	否	否	否	是	否
AWS Microservice Extractor for .NET	是	否	否	否	是	否
AWS Migration Acceleration Program 積分	是	是	否	否	是	否
AWS Migration Hub	是	是	否	否	是	是
AWS Migration Hub 協調器	是	是	否	是	是	是
AWS Migration Hub Refactor Spaces	是	是	是	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Migration Hub Strategy Recommendations	是	否	否	否	是	是
Amazon Monitron	是	是	否	是	是	是
Amazon MQ	是	是	否	是	是	是
Amazon Neptune	是	是	否	否	是	是
Amazon Neptune Analytics	是	是	否	是	是	否
AWS Network Firewall	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Network Manager	是	是	否	是	是	是 (資訊)
AWS Network Manager 聊天	是	否	否	否	是	否
Amazon Nimble Studio	是	是	否	是	是	否
Amazon One Enterprise	是	是	否	是	是	否
Amazon OpenSearch 攝入	是	是	否	是	是	是
Amazon OpenSearch 無服務器	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon OpenSearch 服務	是	是	是	是	是	是
AWS OpsWorks	是	是	否	否	是	否
AWS OpsWorks 組態管理	是	是	否	否	是	否
AWS Organizations	是	是	否	是	否	是
AWS Outposts	是	是	否	是	是	是
AWS Panorama	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Partner 中央帳戶管理	是	否	否	否	是	否
AWS Payment Cryptography	是	是	否	是	是	否
AWS 付款	是	否	否	否	是	否
AWS Performance Insights (績效詳情)	是	是	否	否	是	否
Amazon Personalize	是	是	否	否	是	否
Amazon Pinpoint	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Pinpoint 電子郵件服務	是	是	否	是	是	否
Amazon Pinpoint 簡訊和語音服務	是	否	否	否	是	否
Amazon Pinpoint SMS and Voice Service V2	是	是	否	是	是	否
Amazon Polly	是	是	否	否	是	否
AWS 價格表	是	否	否	否	是	否
AWS 私人 5G	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Private CA Connector for Active Directory	是	是	否	是	是	否
AWS Private CA 連接器應用於 SCEP	是	是	否	是	是	否
AWS Private Certificate Authority (AWS Private CA)	是	是	是	是	是	否
AWS Proton	是	是	否	是	是	是
AWS 採購單主控台	是	是	否	是	是	否
Amazon Q Business	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Q 商務 Q 應用程序	是	是	否	否	是	否
Amazon Q 開發者	是	否	否	否	是	是
Amazon Q in Connect	是	是	否	是	是	否
Amazon Quantum Ledger Database (Amazon QLDB)	是	是	否	是	是	否
Amazon QuickSight	是	是	否	是	是	否
Amazon RDS Data API	是	是	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon RDS IAM 身分驗證	是	是	否	否	是	否
AWS 資源回收筒	是	是	否	是	是	否
Amazon Redshift	是	是	否	是	是	是
Amazon Redshift 資料 API	是	是	否	否	是	否
Amazon Redshift Serverless	是	是	是	是	是	否
Amazon Rekognition	是	是	部分 (資訊)	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Relational Database Service (Amazon RDS) (資訊)	是	是	否	是	是	是
AWS re:Post 私有	是	是	否	是	是	是
AWS Resilience Hub	是	是	否	是	是	否
AWS Resource Access Manager (AWS RAM)	是	是	否	是	是	是
AWS 資源總管	是	是	否	是	是	是
AWS Resource Groups	是	是	否	是	部分 (資訊)	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Resource Groups Tagging API	是	否	否	否	是	否
Amazon RHEL 知識庫入口網站	是	否	否	否	是	否
AWS RoboMaker	是	是	否	是	是	是
Amazon Route 53	是	是	否	否	是	否
Amazon Route 53 Application Recovery Controller - 區域移位	是	是	否	否	是	否
Amazon Route 53 網域	是	否	否	否	否	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon 路線 53 設定檔	是	是	否	是	是	否
Amazon Route 53 Recovery Cluster	是	是	否	否	是	否
Amazon Route 53 Recovery Control Config	是	是	否	是	是	否
Amazon Route 53 Recovery Readiness	是	是	否	是	是	是
Amazon Route 53 Resolver	是	是	否	是	是	是
Amazon S3 Express	是	是	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon S3 Glacier	是	是	是	是	是	否
Amazon SageMaker	是	是	否	是	是	部分 (資訊)
Amazon SageMaker 地理空間	是	是	否	是	是	否
Amazon SageMaker Ground Truth 合成	是	否	否	否	是	否
AWS Savings Plans	是	是	否	是	是	否
AWS Secrets Manager	是	是	是	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Security Hub	是	是	否	是	是	是
Amazon Security Lake	是	是	否	否	是	是
AWS Security Token Service (AWS STS)	是	部分 (資訊)	否	是	部分 (資訊)	否
AWS Serverless Application Repository	是	是	是	否	是	否
AWS Service Catalog	是	是	否	是	是	是
Service Quotas	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Shield	是	是	否	是	是	是
AWS Signer	是	是	是	是	是	否
AWS 登入	是	否	否	否	是	否
Amazon SimpleDB	是	是	否	否	是	否
Amazon 簡單的電子郵件服務-郵件	是	是	否	是	是	是
Amazon Simple Email Service (Amazon SES) v2	是	部分 (資訊)	是	是	部分 (資訊)	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Simple Notification Service (Amazon SNS)	是	是	是	是	是	否
Amazon Simple Queue Service (Amazon SQS)	是	是	是	部分	是	否
Amazon Simple Storage Service (Amazon S3)	是	是	是	部分 (資訊)	是	部分 (資訊)
Amazon Simple Storage Service (Amazon S3) 物件 Lambda	是	是	否	否	是	否
Amazon Simple Storage Service (Amazon S3) AWS Outposts	是	是	是	否	是	是
Amazon Simple Workflow Service (Amazon SWF)	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS SimSpace織布工	是	是	否	是	是	否
AWS Site-to-Site VPN	是	是	否	否	是	是
AWS Snowball	是	否	否	否	是	否
AWS Snowball Edge	是	否	否	否	是	否
AWS Snow Device Management	是	是	否	是	是	否
AWS SQL Workbench	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Step Functions	是	是	否	是	是	否
AWS Storage Gateway	是	是	否	是	是	否
AWS Supply Chain	是	是	否	是	是	否
AWS Support App in Slack	是	否	否	否	是	否
AWS Support	是	否	否	否	是	是
AWS Support 计划	是	否	否	否	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Support 建議	是	否	否	否	是	否
AWS 永續性	是	否	否	否	是	否
AWS Systems Manager	是	是	否	是	是	是
AWS Systems Manager 適用於 SAP	是	是	否	是	是	否
AWS Systems Manager 圖形介面 Connect	是	否	否	否	是	否
AWS Systems Manager Incident Manager	是	是	是	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Systems Manager Incident Manager 聯絡案例	是	是	是	否	是	否
標籤編輯器	是	否	否	否	是	否
AWS 稅金設定	是	否	否	否	是	否
AWS 電信網絡生成器	是	是	否	是	是	否
Amazon Textract	是	否	否	否	是	否
Amazon Timestream	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon Timestream	是	是	否	是	是	是
AWS 類型 API（用於 Reachability Analyzer）	是	否	否	否	否	否
Amazon Transcribe	是	是	否	是	是	否
AWS Transfer Family	是	是	否	是	是	否
Amazon Translate	是	是	否	是	是	否
AWS Trusted Advisor	部分（資訊）	是	否	否	部分	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS 使用者通知	是	是	否	是	是	是
AWS 用戶通知聯系人	是	是	否	是	是	否
AWS 使用者訂閱	是	否	否	否	是	否
AWS Verified Access	是	否	否	否	是	否
Amazon Verified Permissions	是	是	否	否	是	否
Amazon Virtual Private Cloud (Amazon VPC)	是	部分 (資訊)	部分 (資訊)	是	是	部分 (資訊)

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon VPC Lattice	是	是	否	是	是	否
Amazon VPC Lattice Services	是	是	否	否	是	否
AWS WAF	是	是	否	是	是	是
AWS WAF Classic	是	是	否	是	是	是
AWS WAF Regional	是	是	否	是	是	是
AWS Well-Architected Tool	是	是	否	是	是	否

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
AWS Wickr	是	是	否	是	是	否
Amazon WorkDocs	是	否	否	否	是	否
Amazon WorkMail	是	是	否	是	是	是
Amazon WorkMail 消息流	是	是	否	否	是	否
Amazon WorkSpaces	是	是	否	是	是	否
Amazon 瀏 WorkSpaces 覽器	是	是	否	是	是	是

服務	動作	資源層級許可	資源型政策	ABAC	暫時性憑證	服務連結角色
Amazon WorkSpaces 瘦客戶端	是	是	否	是	是	否
AWS X-Ray	是	部分 (資訊)	否	部分 (資訊)	是	否

其他資訊

Amazon CloudFront

CloudFront 沒有服務連結的角色，但 Lambda @Edge 有。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的 Lambda @Edge 的服務連結角色。

AWS CloudTrail

CloudTrail 僅在用於與外部事件來源的 CloudTrail Lake 整合的 CloudTrail 管道上支援以資源為基礎的 AWS政策。

CloudTrail 支援 CloudTrail Lake 事件資料存放區和通道的以標籤為基礎的存取控制。 CloudTrail 不支持跟踪基於標籤的訪問控制。

Amazon CloudWatch

CloudWatch 無法使用建立服務連結角色 AWS Management Console，且僅支援警示動作功能。

AWS CodeBuild

CodeBuild 支援跨帳號資源共用使用 AWS RAM.

CodeBuild 支持 ABAC 基於項目的行動。

AWS Config

AWS Config 支援多帳戶多區域資料彙總和規則的資源層級權限。 AWS Config 如需支援的資源清單，請參閱 AWS Config API 指南的多帳戶多區域資料彙總與 AWS Config 規則章節。

AWS Database Migration Service

您可以建立和修改附 AWS KMS 加至您建立之加密金鑰的策略，以加密移轉至支援目標端點的資料。支援的目標端點包含 Amazon Redshift 和 Amazon S3。如需詳細資訊，請參閱使用者指南中的建立和使用 AWS KMS 金 AWS KMS 鑰加密 Amazon Redshift 目標資料和建立加密 Amazon S3 目標物AWS Database Migration Service 件的金鑰。

Amazon Elastic Compute Cloud

Amazon EC2 服務連結角色只能用於下列功能：Spot 執行個體請求、Spot 機群請求、Amazon EC2 Fleet 以及快速啟動 Windows 執行個體。

Amazon Elastic Container Service

僅部分 Amazon ECS 動作支援資源層級的許可。

AWS Elemental MediaPackage

MediaPackage 支援服務連結角色，用於將客戶存取記錄發佈至其他 API 動作， CloudWatch 但不支援。

AWS Identity and Access Management

僅支援一種稱為角色信任政策，且已連接到 IAM 角色的以資源為基礎的政策。如需詳細資訊，請參閱向使用者授予切換角色的許可。

IAM 支援大多數 IAM 資源的以標籤為基礎的存取控制。如需詳細資訊，請參閱 標記 IAM 資源。

只有 IAM 的部分 API 動作可使用暫時憑證來呼叫。如需詳細資訊，請參閱比較 API 選項。

AWS IoT

連線到的裝置 AWS IoT 會使用 X.509 憑證或使用 Amazon 認可身分進行驗證。您可以將 AWS IoT 政策附加到 X.509 憑證或 Amazon Cognito 身分識別，以控制裝置授權執行的動作。如需詳細資訊，請參閱AWS IoT 開發人員指南中的 AWS IoT的安全與身分。

AWS Lambda

Lambda 支援屬性型存取控制 (ABAC)，適用於使用 Lambda 函數作為必要資源的 API 動作。不支援圖層、事件來源映射和程式碼簽章組態資源。

Lambda 沒有服務連結角色，但 Lambda@Edge 則有。如需詳細資訊，請參閱 Amazon CloudFront 開發人員指南中的 Lambda @Edge 的服務連結角色。

Amazon Lightsail

Lightsail 部分支援資源層級許可和 ABAC。如需詳細資訊，請參閱適用於 Amazon Lightsail 的動作、資源和條件金鑰。

Amazon Managed Streaming for Apache Kafka (MSK)

您可以將叢集政策附加到已設定為多 V PC 連線的 Amazon MSK 叢集。

AWS Network Manager

AWS 雲端 WAN 也支援服務連結角色。如需詳細資訊，請參閱 Amazon VPC AWS 雲端廣域網路指南中的 AWS 雲端 WAN 服務連結角色。

Amazon Relational Database Service

Amazon Aurora 為全受管關聯式資料庫引擎，可與 MySQL 和 PostgreSQL 相容。透過 Amazon RDS 設定新的資料庫伺服器時，您可以選擇 Aurora MySQL 或 Aurora PostgreSQL 做為資料庫引擎選項。如需詳細資訊，請參閱《Amazon Aurora 使用者指南》中的 適用於 Amazon Aurora 的 Identity and Access Management。

Amazon Rekognition

僅在複製 Amazon Rekognition 自訂標籤模型時才支援以資源為基礎的政策。

AWS Resource Groups

使用者可以使用允許資源群組操作的政策來擔任角色。

Amazon SageMaker

服務連結的角色目前可供 SageMaker Studio 和 SageMaker 訓練工作使用。

AWS Security Token Service

AWS STS 沒有「資源」，但允許以類似的方式限制訪問用戶。如需詳細資訊，請參閱拒絕依名稱存取暫時安全憑證。

只有一些 API 操作用於 AWS STS 支持使用臨時憑據調用。如需詳細資訊，請參閱比較 API 選項。

Amazon Simple Email Service

您可以在參考與傳送電子郵件相關動作的政策陳述式中只使用資源層級的許可，例如 `ses:SendEmail` 或 `ses:SendRawEmail`。對於參考任何其他動作的政策陳述式，資源元素只能包含 *。

只有 Amazon SES API 支援暫時安全憑證。Amazon SES SMTP 界面不支援自臨時安全憑證所衍生的 SMTP 憑證。

Amazon Simple Storage Service

Amazon S3 僅對物件資源支援以標籤為基礎的授權。

Amazon S3 支援 Amazon S3 Storage Lens 的服務連結角色。

AWS Trusted Advisor

API 存取權 Trusted Advisor 是透過 AWS Support API 進行的，且由 AWS Support IAM 政策控制。

Amazon Virtual Private Cloud

在 IAM 使用者政策中，您無法限制特定 Amazon VPC 端點的許可。包含 `Action` 或 `ec2:*VpcEndpoint*` API 動作的任何 `ec2:DescribePrefixLists` 元素必須指定「`"Resource": "*"`」。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的 VPC 端點和 VPC 端點服務的身分與存取管理。

Amazon VPC 支援將單一資源政策連接至 VPC 端點，以限制可透過該端點存取的項目。如需有關使用資源型政策控制從特定 Amazon VPC 端點存取資源的詳細資訊，請參閱AWS PrivateLink 指南中的使用端點政策控制對服務的存取。

Amazon VPC 沒有服務連結角色，但 AWS Transit Gateway 有。如需詳細資訊，請參閱 Amazon VPC AWS Transit Gateway 指南中的傳輸閘道使用服務連結角色。

AWS X-Ray

X-Ray 未針對所有動作支援資源層級的許可。

X-Ray 支援以標籤為基礎的群組和抽樣規則的存取控制。

簽署 AWS API 要求

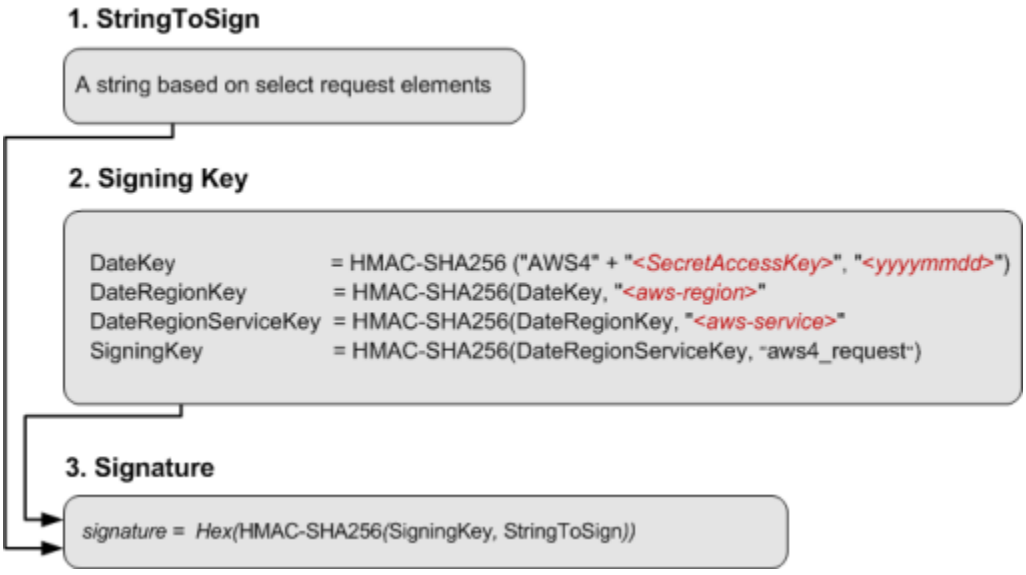
> **Important**
>
> 如果您使用 AWS SDK (請參閱範例程式碼和程式庫) 或 AWS 命令列 (CLI) 工具將 API 要求傳送至 AWS，您可以略過此區段，因為 SDK 和 CLI 用戶端會使用您提供的存取金鑰來驗證您的要求。除非有充分的理由不這樣做，否則建議您始終使用 SDK 或 CLI。
>
> 在支援多個簽章版本的區域中，手動簽署要求意味著您必須指定要使用的簽章版本。當您向多區域存取點提供請求時，SDK 和 CLI 會自動切換為使用第 4A 版簽署程序，而不需其他組態。

您在請求中傳送的驗證資訊必須包含簽章。要計算簽章，首先串連 select 請求元素以形成一個字串 (稱為要簽署的字串)。然後，您可以使用簽署金鑰來計算要簽署的字串的雜湊訊息驗證碼 (HMAC)。

在「 AWS 簽名版本 4」中，您不會使用私密存取金鑰來簽署要求。相反，首先使用私密存取金鑰來衍生簽署金鑰。衍生的簽署金鑰根據日期、服務和區域而特定。如需有關如何在不同的程式設計語言中衍生簽署金鑰的詳細資訊，請參閱 請求簽章範例。

簽名版本 4 是 AWS 簽名協議。 AWS 也支援擴充功能「簽名版本 4A」，支援多區域 API 要求的簽章。如需詳細資訊，請參閱上 GitHub的 sigv4 a-signing-examples 專案。

下圖說明計算簽章的一般程序。



- 要簽署的字串取決於請求類型。例如，當您使用 HTTP 授權標頭或查詢參數進行驗證時，請使用不同的請求元素組合來建立要簽署的字串。對於 HTTP POST 請求，請求中的 POST 政策是您簽署的字串。
- 對於簽署金鑰，圖表顯示了一系列的計算，然後將每一步驟的結果饋送到下一個步驟。最後的步驟是簽署金鑰。
- 當 AWS 服務收到已驗證的要求時，會使用要求中包含的驗證資訊重新建立簽章。如果簽章相符，則服務會處理請求。否則，它會拒絕請求。

目錄

簽署請求的時機

當您撰寫傳送 API 要求的自訂程式碼時 AWS，您必須包含簽署要求的程式碼。您可能會因為以下原因編寫自訂程式碼：

- 您使用的程式設計語言沒有 AWS 軟體開發套件。
- 您需要完全控制要求傳送至的方式 AWS。

為什麼要簽署請求

簽署程序有助於以下列方式保護請求的安全：

- 驗證請求者的身分

  經過驗證的請求需要您使用存取金鑰 (存取金鑰 ID、私密存取金鑰) 建立的簽章。如果您使用暫時安全憑證，則簽章計算還需要安全字符。如需詳細資訊，請參閱 AWS 安全憑證程式設計存取權。
- 保護傳輸中的資料

為了防止傳送中的請求遭到竄改，有些請求元素可用來計算請求的雜湊 (摘要)，而產生的雜湊值會包含在請求中。當 AWS 服務 收到請求時，它會使用相同的信息來計算散列並將其與請求中的哈希值進行匹配。如果值不匹配，則 AWS 拒絕請求。

- 抵禦潛在重播攻擊的侵害

  在大多數情況下，請求必須在請求中的時間戳記的五分鐘 AWS 內到達。否則， AWS 拒絕請求。

AWS API 請求簽名的元素

> **Important**
>
> 除非您使用 AWS SDK 或 CLI，否則您必須撰寫程式碼來計算在要求中提供驗證資訊的簽章。簽名版本 4 中的 AWS 簽名計算可能是一項複雜的任務，我們建議您盡可能使用 AWS SDK 或 CLI。

凡是使用 Signature Version 4 簽署的每個 HTTP/HTTPS 請求都必須包含下列元素。

元素

- 端點規格
- 動作
- 動作參數
- 日期
- 身分驗證資訊

端點規格

指定您要向其傳送請求的端點的 DNS 名稱。此名稱通常包含服務代碼和區域。例如，`us-east-1` 區域中 Amazon DynamoDB 的端點為 `dynamodb.us-east-1.amazonaws.com`。

若為 HTTP/1.1 請求，必須包含 `Host` 標頭。若為 HTTP/2 請求，則可包含 `:authority` 標頭或 `Host` 標頭。為符合 HTTP/2 規格，應僅使用 `:authority` 標頭。並非所有服務都支援 HTTP/2 請求。

如需每個服務支援的端點，請參閱 AWS 一般參考 中的服務端點和配額。

動作

指定服務的 API 動作。例如，DynamoDB `CreateTable` 動作或 Amazon EC2 `DescribeInstances` 動作。

如需每個服務支援的動作，請參閱服務授權參考。

動作參數

指定請求中指定的動作參數。每個 AWS API 動作都有一組必要和可選參數。API 版本通常是必要參數。

如需 API 動作支援的參數，請參閱服務的《API 參考》。

日期

指定請求的日期和時間。請求中附上日期和時間有助於防止第三方攔截您的請求再於稍後重新提交。您在憑證範圍中指定的日期必須與請求日期相符。

此時間戳記必須為 UTC 時間，而且必須使用下列 ISO 8601 格式：YYYYMMDDTHHMMSSZ。例如 `20220830T123600Z`。不包含時間戳記的毫秒數。

您可以使用 `date` 標頭或 `x-amz-date` 標頭，或包含 `x-amz-date` 作為查詢參數。如果找不到 `x-amz-date` 標頭，那麼我們會尋找 `date` 標頭。

身分驗證資訊

您傳送的每個要求都必須包含下列資訊。 AWS 使用此信息來確保請求的有效性和真實性。

- 演算法 – 搭配使用 `AWS 4-HMAC-SHA256` 與 `HMAC-SHA256` 雜湊演算法來指定 Signature Version 4。
- 憑證 – 由您的存取金鑰 ID、YYYYMMDD 格式的日期、區域代碼、服務代碼和 `aws4_request` 終止字串所組成的字串，以斜線 (/) 分隔。區域代碼、服務代碼和終止字串必須使用小寫字元。

  ```
  AKIAIOSFODNN7EXAMPLE/YYYYMMDD/region/service/aws4_request
  ```

- 已簽署的標頭 – 要包含在簽章中的 HTTP 標頭，以分號 (;) 分隔。例如 `host;x-amz-date`。
- 簽章 – 表示已計算出之簽章的十六進位編碼字串。您必須使用 `Algorithm` 參數所指定的演算法計算簽章。

身分驗證方法

> **Important**
>
> 除非您使用 AWS SDK 或 CLI，否則您必須撰寫程式碼來計算在要求中提供驗證資訊的簽章。簽名版本 4 中的 AWS 簽名計算可能是一項複雜的任務，我們建議您盡可能使用 AWS SDK 或 CLI。

您可以使用下列其中一種方法來表示身分驗證資訊。

HTTP 授權標頭

HTTP `Authorization` 標頭是驗證請求的最常見方法。所有 REST API 操作 (使用 `POST` 請求的以瀏覽器為基礎的上傳除外) 需要此標頭。如需有關授權標頭值以及如何計算簽名和相關選項的詳細資訊，請參閱 Amazon S3 API 參考中的驗證請求：使用授權標頭 (AWS 簽名版本 4)。

以下是 `Authorization` 標頭值的範例。為了便於閱讀，向此範例新增了分行符號。在程式碼中，標頭必須是一個連續字串。演算法與憑證之間沒有逗號，但其他元素必須以逗號分隔。

```
Authorization: AWS 4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

下表描述上述範例中授權標頭值的各種元件：

元件	描述
授權	用於計算簽章的演算法。當您使用 AWS 簽章版本 4 進行驗證時，必須提供此值。字串會指定 AWS 簽章版本 4 (`AWS 4`) 和簽署演算法 ()。`HMAC-SHA256`
Credential	您的存取金鑰 ID 和範圍資訊，包括用於計算簽章的日期、區域和服務。 此字串的格式如下：

元件	描述
	<your-access-key-id>/<date>/<aws-region>/<aws-service>/aws4_request 其中：使用 YYYYMMDD 格式指定 <date> 值。將請求傳送至 Amazon S3 時 <aws-service> 值為 s3。
SignedHeaders	用於計算 Signature 的請求標頭清單 (以分號分隔)。此清單僅包含標頭名稱，且標頭名稱必須為小寫。例如：主機；範圍；x-amz-date
簽章	256 位元簽章以 64 個小寫十六進位字元表示。例如：fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024 請注意，簽章計算因您選擇傳輸承載的選項而異。

查詢字串參數

您可以使用查詢字串以完全在 URL 中表達請求。在此情況下，您可以使用查詢參數來提供請求資訊，包括驗證資訊。由於請求簽章是 URL 的一部分，因此此類型 URL 通常稱為預先簽章的 URL。您可以使用預先簽章的 URL 在 HTML 中嵌入可點選連結，有效期最長可達七天。如需詳細資訊，請參閱 Amazon S3 API 參考[中的驗證請求：使用查詢參數 (AWS 簽章版本 4)]。

以下是預先簽章的 URL 範例。為了便於閱讀，向此範例新增了分行符號：

```
https://s3.amazonaws.com/examplebucket/test.txt ?
X-Amz-Algorithm=AWS4-HMAC-SHA256 &
X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request &
X-Amz-Date=20130721T201207Z &
X-Amz-Expires=86400 &
X-Amz-SignedHeaders=host &X-Amz-Signature=<signature-value>
```

> **Note**
>
> URL 中的 X-Amz-Credential 值僅為了便於閱讀而顯示 "/" 字元。實際上，應將其編碼為 %2F。例如：
> &X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request

下表描述 URL 中提供驗證資訊的查詢參數。

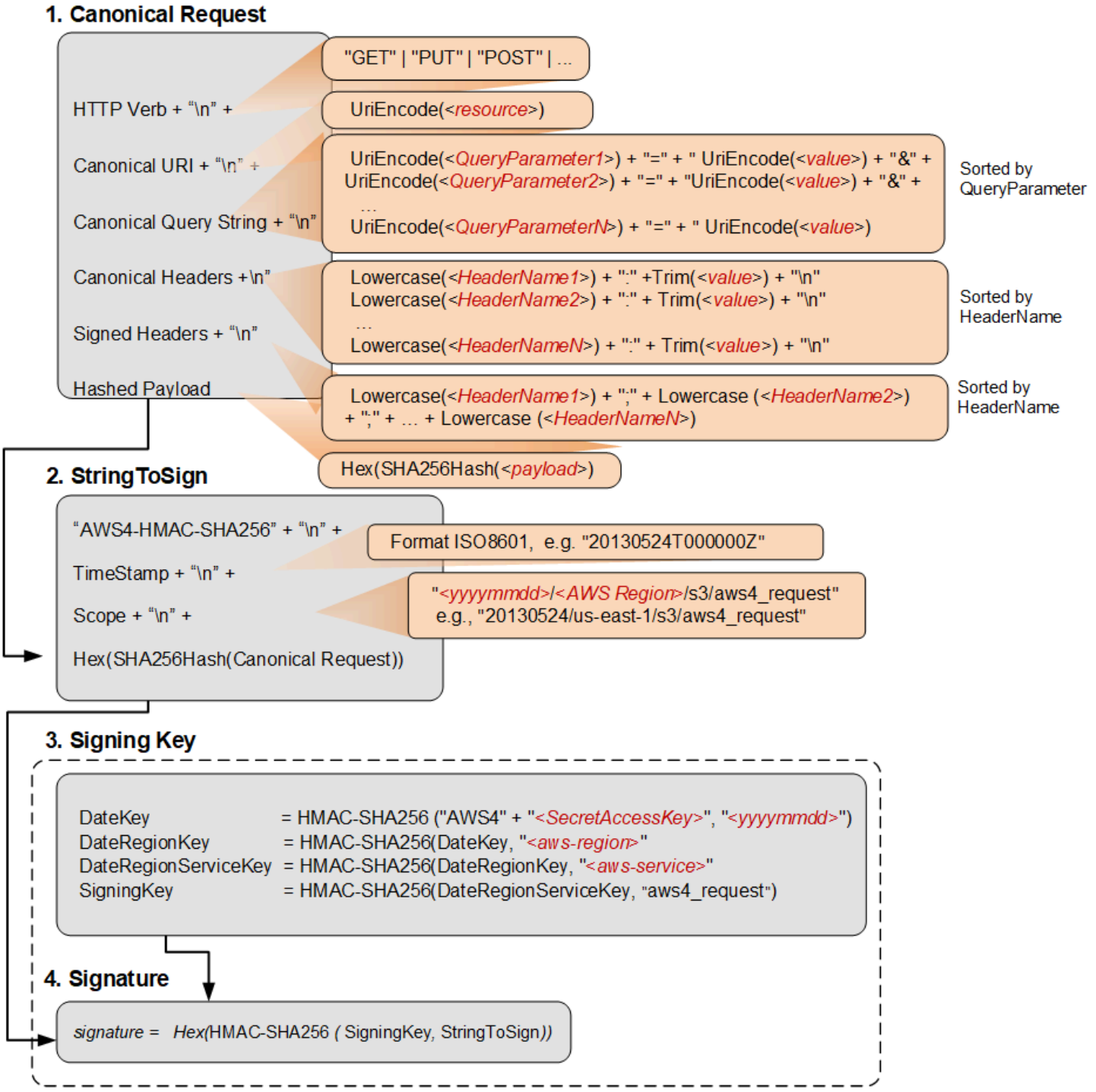
查詢字串參數名稱	描述
X-Amz-Algorithm	識別 AWS 簽名的版本以及您用來計算簽名的演算法。對於簽 AWS 名版本 4，您可以將此參數值設定為。AWS 4-HMAC-SHA256 此字串識別 AWS 第 4 版簽署程序 (AWS 4) 和 HMAC-SHA256 演算法 (HMAC-SHA256)。
X-Amz-Credential	除了您的存取金鑰 ID 之外，此參數也會提供簽章有效的範圍 (AWS 地區和服務)。此值必須與您在簽章計算中使用的範圍相符，這將在下一節中討論。 此參數值的一般格式如下： <your-access-key-id>/<date>/<AWS Region>/<AWS-service>/aws4_request 例如：AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request 如需地 AWS 區字串的清單，請參閱AWS 一般參考中的地區端點。
X-Amz-Date	日期和時間格式必須遵循 ISO 8601 標準，且必須使用 yyyyMMddTHHmmssZ 格式進行格式化。例如，如果日期和時間是 "08/01/2016 15:32:41.982-700"，則必須先將其轉換為 UTC

查詢字串參數名稱	描述
	(國際標準時間)，然後以 "20160801T223241Z" 形式提交。
X-Amz-Expires	提供所產生預先簽章的 URL 有效的時段 (以秒為單位)。例如，86400 (24 小時)。此值為整數。您可以設定的最小值為 1，最大值為 604,800 (七天)。預先簽章的 URL 的有效期最長為七天，因為您在簽章計算中使用的簽署金鑰有效期最長為七天。
X-阿姆斯特丹 SignedHeaders	列出您用來計算簽章的標頭。簽章計算中需要下列標頭： • HTTP 主機標頭。 • 您計劃新增至請求的任何 x-amz-* 標頭。 為了提高安全性，您應簽署計劃包含在請求中的所有請求標頭。
X-Amz-Signature	提供簽章以驗證您的請求。此簽章必須與服務計算的簽章相符；否則，服務會拒絕請求。例如：`733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7` 簽章計算將在下一節中描述。
X-Amz-Security-Token	選用的憑證參數 (如果使用來自 STS 服務的憑證)。

建立已簽署的 AWS API 要求

> **Important**
>
> 如果您使用 AWS SDK (請參閱範例程式碼和程式庫) 或 AWS 命令列 (CLI) 工具將 API 要求傳送至 AWS，您可以略過此區段，因為 SDK 和 CLI 用戶端會使用您提供的存取金鑰來驗證您的要求。除非有充分的理由不這樣做，否則建議您始終使用 SDK 或 CLI。
>
> 在支援多個簽章版本的區域中，手動簽署要求意味著您必須指定要使用的簽章版本。當您向多區域存取點提供請求時，SDK 和 CLI 會自動切換為使用第 4A 版簽署程序，而不需其他組態。

以下是建立已簽署請求的程序概觀。若要計算簽章，首先需要一個要簽名的字串。然後，您可以使用簽署金鑰計算要簽署的字串的 HMAC-SHA256 雜湊。下圖說明此程序，包括您為簽署建立的字串的各種元件。



下表描述圖表中顯示的函數。您需要為這些函數實作程式碼。如需詳細資訊，請參閱 AWS SDK 中的程式碼範例。

函式	描述
`Lowercase()`	將字串轉換成小寫。
`Hex()`	小寫基數 16 編碼。
`SHA256Hash()`	安全雜湊演算法 (SHA) 加密雜湊函數。
`HMAC-SHA256()`	使用 SHA256 演算法搭配提供的簽署金鑰來計算 HMAC。這是最後的簽章。
`Trim()`	移除任何前置和結尾空格。
`UriEncode()`	URI 編碼每個字節。 UriEncode ( ) 必須執行以下規則： • URI 對除未預留字元之外的每個位元組進行編碼：'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', and '~'。 • 空格字元是預留字元，必須編碼為 "%20" (而不是 "+")。 • 每個 URI 編碼的位元組都是由 '%' 和位元組的兩位數十六進位值組成。 • 十六進位值中的字母必須為大寫，例如 "%1A"。 • 對所有位置的正斜線字元 '/' 進行編碼 (物件索引鍵名稱除外)。例如，如果物件索引鍵名稱為 `photos/Jan/sample.jpg` ，則不會對索引鍵名稱中的正斜線進行編碼。 **Important** 您的開發平台提供的標準 UriEncode 功能可能因為基礎 RFC 中的實作和相關模糊性的差異而無法運作。我們建議您編寫自己的自定義 UriEncode 函數，以確保您的編碼能夠正常工作。

函式	描述
	若要查看 Java 中 UriEncode 函數的範例，請參閱 GitHub 網站上的 Java 公用程式。

> **Note**
>
> 簽署請求時，您可以使用 AWS 簽名版本 4 或 AWS 簽名版本 4A。兩者之間的主要差異取決於簽章的計算方式。使用「 AWS 簽名版本 4A」時，簽章不包含區域特定資訊，並使用演算法計算。`AWS 4-ECDSA-P256-SHA256`

暫時安全憑證

您可以使用 AWS Security Token Service (AWS STS) 提供的臨時安全登入資料，而不是使用長期認證來簽署要求。

當您使用臨時安全憑證時，必須將 `X-Amz-Security-Token` 新增至授權標頭或查詢字串，以保留工作階段權杖。某些服務會要求您將 `X-Amz-Security-Token` 新增至正式請求。其他服務只需要您在計算簽章之後於結尾新增 `X-Amz-Security-Token`。請查看每個文檔以獲取 AWS 服務 詳細信息。

簽署步驟摘要

步驟 1：建立正式請求

將您的請求內容 (主機、動作、標頭等) 編排為標準正式格式。正式請求就是其中一個用於建立登入字串的輸入。如需詳細資訊，請參閱 AWS API 請求簽名的元素。

步驟 2：建立正式請求雜湊

透過在要求日期、區域和服務執行連續的金鑰加上金鑰的金鑰做為初始雜湊作業的 AWS 金鑰，以衍生簽署金鑰。

步驟 3：建立要簽署的字串

使用正式請求和額外資訊，例如演算法、請求日期、登入資料範圍及正式請求的摘要 (雜湊)，建立登入字串。

步驟 4：計算簽章

您衍生簽署金鑰之後，可在登入字串上執行金鑰式雜湊操作，即可計算簽章。使用衍生的金鑰做為此操作的雜湊金鑰。

步驟 5：將簽章新增至請求

在您計算簽章、將簽章加到 HTTP 標頭，或加到請求的查詢字串之後。

步驟 1：建立正式請求

透過串連以下字串 (以換行符號字元分隔) 建立正式請求。這有助於確保您計算的簽名和計 AWS 算的簽名可以相符。

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

- *HTTPMethod* – HTTP 方法，例如 GET、PUT、HEAD 和 DELETE。
- *CanonicalUri*— 絕對路徑元件 URI 的 URI 編碼版本，從網域名稱後面的「/」開始，直到字串結尾或問號字元 ('?') 如果你有查詢字符串參數。如果絕對路徑空白，請使用斜線字元 (/)。下列範例中的 URI /examplebucket/myphoto.jpg 是絕對路徑，您不會在絕對路徑中編碼 "/"：

  ```
  http://s3.amazonaws.com/examplebucket/myphoto.jpg
  ```

- *CanonicalQueryString*— URI 編碼的查詢字串參數。您可以個別對每個名稱和值進行 URI 編碼。您還必須依索引鍵名稱的字母順序來排序正式查詢字串中的參數。編碼後進行排序。下列 URI 範例中的查詢字串為：

  ```
  http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=2
  ```

  正式查詢字串如下所示 (為了便於閱讀，向此範例新增了分行符號)：

  ```
  UriEncode("marker")+"="+UriEncode("someMarker")+"&"+
  UriEncode("max-keys")+"="+UriEncode("20") + "&" +
  UriEncode("prefix")+"="+UriEncode("somePrefix")
  ```

  請求以子資源為目標時，對應的查詢參數值將是空字串 ("")。例如，下列 URI 會識別 examplebucket 儲存貯體上的 ACL 子資源：

```
http://s3.amazonaws.com/examplebucket?acl
```

CanonicalQueryString 在這種情況下，如下所示：

```
UriEncode("acl") + "=" + ""
```

如果 URI 不包含 '?'，則請求中沒有查詢字串，並且您將正式查詢字串設定為空字串 ("")。您仍然需要包含 "\n"。

- *CanonicalHeaders*— 請求標頭及其值的清單。個別標頭名稱和值對以新行字元 ("\n") 分隔。以下是 canonicalheader 的範例：

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

CanonicalHeaders 列表必須包括以下內容：

- HTTP host 標頭。
- 如果標Content-Type頭存在於請求中，則必須將其添加到*CanonicalHeaders*列表中。
- 還必須新增您計劃包含在請求中的任何 x-amz-* 標頭。例如，如果您使用的是暫時安全憑證，需要將 x-amz-security-token 包含在您的請求中。您必須在的清單中新增此標頭*CanonicalHeaders*。

> **Note**
>
> Amazon S3 AWS 請求需要x-amz-content-sha256標頭。其提供請求承載的雜湊。如果沒有承載，您必須提供空字串的雜湊。

每個標頭名稱必須：

- 使用小寫字元。
- 依字母順序顯示。
- 後接冒號 (:)。

對於值，您必須：

- 修剪任何前置或結尾空格。
- 將連續空格轉換為單一空格。
- 使用逗號分隔多值標頭的值。
- 簽章中必須包含主機標頭 (HTTP/1.1) 或 :authority 標頭 (HTTP/2)，以及簽章中的任何 `x-amz-*` 標頭。可以選擇性地在簽章中包含其他標準標頭，例如 content-type。

本範例中使用的 `Lowercase()` 和 `Trim()` 函數在上一節中進行了描述。

以下是 `CanonicalHeaders` 字串範例。標頭名稱為小寫且已排序。

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

> **Note**
>
> 對於計算授權簽章而言，僅需要主機和任何 `x-amz-*` 標頭；但是，為了防止資料竄改，您應考慮在簽章計算中包含所有標頭。

- *`SignedHeaders`*— 按字母順序排序、以分號分隔的小寫要求標頭名稱清單。清單中的請求標頭與您在 `CanonicalHeaders` 字串中包含的標頭相同。例如，對於上一個範例，的值*`SignedHeaders`*將如下所示：

```
host;x-amz-content-sha256;x-amz-date
```

- *`HashedPayload`*— 使用 HTTP 要求主體中的有效負載建立的字串，做為雜湊函數的輸入。此字串使用小寫十六進位字元。

```
Hex(SHA256Hash(<payload>)
```

如果請求中沒有承載，則計算空字串的雜湊，如下所示：

```
Hex(SHA256Hash(""))
```

雜湊會傳回下列值：

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

例如，使用 PUT 請求上傳物件時，請在主體中提供物件資料。使用 GET 請求擷取物件時，請計算空字串雜湊。

步驟 2：建立正式請求雜湊

使用與您用來建立承載雜湊相同的演算法，建立正式請求的雜湊 (摘要)。正式請求的雜湊是小寫十六進位字元字串。

步驟 3：建立登入字串

透過串連以下字串 (以換行符號字元分隔) 建立字串。請勿以換行符號字元結束此字串。

```
Algorithm \n
RequestDateTime \n
CredentialScope  \n
HashedCanonicalRequest
```

- *Algorithm* - 用來建立正式請求雜湊的演算法。對於 SHA-256 而言，演算法為 AWS4-HMAC-SHA256。
- *RequestDateTime*— 認證範圍中使用的日期和時間。此值是 ISO 8601 格式的目前 UTC 時間 (例如，20130524T000000Z)。
- *CredentialScope*— 認證範圍。這會將產生的簽章限制到指定的區域和服務。該字串具有以下格式：*YYYYMMDD*/*region*/*service*/aws4_request。
- *HashedCanonicalRequest*— 規範要求的雜湊值。此值是在步驟 2 中計算的。

以下是登入字串範例。

```
"AWS4-HMAC-SHA256" + "\n" +
timeStampISO8601Format + "\n" +
<Scope> + "\n" +
Hex(SHA256Hash(<CanonicalRequest>))
```

步驟 4：計算簽章

在 S AWS ignature Version 版本 4 中，您不需要使用 AWS 存取金鑰來簽署請求，而是建立範圍為特定區域和服務的簽署金鑰，做為要新增至要求的驗證資訊。

```
DateKey = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

如需區域字串清單，請參閱《AWS 一般參考》中的區域端點。

對於每個步驟，使用所需金鑰和資料呼叫雜湊函數。對雜湊函數的每次呼叫結果都會變成雜湊函數的下一次呼叫輸入。

需要輸入

- 包含私密存取金鑰的字串 Key
- 字串 Date，其中包含憑證範圍中使用的日期，格式為 YYYYMMDD
- 包含區域代碼 (例如，us-east-1) 的字串 Region
- 包含服務代碼 (例如，ec2) 的字串 Service
- 在之前步驟中建立的登入字串。

若要計算簽章。

1. 串連 "AWS4" 和私密存取金鑰。呼叫雜湊函數，將串連的字串作為索引鍵，日期字串做為資料。

   ```
   kDate = hash("AWS4" + Key, Date)
   ```

2. 呼叫雜湊函數，將上一次呼叫的結果作為索引鍵，區域字串作為資料。

   ```
   kRegion = hash(kDate, Region)
   ```

3. 呼叫雜湊函數，將上一次呼叫的結果作為索引鍵，服務字串作為資料。

   ```
   kService = hash(kRegion, Service)
   ```

4. 呼叫雜湊函數，將上一次呼叫的結果作為索引鍵，"aws4_request" 作為資料。

```
kSigning = hash(kService, "aws4_request")
```

5. 呼叫雜湊函數，將上一次呼叫的結果作為索引鍵，登入字串作為資料。結果是作為二進位值的簽章。

```
signature = hash(kSigning, string-to-sign)
```

6. 將簽章的表示形式從二進位轉換為十六進位，使用小寫字元。

步驟 5：將簽章新增至請求

Example 範例：授權標頭

以下範例顯示針對 DescribeInstances 動作的 Authorization 標頭。為了便於閱讀，此範例使用換行符號進行格式化。在代碼中，這必須是一個連續字串。演算法和 Credential 之間沒有逗號。但是，必須使用逗號分隔其他元素。

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request,
SignedHeaders=host;x-amz-date,
Signature=calculated-signature
```

Example 範例：查詢字串中包含身分驗證參數的請求

以下範例顯示針對 DescribeInstances 動作的查詢，其中包含身分驗證資訊。為了便於閱讀，此範例使用換行符號進行格式化，而不是 URL 編碼。在程式碼中，查詢字串必須是 URL 編碼的連續字串。

```
https://ec2.amazonaws.com/?
Action=DescribeInstances&
Version=2016-11-15&
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request&
X-Amz-Date=20220830T123600Z&
X-Amz-SignedHeaders=host;x-amz-date&
X-Amz-Signature=calculated-signature
```

AWS 軟體開發套件中的原始程式碼

AWS SDK 包含 GitHub 用於簽署 AWS API 要求的原始程式碼。如需程式碼範例，請參閱 範例儲存庫中的 AWS 範例專案

- AWS SDK for .NET — AWS4Signer.cs
- AWS SDK for C++ — AWSAuthV4Signer. CPP
- AWS SDK for Go — v4.go
- AWS SDK for Java — BaseAws4Signer.java
- AWS SDK for JavaScript — v4.js
- AWS SDK for PHP — SignatureV4.php
- AWS SDK for Python (Boto) — signers.py
- AWS SDK for Ruby — 簽名者

請求簽章範例

下列 AWS 簽署要求範例說明如何在沒有 AWS SDK 或 AWS 命令列工具的情況下使用 Sigv4 簽署傳送的要求。

使用 HTTP POST 的基於瀏覽器的 Amazon S3 上傳

驗證請求：基於瀏覽器的上傳描述簽章和相關資訊，Amazon S3 在收到請求時會使用其來計算簽章。

範例：使用 HTTP POST (使用 AWS 簽名版本 4) 的瀏覽器式上傳提供更多資訊，其中包含 POST 原則範例以及可用來上傳檔案的表單。範例政策和虛擬憑證會顯示工作流程以及產生的簽章和政策雜湊。

VPC Lattice 驗證的請求

第 4 版簽署程序 (SigV4) 驗證的請求範例提供了 Python 和 Java 範例，顯示如何在使用和不使用自訂攔截器的情況下執行請求簽署。

搭配使用第 4 版簽署程序與 Amazon Translate

搭配使用第 4 版簽署程序與 Amazon Translate 說明了如何使用 Python 程式將身分驗證資訊新增至 Amazon Translate 請求。此範例會提出 POST 請求、建立 JSON 結構 (其中包含要在請求的內文 (承載) 中翻譯的文字)，並以授權標頭傳遞身分驗證資訊。

搭配使用第 4 版簽署程序與 Neptune

範例：搭配使用 Python 與第 4 版簽署程序來連接到 Neptune 說明如何使用 Python 向 Neptune 發出已簽署的請求。此範例包括使用存取金鑰或暫時憑證的變化情況。

簽署對 S3 Glacier 的 HTTP 請求

串流 API 的簽章計算範例逐步說明為上傳封存 (POST 封存) 建立簽章的詳細資訊，這是 S3 Glacier 中的兩個串流 API 之一。

向 Amazon SWF 發出 HTTP 請求

向 Amazon SWF 發出 HTTP 請求顯示向 Amazon SWF 發出 JSON 請求的標頭內容。

Amazon OpenSearch 服務中流式 API 的簽名計算

使用適用於 PHP 版本 3 的 AWS SDK 簽署 Amazon OpenSearch 服務搜尋請求包括如何將已簽署的 HTTP 請求傳送至 Amazon OpenSearch 服務的範例。

範例儲存庫中的 AWS 範例專案

下列範例專案會示範如何簽署要求，以便向使用常用語言 (例如 Python、Node.js、Java、C#、Go 和 Rust) 的 AWS 服務發出其餘 API 要求。

4a 版簽署程序專案

sigv4 簽名示例項目提供了如何使用 SigV4a 簽署請求的示例，以便使用常用語言（例如 Python，Node.js，Java，C＃，Go 和 Rust）向 AWS 服務 其餘 API 請求進行簽名。

sigv4 a-signing-examples 專案提供簽署多區域 API 請求的範例，例如 Amazon S3 中的多區域存取點。

發佈至 AWS IoT Core

AWS IoT Core 使用 HTTPS 協議發布到的 Python 代碼提供了有關如何 AWS IoT Core 使用 HTTPS 協議和 AWS Sigv4 身份驗證發布消息的指導。它有兩個引用實現-一個在 Python 中，另一個在 NodeJs.

AWS IoT Core 使用 HTTPS 協議發布到的 .NET 框架應用程序提供了有關如何 AWS IoT Core 使用 HTTPS 協議和 AWS Sigv4 身份驗證發布消息的指導。此專案還包括 .NET 核心同等實作。

對已簽署的 AWS API 請求進行排疑解難

> **Important**
>
> 除非您使用 AWS SDK 或 CLI，否則您必須撰寫程式碼來計算在要求中提供驗證資訊的簽章。SigV4 簽章計算可能是一項複雜的工作，我們建議您儘可能使用 AWS SDK 或 CLI。

當您開發建立已簽署要求的程式碼時，您可能會收到`SignatureDoesNotMatch`來自 AWS 服務. 這些錯誤意味著 HTTP 請求中的簽名值與 AWS 服務 計算的簽名 AWS 不匹配。當許可不允許呼叫者發出請求時，將傳回 HTTP 401 `Unauthorized` 錯誤。

如有以下情形，API 請求可能傳回錯誤：

- API 請求未簽署，且 API 請求使用 IAM 身分驗證。
- 用於簽署請求的 IAM 憑證不正確或沒有調用 API 的許可。
- 已簽署 API 要求的簽章與 AWS 服務計算的簽章不符。
- API 請求標頭不正確。

> **Note**
>
> 在探索其他錯誤解決方案之前，請將簽 AWS 名協議從 AWS 簽名版本 2（SigV2）更新為簽名版本 4（Sigv4）。服務 (例如 Amazon S3) 和區域不再支援 SigV2 簽署。

可能原因

- 憑證錯誤
- 正式請求與簽署字串錯誤
- 憑證範圍錯誤
- 金鑰簽署錯誤

憑證錯誤

確保使用 SigV4 簽署 API 請求。如果 API 請求未簽署，則您有可能收到錯誤：`Missing Authentication Token`。新增缺少的簽章並重新傳送請求。

驗證存取金鑰和私密金鑰的身分驗證憑證是正確的。如果存取金鑰錯誤，則您有可能收到錯誤：Unauthorized。確保用於簽署請求的實體有權發出請求。如需詳細資訊，請參閱 排查拒絕存取錯誤訊息問題。

正式請求與簽署字串錯誤

如果您錯誤計算 步驟 2：建立正式請求雜湊 或 步驟 3：建立登入字串 中正式請求，則該服務執行的簽章驗證步驟便會失敗，並顯示錯誤訊息：

```
The request signature we calculated does not match the signature you provided
```

當 AWS 服務收到已簽署的要求時，會重新計算簽章。如果值之間有差異，則簽名不匹配。使用錯誤訊息中的值，比對正式請求和字串以及您的已簽署請求。若有任何差異，請修改簽署程序。

> **Note**
>
> 您也可以驗證您未透過可修改標頭或請求的代理傳送請求。

Example 正式請求範例

```
GET                                                       -------- HTTP method
/                                                         -------- Path. For API stage
 endpoint, it should be /{stage-name}/{resource-path}
                                                          -------- Query string key-
value pair. Leave it blank if the request doesn't have a query string.
content-type:application/json                             -------- Header key-value
 pair. One header per line.
host:0123456789.execute-api.us-east-1.amazonaws.com       -------- Host and x-amz-date
 are required headers for all signed requests.
x-amz-date:20220806T024003Z

content-type;host;x-amz-date                              -------- A list of signed
 headers
d167e99c53f15b0c105101d468ae35a3dc9187839ca081095e340f3649a04501        -------- Hash
 of the payload
```

要驗證私密金鑰是否與存取金鑰 ID 相符，可以使用已知有效的實作來測試它們。例如，使用 AWS SDK 或 AWS CLI 向 AWS.

API 請求標頭

請確保您在 步驟 4：計算簽章 中新增的 SigV4 授權標頭包含與下列金鑰類似的正確憑證金鑰：

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=example-generated-signature
```

如果憑證金鑰缺失或錯誤，您有可能收到錯誤：`Authorization header requires 'Credential' parameter. Authorization header requires 'Signature' parameter.`。確保 SigV4 授權請求還包含使用 HTTP Date 或 x-amz-date 標頭的請求日期。

憑證範圍錯誤

您在 步驟 3：建立登入字串 中建立的憑證範圍會將簽章限制在特定日期、區域和服務。此字串的格式如下所示：

```
YYYYMMDD/region/service/aws4_request
```

> **Note**
>
> 如果您使用的是 SigV4a，區域不包含在憑證範圍之中。

日期

如果憑證範圍未指定與 x-amz-date 標頭相同的日期，則簽名驗證步驟會失敗，並顯示下列錯誤訊息：

```
Date in Credential scope does not match YYYYMMDD from ISO-8601 version of date from
 HTTP
```

如果請求指定未來的時間，則簽章驗證步驟會失敗，並顯示下列錯誤訊息：

```
Signature not yet current: date is still later than date
```

如果請求已過期，簽章驗證步驟會失敗，並顯示下列錯誤訊息：

```
Signature expired: date is now earlier than date
```

區域

如果憑證範圍未指定與請求相同的區域，則簽名驗證步驟會失敗，並顯示下列錯誤訊息：

```
Credential should be scoped to a valid Region, not region-code
```

服務

如果憑證範圍未指定與 host 標頭相同的服務，則簽名驗證步驟會失敗，並顯示下列錯誤訊息：

```
Credential should be scoped to correct service: 'service'
```

終止字串

如果憑證範圍沒有以 aws4_request 結尾，則簽名驗證步驟會失敗，並顯示下列錯誤訊息：

```
Credential should be scoped with a valid terminator: 'aws4_request'
```

金鑰簽署錯誤

簽署金鑰衍生不正確或不當使用密碼編譯而造成的錯誤，較難進行故障診斷。確認正式字串和登入字串正確無誤之後，您也可以檢查下列其中一個問題：

- 私密存取金鑰與您指定的存取金鑰 ID 不符。
- 您的金鑰衍生程式碼發生問題。

要驗證私密金鑰是否與存取金鑰 ID 相符，可以使用已知有效的實作來測試它們。例如，使用 AWS SDK 或向 AWS。 AWS CLI 如需取得範例，請參閱「請求簽章範例」。

IAM JSON 政策參考

本節介紹 IAM 中 JSON 政策的元素、變數和評估邏輯的詳細語法、描述和範例。如需一般詳細資訊，請參閱JSON 政策概觀。

本參考包括下列章節。

- IAM JSON 政策元素參考 — 進一步了解有關在建立政策時可以使用的元素。查看其他政策範例和了解條件，支援的資料類型以及它們在各種服務中的使用方式。
- 政策評估邏輯— 本節說明 AWS 要求、驗證方式，以及如何 AWS 使用原則來判斷資源的存取權限。

- IAM JSON 政策語言的文法 — 本節介紹用來在 IAM 中建立政策的語言的正式語法。
- AWS 受管理的工作職能政策 — 本節列出了直接對應到 IT 產業中常見任務功能的所有 AWS 受管政策。使用這些政策來授予執行特定任務角色中的某人所預期的任務所需的許可權。這些政策將許多服務的許可合整合到單一政策中。
- AWS 全域條件內容索引鍵— 本節包含可用來限制 IAM 政策中許可的所有 AWS 全域條件金鑰清單。
- IAM 和 AWS STS 條件內容金鑰— 本節包含可用來限制 IAM 政策中許可的所有 IAM 和 AWS STS 條件金鑰清單。
- AWS 服務的動作、資源和條件金鑰 — 本節顯示可在 IAM 政策中用作許可的所有 AWS API 作業清單。它還包括可用來進一步強化請求的服務專用的條件索引鍵。

IAM JSON 政策元素參考

JSON 政策文件由元素組成。這些元素按照在政策中使用的大致順序列出。元素的順序不會有影響，例如，`Resource` 元素可排在 `Action` 元素之前。您不需要指定政策中的任何 `Condition` 元素。如需進一步了解 JSON 政策文件的一般結構和目的，請參閱 JSON 政策概觀。

部分 JSON 政策元素會相互排斥。這表示您不能建立同時使用兩個元素的政策。例如，
您不能在同一政策陳述式中同時使用 `Action` 與 `NotAction`。相互排斥的其他組合包括 `Principal/NotPrincipal` 以及 `Resource/NotResource`。

政策中所包含的具體內容因服務而異，取決於服務所提供的動作、內含的資源類型等。當您針對特定服務編寫政策時，參考該服務的範例將會有所幫助。若需要支援 IAM 之所有服務的清單，以及探討這些服務的 IAM 和政策的文件連結，請參閱 AWS 與 IAM 搭配使用的服務。

當您建立或編輯 JSON 政策時，IAM 可以執行政策驗證以協助您建立有效的政策。IAM 會識別 JSON 語法錯誤，而 IAM Access Analyzer 會提供額外的政策檢查及建議，協助您進一步改良政策。若要進一步了解政策驗證的資訊，請參閱 驗證 IAM 政策。若要進一步了解 IAM Access Analyzer 政策檢查和可動作的建議，請參閱 IAM Access Analyzer 政策驗證。

主題

- IAM JSON 政策元素：Version
- IAM JSON 政策元素：Id
- IAM JSON 政策元素：Statement
- IAM JSON 政策元素：Sid
- IAM JSON 政策元素：Effect
- AWS 政策元素：Principal

- AWS 政策元素：NotPrincipal
- IAM JSON 政策元素：Action
- IAM JSON 政策元素：NotAction
- IAM JSON 政策元素：Resource
- IAM JSON 政策元素：NotResource
- IAM JSON 政策元素：Condition
- IAM 政策元素：變數與標籤
- IAM JSON 政策元素：支援的資料類型

IAM JSON 政策元素：Version

> **消除歧義注意事項**
>
> 此 Version JSON 政策元素與「政策版本」不同。Version 政策元素是在政策內使用，並定義政策語言的版本。另一方面，政策版本會在您在 IAM 中變更客戶受管政策時建立。變更的政策不會覆寫現有的政策。IAM 反而會建立新版本的受管政策。如果您在搜尋有關受管政策可用的多版本支援的詳細資訊，請參閱the section called “版本控制 IAM 政策”。

Version 政策元素指定用於處理政策的語言語法規則。若要使用所有可用的政策功能，請在所有政策的 Statement 元素之外包含下列 Version 元素。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

IAM 支援下列 Version 元素值：

- 2012-10-17. 這是政策語言的目前版本，您應該一律包含 Version 元素，並將其設定為 2012-10-17。否則，您無法使用此版本所引進的功能 (例如政策變數)。

- 2008-10-17. 這是舊版本的政策語言。您可以在較舊的現有政策上看到此版本。對於任何新的政策，或您更新任何現有政策時，請不要使用此版本。較新的功能 (例如政策變數) 將無法使用您的政策。例如，${aws:username} 這類變數無法辨識為變數，而是視為政策中的文字字串。

IAM JSON 政策元素：Id

Id 元素指定政策的選用識別符。該 ID 在不同的服務中的使用各有不同。以資源為基礎的政策中允許使用 ID，但是以身分為基礎的政策不允許使用 ID。

對於可讓您設定 ID 元素的服務，我們建議您對該值使用 UUID (GUID)，或者將 UUID 做為 ID 的一部分合併以確保唯一性。

```
{
  "Version": "2012-10-17",
  "Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 某些 AWS 服務 (例如 Amazon SQS 或 Amazon SNS) 可能需要此元素，而且對其具有唯一性需求。如需有關編寫政策的服務特定資訊，請參閱您正在使用的服務文件。

IAM JSON 政策元素：Statement

Statement 元素是政策的主要元素。此元素為必要。Statement 元素可包含單一陳述式，或是個別陳述式的陣列。每個個別的陳述式區塊都必須用大括號 { } 括起。針對多個陳述式，陣列必須用方括號 [ ] 括起。

```
"Statement": [{...},{...},{...}]
```

以下範例顯示政策，其中包含在單一 Statement 元素中的一系列三個陳述式 (政策可讓您存取 Amazon S3 主控台中您自己的「主資料夾」)。政策包含 aws:username 變數，這會在政策評估期間以請求的使用者名稱取代。如需更多詳細資訊，請參閱 簡介。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

IAM JSON 政策元素：Sid

您可以提供 Sid (陳述式 ID) 作為原則陳述式的選用識別碼。您可以將 Sid 值指派給陳述式陣列中的每個陳述式。您可以使用 Sid 值做為政策陳述式的描述。在允許您指定 ID 元素的服務 (例如 SQS 和 SNS) 中，Sid 值只是政策文件 ID 的子 ID。在 IAM 中，Sid 值在 JSON 政策中必須是唯一的。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStatementID",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

Sid 元素支援 ASCII 大寫字母 (A-Z)、小寫字母 (A-Z) 和數字 (0-9)。

IAM 未在 IAM API 中公開 Sid。您無法根據此 ID 擷取特定陳述式。

> **Note**
>
> 某些 AWS 服務 (例如 Amazon SQS 或 Amazon SNS) 可能需要此元素，而且對其具有唯一性需求。如需有關編寫政策的服務特定資訊，請參閱您正在使用的服務文件。

IAM JSON 政策元素：Effect

Effect 元素是必要的，且指定陳述式會導致允許或明確拒絕。Effect 的有效值為 Allow 和 Deny。Effect 值會區分大小寫。

```
"Effect":"Allow"
```

在預設情況下，拒絕存取資源。若要允許存取資源，您必須將 Effect 元素設定為 Allow。若要覆寫允許 (例如，以強制方式覆寫允許)，您必須將 Effect 元素設定為 Deny。如需更多詳細資訊，請參閱政策評估邏輯。

AWS 政策元素：Principal

使用資源型 JSON 政策中的 Principal 元素來指定允許或拒絕存取資源的主體。

您可以在資源型政策中使用 Principal 元素。許多服務支援資源型政策，包括 IAM。IAM 資源型政策類型是角色信任政策。在 IAM 角色中，使用角色信任政策中的 Principal 元素來指定誰可擔任該

角色。對於跨帳戶存取，您必須指定信任帳戶的 12 位數識別碼。若要了解在您信任區域 (受信任組織或帳戶) 外帳戶中的主體是否具有擔任您角色的許可，請參閱什麼是 IAM Access Analyzer？。

> **Note**
>
> 在您建立角色後，您可以變更帳戶為「*」，以允許每個人擔任該角色。若執行此操作，我們強烈建議您限制誰可以透過其他方式存取角色，例如 `Condition` 元素會限制只能存取特定 IP 地址。切勿讓任何人都能存取您的角色！

支援資源型政策的其他資源範例包括 Amazon S3 儲存貯體或 AWS KMS key。

您不能使用身分型政策中的 `Principal` 元素。身分型政策是指您連接到 IAM 身分 (使用者、群組或角色) 的許可政策。在這些情況中，主體由政策連接的身分隱含識別。

主題

- 指定主體
- AWS 帳戶 校長
- IAM 角色主體
- 角色工作階段主體
- IAM 使用者主體
- IAM Identity Center 主體
- AWS STS 同盟使用者工作階段主體
- AWS 服務主體
- AWS 選擇加入區域中的服務主體
- 所有主體
- 其他資訊

指定主體

您可以在資源型政策的 `Principal` 元素中指定主體，或在支援主體的條件金鑰中指定。

您可以在政策中指定以下任何主體：

- AWS 帳戶 和根使用者
- IAM 角色

- 角色工作階段
- IAM 使用者
- 聯合身分使用者工作階段
- AWS 服務
- 所有主體

您無法將使用者群組識別為政策 (例如資源型政策) 中的主體，因為群組與許可 (而非驗證) 相關，並且主體是經過驗證的 IAM 實體。

您可以使用陣列在以下幾節中為每個主體類型指定一個以上的主體類型。陣列可以使用一個或多個值。當您在元素中指定多個主體時，便會授予每個主體許可。這是邏輯 OR，而不是邏輯 AND，因為您一次只會驗證為一位主體。如果包含多個值，請使用方括號 ([和])，並以逗號分隔陣列的每個項目。下列範例政策會定義 123456789012 帳戶或 555555555555 帳戶的許可。

```
"Principal" : {
"AWS": [
  "123456789012",
  "555555555555"
  ]
}
```

> **Note**
>
> 您不能使用萬用字元來符合部分主體名稱或 ARN。

AWS 帳戶 校長

您可以在以資源為基礎的策略的Principal元素或支援主參與者的條件索引鍵中指定 AWS 帳戶 識別碼。這會委派帳戶的授權。當您允許存取其他帳戶時，該帳戶中的管理員必須接著授予該帳戶中身分 (IAM 使用者或角色) 的存取權。當您指定時 AWS 帳戶，您可以使用帳戶 ARN (arn: aw:iam::: 帳戶 ID: *#)############ ID* 組成的縮短表單。"AWS":

例如，指定帳戶 ID 123456789012，您就可以使用以下其中一種方法來指定 Principal 元素中的帳戶：

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

```
"Principal": { "AWS": "123456789012" }
```

帳戶 ARN 和簡化帳戶 ID 的行為方式相同。兩者都會委派帳戶的許可。使用 Principal 元素中的帳戶 ARN 不會將許可限制為帳戶的根使用者。

> **Note**
>
> 當您儲存包含簡化帳戶 ID 的資源型政策時，服務可能會將其轉換為主體 ARN。如此並不會變更政策的功能。

某些 AWS 服務支援指定帳戶主體的其他選項。例如，Amazon S3 可讓您使用以下格式指定正式使用者 ID：

```
"Principal": { "CanonicalUser":
 "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be" }
```

您也可以使用陣列指定多個 AWS 帳戶、(或規範使用者 ID) 做為主體。例如，您可以使用全部三種方法在儲存貯體政策中指定主體。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::123456789012:root",
    "999999999999"
  ],
  "CanonicalUser": "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"
}
```

IAM 角色主體

您可以在資源型政策的 Principal 元素中指定 IAM 角色主體 ARN，或在支援主體的條件金鑰中指定。IAM 角色是身分。在 IAM 中，身分是您可以指派許可的資源。角色信任另一個已驗證的身分來擔任該角色。這包括 AWS 中的主體或來自外部身分提供者 (IdP) 的使用者。當主體或身分擔任角色時，他們會收到具有擔任角色許可的暫時安全憑證。當他們使用這些工作階段認證在中執行作業時 AWS，它們會成為角色工作階段主體。

IAM 角色是存在於 IAM 中的身分。角色信任另一個已驗證的身分，例如中的主體 AWS 或來自外部身分識別提供者的使用者。當主體或身分擔任角色時，它們會接收暫時安全憑證。然後，他們可以使用這些憑證，作為角色工作階段主體執行 AWS中的操作。

當您在資源型政策中指定角色主體時，主體的有效許可會受限於限制角色許可的任何政策類型。其中包括工作階段政策和許可界限。如需有關如何評估角色工作階段的有效許可的詳細資訊，請參閱 政策評估邏輯。

若要在 Principal 元素中指定角色 ARN，請使用以下格式：

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:role/role-name" }
```

> **Important**
>
> 如果角色信任政策中您的 Principal 元素包含指向特定 IAM 角色的 ARN，則該 ARN 會在儲存政策時轉換為角色的唯一主體 ID。如果有人希望藉由刪除並重新建立角色來提升特權，這麼做可有助於減輕此類風險。您通常不會在主控台中看到此 ID，因為在顯示信任政策時，IAM 會反向轉換回角色 ARN。不過，如果您刪除角色，則關係會中斷。即使重新建立角色，您的政策都不再適用，因為新角色有不符合信任政策中所儲存的主體 ID 的新主體 ID。發生這種情況時，主體識別碼會顯示在 AWS 以資源為基礎的策略中，因為無法再將其對應回有效的 ARN。結果是，如果您刪除並重新建立了信任政策的 Principal 元素所引用的角色，您必須編輯政策中的角色來以正確 ARN 替換主體 ID。當您儲存政策時，ARN 再次轉換為角色的新主體 ID。

或者，您可以指定角色主體作為資源型政策中的主體，或使用 aws:PrincipalArn 條件金鑰建立廣泛許可政策。當您使用此索引鍵時，角色工作階段主體會根據擔任的角色 ARN 授予許可，而不是根據所產生之工作階段的 ARN 授予許可。由於 AWS 不會將條件索引鍵 ARN 轉換為 ID，因此如果您刪除角色，然後建立具有相同名稱的新角色，則授與角色 ARN 的權限仍然存在。在資源型政策的 Principal 元素中，使用 aws:PrincipalArn 條件金鑰及萬用字元 (*) 授予的許可，不受身分型政策類型 (例如許可界限或工作階段政策) 的限制，除非身分型政策包含明確拒絕。

角色工作階段主體

您可以在資源型政策的 Principal 元素中指定角色工作階段，或在支援主體的條件金鑰中指定。當主體或身分擔任角色時，他們會收到具有擔任的角色許可的暫時安全憑證。當他們使用這些工作階段認證在中執行作業時 AWS，它們會成為角色工作階段主體。

您用於角色工作階段主參與者的格式取決於用來擔任該角色的 AWS STS 作業。

此外，系統管理員可以設計程序來控制如何發出角色工作階段。例如，他們可以為使用者提供一鍵式解決方案，以建立可預測的工作階段名稱。如果您的系統管理員執行這項操作，則可以在政策或條件金

鑰中使用角色工作階段主體。否則，您可以在 aws:PrincipalArn 條件金鑰中將角色 ARN 指定為主體。將角色指定為主體的方式可以變更產生工作階段的有效許可。如需詳細資訊，請參閱 IAM 角色主體。

擔任角色工作階段主體

假定角色工作階段作業主參與者是使用作業所產生的工作階段主體。 AWS STS AssumeRole如需哪些主體可以使用此操作擔任角色的詳細資訊，請參閱 比較 AWS STS API 操作。

若要在 Principal 元素中指定擔任角色工作階段 ARN，請使用以下格式：

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name" }
```

當您在 Principal 元素中指定擔任角色工作階段時，您無法使用萬用字元 (*) 來表示所有工作階段。主體一律必須命名特定工作階段。

OIDC 工作階段主體

OIDC 工作階段主體是使用作業所 AWS STS AssumeRoleWithWebIdentity產生的工作階段主體。您可以使用外部 OIDC 提供者 (IdP) 登入，然後使用此操作擔任 IAM 角色。這會利用聯合身分，並發出角色工作階段。如需哪些主體可以使用此操作擔任角色的詳細資訊，請參閱 比較 AWS STS API 操作。

當您從 OIDC 提供者發出角色時，您會取得這種特殊類型的工作階段主體，其中包含 OIDC 提供者的相關資訊。

在您的政策中使用此主體類型，可根據信任的 Web 身分提供者允許或拒絕存取。若要在角色信任原則的Principal元素中指定 OIDC 角色工作階段 ARN，請使用下列格式：

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
```

```
"Principal": { "Federated": "www.amazon.com" }
```

```
"Principal": { "Federated": "graph.facebook.com" }
```

```
"Principal": { "Federated": "accounts.google.com" }
```

SAML 工作階段主體

S AML 工作階段主體是使用作業所產生的工 AWS STS `AssumeRoleWithSAML`作階段主體。您可以使用外部 SAML 身分提供者 (IdP) 登入，然後使用此操作擔任 IAM 角色。這會利用聯合身分，並發出角色工作階段。如需哪些主體可以使用此操作擔任角色的詳細資訊，請參閱 比較 AWS STS API 操作。

當您從 SAML 身分提供者發出角色時，會取得此特殊類型的工作階段主體，其中包含 SAML 身分提供者的相關資訊。

在您的政策中使用此主體類型，可根據信任的 SAML 身分提供者允許或拒絕存取。若要在角色信任政策的 `Principal` 元素中指定 SAML 身分角色工作階段 ARN，請使用下列格式：

```
"Principal": { "Federated": "arn:aws:iam::AWS-account-ID:saml-provider/provider-name" }
```

IAM 使用者主體

您可以在資源型政策的 `Principal` 元素中指定 IAM 使用者，或在支援主體的條件金鑰中指定。

> **Note**
>
> 在 `Principal` 元素中，Amazon Resource Name (ARN) 的使用者名稱部分區分大小寫。

```
"Principal": { "AWS": "arn:aws:iam::AWS-account-ID:user/user-name" }
```

```
"Principal": {
  "AWS": [
    "arn:aws:iam::AWS-account-ID:user/user-name-1",
    "arn:aws:iam::AWS-account-ID:user/user-name-2"
  ]
}
```

當您在 `Principal` 元素中指定使用者時，您無法使用萬用字元 (*) 來表示「所有使用者」。主體必須一律指定特定的使用者。

> **Important**
>
> 如果角色信任政策中您的 `Principal` 元素包含指向特定 IAM 使用者的 ARN，則該 ARN 會在儲存政策時將 ARN 轉換為使用者的唯一主體 ID。如果有人希望藉由刪除並重新建立使用者來

提升特權，這麼做可有助於減輕此類風險。您通常不會在主控台中看到此 ID，因為在顯示信任政策時還會反向轉換回使用者的 ARN。不過，如果您刪除使用者，則關係會中斷。即使重新建立使用者，政策都不再適用。這是因為新的使用者有新的主體 ID，其不符合儲存在信任政策中的 ID。發生這種情況時，主體識別碼會顯示在 AWS 以資源為基礎的策略中，因為無法再將其對應回有效的 ARN。結果是，如果您刪除並重新建立了信任政策的 `Principal` 元素所引用的使用者，您必須編輯角色來以正確 ARN 替換不正確的主體 ID。當您儲存政策時，IAM 再次將 ARN 轉換為使用者的新主體 ID。

IAM Identity Center 主體

在 IAM Identity Center 中，資源型政策中的主體必須定義為 AWS 帳戶 主體。若要指定存取權，請參照條件區塊中許可集合的角色 ARN。如需詳細資訊，請參閱《IAM Identity Center 使用者指南》中的資源政策、Amazon EKS 和 AWS KMS中的參考許可集合。

AWS STS 同盟使用者工作階段主體

您可以在資源型政策的 `Principal` 元素中指定聯合身分使用者工作階段，或在支援主體的條件金鑰中指定。

Important

AWS 建議您僅在必要時使用 AWS STS 聯合使用者工作階段，例如需要 root 使用者存取權時。反之，使用角色以委派許可。

AWS STS 同盟使用者工作階段主參與者是使用作業所產生的工 AWS STS `GetFederationToken`作階段主參與者。在此情況中， AWS STS 使用聯合身分作為獲取暫時存取字符的方法，而不是使用 IAM 角色。

在中 AWS，IAM 使用者或 AWS 帳戶根使用者 可以使用長期存取金鑰進行驗證。如需哪些主體可以使用此操作聯合身分的詳細資訊，請參閱 比較 AWS STS API 操作。

- IAM 聯合身分使用者 – IAM 使用者使用 `GetFederationToken` 操作聯合，該操作會產生 IAM 使用者的聯合身分使用者工作階段主體。
- 聯合身分根使用者 – 根使用者使用 `GetFederationToken` 操作聯合，該操作會產生根使用者的聯合身分使用者工作階段主體。

當 IAM 使用者或 root 使用者要求 AWS STS 使用此作業的臨時登入資料時，他們會開始臨時的聯合身分識別使用者工作階段。此工作階段的 ARN 是以聯合的原始身分為基礎。

若要在 Principal 元素中指定聯合身分使用者工作階段 ARN，請使用以下格式：

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:federated-user/user-name" }
```

AWS 服務主體

您可以在以資源為基礎的策略的Principal項目或支援主體的條件索引鍵中指定 AWS 服務。服務主體是服務的識別碼。

服務可承擔的 IAM 角色稱為 AWS 服務角色。服務角色必須包含信任政策。信任政策是連接到角色的資源型政策，這些政策定義可擔任該角色的主體。有些服務角色具有已預先定義的信任政策。不過，在某些情況下，您必須在信任政策中指定服務主體。IAM 政策中的服務主體不能是"Service": "*"。

服務主體的識別碼包含服務名稱，而且通常採用以下格式：

service-name.amazonaws.com

服務主體是由服務定義。您可以藉由開啟 AWS 與 IAM 搭配使用的服務，尋找某些服務的服務主體，檢查 Service-linked role (服務連結角色) 資料欄中是否為 Yes (是)，並開啟 Yes (是) 連結，以檢視該服務的服務連結角色文件。尋找該服務的 Service-Linked Role Permissions (服務連結角色許可) 區段，以檢視服務主體。

以下範例顯示一個可以連接到服務角色的政策。政策啟用兩種服務，分別為 Amazon ECS 和 Elastic Load Balancing，來擔任角色。然後服務可以執行指派給角色的許可政策所授予的任何任務 (不顯示)。若要指定多個服務主體，不用指定兩個 Service 元素；您可以只使用一個該元素。實際上，您可以將一組多個服務主體作為單一 Service 元素的值。

```
"Principal": {
    "Service": [
        "ecs.amazonaws.com",
        "elasticloadbalancing.amazonaws.com"
   ]
}
```

AWS 選擇加入區域中的服務主體

您可以在多個 AWS 區域啟動資源，以及您必須選擇加入的部分地區。如需您必須選擇加入的區域的完整清單，請參閱AWS 一般參考指南中的管理 AWS 區域。

當選擇加入區域中的 AWS 服務在相同區域內提出要求時，服務主體名稱格式會識別為其服務主要名稱的非區域化版本：

service-name.amazonaws.com

當選擇加入區域中的 AWS 服務向另一個區域發出跨區域要求時，服務主體名稱格式會識別為其服務主要名稱的區域化版本：

service-name.*{region}*.amazonaws.com

例如，您有一個位於區域 ap-southeast-1 中的 Amazon SNS 主題，而 Amazon S3 儲存貯體位於選擇加入區域 ap-east-1。您想要設定 S3 儲存貯體通知，將訊息發佈到 SNS 主題。若要允許 S3 服務將訊息張貼到 SNS 主題，您必須透過該主題以資源為基礎的存取政策授與 S3 服務主體 sns:Publish 權限。

如果您指定 S3 服務主體的非區域化版本 s3.amazonaws.com，則在存取政策主題中，從儲存貯體到主題的 sns:Publish 要求將會失敗。下列範例會在 SNS 主題存取政策的 Principal 政策元素中指定非區域化 S3 服務主體。

```
"Principal": { "Service": "s3.amazonaws.com" }
```

由於儲存貯體位於選擇加入的區域，而且要求是在同一個區域之外發出，因此 S3 服務主體會顯示為區域化服務主體名稱 s3.ap-east-1.amazonaws.com。當選擇加入區域中的服務向另一個區域提出要求時，您必須使用區域化 AWS 服務主要名稱。指定區域化服務主體名稱後，如果儲存貯體向位於另一個區域的 SNS 主題發出 sns:Publish 要求，則該要求將會成功。下列範例會在 SNS 主題存取政策的 Principal 政策元素中指定區域化 S3 服務主體。

```
"Principal": { "Service": "s3.ap-east-1.amazonaws.com" }
```

只有在您指定區域化服務主體名稱時，從選擇加入區域至另一個區域的跨區域要求的資源政策或服務主體型允許清單才會成功。

> **Note**
>
> 對於 IAM 角色信任政策，建議使用非區域化服務主體名稱。IAM 資源是全球性的，因此可以在任何區域中使用相同的角色。

所有主體

可使用萬用字元 (*) 在資源型政策的 Principal 元素中或支援主體的條件金鑰中指定所有主體。資源型政策 授予許可和條件金鑰用於限制政策陳述式的條件。

> **Important**
>
> 強烈建議您不要在資源型政策的 Principal 中搭配使用萬用字元 (*) 與 Allow 效果，除非您打算授予公開或匿名存取。否則，請指定預定的主體、服務或 Principal 元素中的 AWS 帳戶，然後進一步限制 Condition 元素中的存取。IAM 角色信任政策尤其如此，因為它們允許其他主體成為您帳戶中的主體。

對於資源型政策，搭配使用萬用字元 (*) 與 Allow 效果可授權存取所有使用者，包括匿名使用者 (公開存取)。對於您帳戶中的 IAM 使用者和角色主體，不需要其他許可。對於其他帳戶中的主體，他們也必須在其帳戶中具有身分型許可，以允許他們存取您的資源。這稱為跨帳戶存取。

對於匿名使用者，下列元素相同：

```
"Principal": "*"
```

```
"Principal" : { "AWS" : "*" }
```

您不能使用萬用字元來比對部分主體名稱或 ARN。

以下範例顯示可用於明確拒絕所有主體的資源型政策 (而不是 使用 Deny 來指定 NotPrincipal)，但在 Condition 元素中指定的主體除外。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UsePrincipalArnInsteadOfNotPrincipalWithDeny",
      "Effect": "Deny",
      "Action": "s3:*",
      "Principal": "*",
      "Resource": [
        "arn:aws:s3:::BUCKETNAME/*",
        "arn:aws:s3:::BUCKETNAME"
      ],
```

```
      "Condition": {
        "ArnNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::444455556666:user/user-name"
        }
      }
    }
  ]
}
```

其他資訊

如需詳細資訊，請參閱下列內容：

- Amazon Simple Storage Service 使用者指南中的儲存貯體政策範例
- Amazon Simple Notification Service 開發人員指南中的 Amazon SNS 政策範例
- Amazon Simple Queue Service 開發人員指南中的 Amazon SQS 政策範例
- AWS Key Management Service 開發人員指南中的索引鍵政策
- AWS 一般參考 中的帳戶識別符
- OIDC 聯盟

AWS 政策元素：NotPrincipal

您可以使用NotPrincipal元素拒絕存取所有主體，但 IAM 使用者、聯合身分使用者、IAM 角色 AWS 帳戶、 AWS 服務或元素中指定的其他主體除外。NotPrincipal

您可以在某些 AWS 服務 (包括 VPC 端點) 的以資源為基礎的政策中使用它。資源型政策是您直接內嵌在資源中的政策。您無法在以 IAM 身分為基礎的政策或 IAM 角色信任政策中使用 NotPrincipal 元素。

NotPrincipal 必須與 "Effect":"Deny" 搭配使用。不支援將其與 "Effect":"Allow" 搭配使用。

> **Important**
>
> 極少有需要使用 NotPrincipal 的情況。我們建議在決定使用 NotPrincipal 前，先了解其他授權選項。使用 NotPrincipal 時，會難以疑難排解多個政策類型的影響。我們建議改為將 aws:PrincipalArn 內容索引鍵與 ARN 條件運算子搭配使用。如需詳細資訊，請參閱所有主體。

使用 **Deny** 來指定 **NotPrincipal**

當您使用 NotPrincipal 和 Deny 時，您還必須指定非拒絕主體的帳戶 ARN。否則，政策可能會拒絕存取包含主體的整個帳戶。根據您包含在政策中的服務， AWS 可能先驗證帳戶，再來才是使用者。如果正在評估假定的角色使用者 (使用角色的使用者)， AWS 可能會先驗證帳戶，然後驗證角色，然後是確認的角色使用者。擔任角色的使用者由該使用者擔任角色時指定的角色工作階段名稱來進行識別。因此，我們強烈建議您明確包含使用者帳戶的 ARN，或同時包含角色 ARN 和包含該角色的帳戶的 ARN。

> **Important**
>
> 對於已附加許可界限政策的 IAM 使用者或角色，請勿使用所含 NotPrincipal 政策元素具有 Deny 效果的資源型政策陳述式。無論在 NotPrincipal 元素中指定何值，具有 Deny 效果的 NotPrincipal 元素將始終拒絕任何已附加許可界限政策的 IAM 主體。這會造成部分本可存取資源的 IAM 使用者或角色失去存取權。我們建議您變更資源型政策陳述式，將條件運算子 ArnNotEquals 與 aws:PrincipalArn 內容索引鍵搭配使用來限制存取，而不是使用 NotPrincipal 元素。如需有關許可界限的資訊，請參閱 IAM 實體的許可界限。

> **Note**
>
> 根據最佳實務，您應該在政策中包含適用於帳戶的 ARN。有些服務需要帳戶 ARN，雖然不是所有情況都是如此。不含必要的 ARN 的任何現有政策將持續運作，但包含這些服務的新政策必須符合這項需求。IAM 不會追蹤這些服務，所以建議您一律包括帳戶 ARN。

以下範例顯示了在同一政策陳述式中有效使用 NotPrincipal 與 "Effect": "Deny" 的方法。

Example 相同或不同帳戶中的 IAM 使用者範例

在下列範例中，除了 AWS 帳戶 444455556666 中名為 Bob 的使用者外，其他所有主參與者都會被明確拒絕存取資源。請注意，最佳作法是，NotPrincipal元素包含使用者 Bob 和 Bob 所屬的 ARN (arn:aws:iam::444455556666:root)。 AWS 帳戶 如果NotPrincipal元素僅包含 Bob 的 ARN，則原則的影響可能是明確拒絕存取 AWS 帳戶 包含使用者 Bob 的。在一些情況下，使用者不能擁有比其父帳戶更多的許可，因此，如果 Bob 的帳戶被明確拒絕存取，則 Bob 可能無法存取該資源。

當此範例屬於以資源為基礎的原則陳述式的一部分時，此範例會依預期方式運作，而該原則是附加至相同或不同的資源 AWS 帳戶 (而非 444455556666)。本範例本身不會向 Bob 授予存取權限，而

只會將 Bob 從被明確拒絕的主體的清單中省略。若要讓 Bob 存取資源，另一個政策陳述式必須使用 "Effect": "Allow" 明確允許存取。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Deny",
        "NotPrincipal": {"AWS": [
            "arn:aws:iam::444455556666:user/Bob",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::BUCKETNAME",
            "arn:aws:s3:::BUCKETNAME/*"
        ]
    }]
}
```

Example 相同或不同帳戶中的 IAM 角色範例

在下列範例中，除了 AWS 帳戶 444455556666 中指 cross-account-audit-app 定的假定角色使用者以外的所有主參與者都會明確拒絕存取資源。最佳作法 AWS 帳戶 是，NotPrincipal元素包含指定角色使用者 (cross-account-audit-app) 的 ARN、角色 (cross-account-read-only-角色) 以及角色所屬的 (444455556666)。如果 NotPrincipal 元素缺少角色的 ARN，政策的效果可能會明確拒絕對角色的存取。同樣，如果 NotPrincipal 元素缺少角色所屬 AWS 帳戶 的 ARN，政策的效果可能會明確拒絕對 AWS 帳戶 以及該帳戶中所有實體的存取。在某些情況下，假定的角色使用者不能擁有比其父角色更多的權限，而且角色不能擁有比其父項更多的權限 AWS 帳戶，因此當角色或帳戶明確拒絕存取時，假定的角色使用者可能無法存取資源。

當此範例屬於以資源為基礎的策略 AWS 帳戶 (附加至不同資源 (而非 444455556666) 的資源中的政策陳述式的一部分時，會如預期般運作。此範例本身並不允許存取指定的角色使用者 cross-account-audit-app，它只會 cross-account-audit-app 從明確拒絕的主參與者清單中省略。要授予對資源的 cross-account-audit-app 訪問權限，另一個策略語句必須明確允許使用"Effect": "Allow".

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Deny",
        "NotPrincipal": {"AWS": [
```

```
            "arn:aws:sts::444455556666:assumed-role/cross-account-read-only-role/cross-
account-audit-app",
            "arn:aws:iam::444455556666:role/cross-account-read-only-role",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::Bucket_AccountAudit",
            "arn:aws:s3:::Bucket_AccountAudit/*"
        ]
    }]
}
```

當您在 NotPrincipal 元素中指定取得角色工作階段時，您無法使用萬用字元 (*) 來表示「所有工作階段」。主體一律必須命名特定工作階段。

IAM JSON 政策元素：Action

Action 元素描述了將允許或拒絕的特定動作。陳述式必須包含 Action 或 NotAction 元素。每個 AWS 服務都有自己的一組動作，描述您可以使用該服務執行的任務。例如，Amazon S3 的動作清單可在 Amazon 簡單儲存服務使用者指南中的政策中的指定許可中找到，Amazon EC2 的動作清單可在 Amazon EC2 API 參考中找到， AWS Identity and Access Management 您可以在 IAM API 參考中找到的動作清單。若要尋找其他服務的動作清單，請參閱該服務的 API 參考文件。

您可以使用服務命名空間做為動作字首 (iam、ec2、sqs、sns、s3 等) 來指定值，其後跟隨要允許或拒絕的動作名稱。該名稱必須符合所支援的服務的動作。字首和動作名稱不區分大小寫。例如，iam:ListAccessKeys 與 IAM:listaccesskeys 相同。以下範例顯示不同服務的 Action 元素。

Amazon SQS 動作

```
"Action": "sqs:SendMessage"
```

Amazon EC2 動作

```
"Action": "ec2:StartInstances"
```

IAM 動作

```
"Action": "iam:ChangePassword"
```

Amazon S3 動作

```
"Action": "s3:GetObject"
```

您可以為 Action 元素指定多個值。

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage", "ec2:StartInstances",
 "iam:ChangePassword", "s3:GetObject" ]
```

您可以使用萬用字元 (*) 來存取特定 AWS 產品提供的所有動作。例如，以下 Action 元素適用於所有 S3 動作。

```
"Action": "s3:*"
```

您還可以使用萬用字元 (*) 做為動作名稱的一部分。例如，以下 Action 元素適用於包含字串 AccessKey 的所有 IAM 動作，包括 CreateAccessKey、DeleteAccessKey、ListAccessKeys 和 UpdateAccessKey。

```
"Action": "iam:*AccessKey*"
```

有些服務可讓您限制可用的動作。例如，Amazon SQS 可讓您僅提供所有可能的 Amazon SQS 動作的子集。在這種情況下，* 萬用字元不允許完全控制佇列；它只允許您已共用的部分動作。如需詳細資訊，請參閱《Amazon Simple Queue Service 開發人員指南》中的了解許可。

IAM JSON 政策元素：NotAction

NotAction 是進階政策元素，明確符合「除了」指定的動作清單外的所有內容。使用 NotAction 可以透過僅列出一些不相符的動作來產生較短的政策，而不是包括相符的長動作清單。中指定NotAction的動作不受政策聲明中的Allow或Deny效果影響。反之，這表示如果使用 Allow 效果，則允許未列出的所有適用動作或服務。此外，如果您使用 Deny 效果，則會拒絕此類未列出的動作或服務。將 NotAction 與 Resource 元素一起使用時，可以為政策提供範圍。這是如何 AWS 確定哪些操作或服務適用。如需詳細資訊，請參閱下列範例政策。

NotAction 與允許

您可以在陳述式中使用NotAction元素，"Effect": "Allow"以提供對 AWS 服務中所有動作的存取權，但中指定的動作除外NotAction。您可以將其與 Resource 元素一起使用，以便為政策提供範圍，將允許的動作限制為可以對指定資源執行的動作。

以下範例可讓使用者存取除了刪除儲存貯體之外可在任何 S3 資源上執行的所有 Amazon S3 動作。這不允許使用者使用 ListAllMyBuckets S3 API 操作，因為該動作需要 "*" 資源。此政策也不允許在其他服務中執行動作，因為其他服務動作不適用於 S3 資源。

```
"Effect": "Allow",
"NotAction": "s3:DeleteBucket",
"Resource": "arn:aws:s3:::*",
```

有時，您可能想要允許存取大量動作。透過使用 NotAction 元素，您可以有效地反轉陳述式，從而產生更短的動作清單。例如，由於 AWS 有太多服務，因此您可能想要建立政策，允許使用者執行除存取 IAM 動作以外的所有項目。

下列範例可讓使用者存取 IAM 以外的每個 AWS 服務中的每個動作。

```
"Effect": "Allow",
"NotAction": "iam:*",
"Resource": "*"
```

在相同陳述式中或在政策中的不同陳述式小心使用 NotAction 元素和 "Effect": "Allow"。NotAction 符合未明確列出或適用於指定資源的所有服務和動作，並可能導致授予使用者比您預期更多的許可。

NotAction 與拒絕

您可以在陳述式中使用 NotAction 元素 "Effect": "Deny" 拒絕存取除 NotAction 元素中指定的動作之外的所有列出的資源。此組合不允許列出的項目，而是明確拒絕未列出的動作。您仍必須允許您要允許的動作。

如果使用者未使用 MFA 登入，則以下條件範例拒絕存取非 IAM 動作。如果使用者使用 MFA 登入，則 "Condition" 測試失敗，最終的 "Deny" 陳述式沒有影響。但請注意，這不會授予使用者存取任何動作的許可；它只會明確拒絕除 IAM 動作以外的所有其他動作。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "DenyAllUsersNotUsingMFA",
        "Effect": "Deny",
        "NotAction": "iam:*",
        "Resource": "*",
        "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}}
    }]
```

```
}
```

如需拒絕特定區域以外動作存取 (特定服務除外) 的範例政策，請參閱 AWS: AWS 根據請求的區域拒絕訪問。

IAM JSON 政策元素：Resource

Resource 元素指定陳述式所涵蓋的一個或多個物件。陳述式必須包含 Resource 或 NotResource 元素。您使用 ARN 指定資源。如需有關 ARN 格式的詳細資訊，請參閱 IAM ARN。

每個服務都有自己的一組資源。雖然您始終使用 ARN 來指定資源，但資源 ARN 的詳細資訊取決於服務和資源。如需有關如何指定資源的詳細資訊，請參閱您要撰寫陳述式的服務文件。

> **Note**
>
> 有些服務不允許您為個別的資源指定動作；相反，您在 Action 或 NotAction 元素中列出的任何動作都套用到該服務中的所有資源。在這些情況下，您可以在 * 元素中使用萬用字元 Resource。

以下範例參考特定的 Amazon SQS 佇列。

```
"Resource": "arn:aws:sqs:us-east-2:account-ID-without-hyphens:queue1"
```

以下範例參考 AWS 帳戶中名為 Bob 的 IAM 使用者。

> **Note**
>
> 在 Resource 元素中，IAM 使用者名稱區分大小寫。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"
```

在資源 ARN 中使用萬用字元

資源 ARN 中可以使用萬用字元。您可以在 ARN 區段中 (由冒號分隔的部分) 使用萬用字元 (* 和 ?) 表示具有星號 (*) 的任何字元組合和具有問號 (?) 的任何單一字元。您可以在每個區段中使用多個 * 或? 字元。如果萬用字元 (*) 是資源 ARN 區段的最後一個字元，則其可以展開，以符合冒號界限以外的內容。我們建議您在由冒號分隔的 ARN 區段內使用萬用字元 (* 和 ?)。

> **Note**
>
> 您無法在識別 AWS 產品的服務區段中使用萬用字元。如需 ARN 客群的詳細資訊，請參閱 Amazon 資源名稱 (ARN)。

以下範例參考路徑為 `/accounting` 的所有 IAM 使用者。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/accounting/*"
```

以下範例參考特定 Amazon S3 儲存貯體中的所有項目。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
```

星號 (*) 字元可以展開以取代區段內的所有項目，包括如正斜線 (/) 等字元，這些字元可能會顯示為指定服務命名空間內的分隔符號。例如，假設下列 Amazon S3 ARN，因為相同的萬用字元擴充邏輯適用於所有服務。

```
"Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*/test/*"
```

ARN 中的萬用字元適用於儲存貯體中的所有以下物件，而不僅僅是列出的第一個物件。

```
DOC-EXAMPLE-BUCKET/1/test/object.jpg
DOC-EXAMPLE-BUCKET/1/2/test/object.jpg
DOC-EXAMPLE-BUCKET/1/2/test/3/object.jpg
DOC-EXAMPLE-BUCKET/1/2/3/test/4/object.jpg
DOC-EXAMPLE-BUCKET/1///test///object.jpg
DOC-EXAMPLE-BUCKET/1/test/.jpg
DOC-EXAMPLE-BUCKET//test/object.jpg
DOC-EXAMPLE-BUCKET/1/test/
```

考慮上一個清單中的最後兩個物件。Amazon S3 物件名稱可有效地以傳統分隔符號正斜線 (/) 字元開頭或結尾。雖然「/」作為分隔符，但在資源 ARN 中使用此字元時卻沒有特定的意義。它被視為與任何其他有效字元相同。ARN 不符合下列物件：

```
DOC-EXAMPLE-BUCKET/1-test/object.jpg
DOC-EXAMPLE-BUCKET/test/object.jpg
DOC-EXAMPLE-BUCKET/1/2/test.jpg
```

指定多個資源

您可以指定多個資源。以下範例參考兩個 DynamoDB 資料表。

```
"Resource": [
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/books_table",
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/magazines_table"
]
```

在資源 ARN 中使用政策變數

在 Resource 元素中，您可以在 ARN 中用於識別特定資源 (即 ARN 的尾隨部分) 的部分中使用 JSON 政策變數。例如，您可以使用索引鍵 {aws:username} 做為資源 ARN 的一部分，以表示目前使用者的名稱應做為資源名稱的一部分包含在內。以下範例顯示如何在 {aws:username} 元素中使用 Resource 索引鍵。此政策允許存取與目前使用者名稱符合的 Amazon DynamoDB 資料表。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:account-id:table/${aws:username}"
  }
}
```

如需有關 JSON 政策變數的詳細資訊，請參閱 IAM 政策元素：變數與標籤。

IAM JSON 政策元素：NotResource

NotResource 是進階政策元素，會明確比對除指定資源以外的每項資源。使用 NotResource 可以透過僅列出一些不相符的資源來產生較短的政策，而不是包括相符的長資源清單。這對套用在單一 AWS 服務中的政策特別有用。

例如，假設您有一個名為 HRPayroll 的群組。不應允許 HRPayroll 成員存取除 HRBucket 儲存貯體中的 Payroll 資料夾以外的任何 Amazon S3 資源。以下政策明確拒絕存取除列出的資源以外的所有 Amazon S3 資源。不過，請注意，此政策不授予使用者存取任何資源的許可。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
```

```
    "Action": "s3:*",
    "NotResource": [
      "arn:aws:s3:::HRBucket/Payroll",
      "arn:aws:s3:::HRBucket/Payroll/*"
    ]
  }
}
```

一般而言，要明確拒絕對資源的存取，您可以編寫使用 "Effect":"Deny" 的政策，其中包含個別列出每個資料夾的 Resource 元素。但是，在這種情況下，每次向 HRBucket 新增資料夾或向 Amazon S3 新增不應存取的資源時，都必須將其名稱新增至 Resource 的清單。如果您使用 NotResource 元素，則會自動拒絕使用者存取新資料夾，除非您將資料夾名稱新增到 NotResource 元素。

使用 NotResource 時，請記住，此元素中指定的資源是「唯一」不受限的資源。因此，這會限制可能套用至動作的所有資源。在上述範例中，政策只影響 Amazon S3 動作，因此只會影響 Amazon S3 資源。如果動作也包含 Amazon EC2 動作，則政策不會拒絕存取任何 EC2 資源。若要瞭解服務中的哪些動作允許指定資源的 ARN，請參閱服AWS 務的動作、資源和條件金鑰。

NotResource 與其他元素

"Effect": "Allow"、"Action": "*" 和 "NotResource": "arn:aws:s3:::HRBucket" 元素絕對不能一起使用。這個陳述式非常危險，因為它允許除 HRBucket S3 儲存貯體以外的所有資源 AWS 上執行所有動作。它甚至允許使用者自行新增允許他們存取 HRBucket 的政策。請勿執行此作業。

在相同陳述式中或在政策中的不同陳述式小心使用 NotResource 元素和 "Effect": "Allow"。NotResource 允許未明確列出的所有服務和資源，並可能導致授予使用者比您預期的更多許可。在相同陳述式中使用 NotResource 元素和 "Effect": "Deny" 會拒絕未明確列出的服務和資源。

IAM JSON 政策元素：Condition

Condition 元素 (或 Condition 區塊) 可讓您於政策生效時指定條件。Condition 元素是選用的。在 Condition 元素中，您所建置的表達式使用條件運算子 (等於、小於等等) 來比對政策中的內容索引鍵和值，以及請求內容中的索引鍵和值。若要進一步了解請求內容，請參閱 請求。

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

您在政策條件中指定的內容索引鍵可以是全域條件內容索引鍵或服務特定的內容索引鍵。全域條件內容索引鍵包含 aws: 字首。服務特定的內容索引鍵包含服務的字首。例如，Amazon EC2 可讓您使用

ec2:InstanceType 內容索引鍵撰寫條件，此索引鍵對於該服務是唯一的。若要檢視包含 iam: 字首的服務特定的 IAM 內容索引鍵，請參閱 IAM 和 AWS STS 條件內容金鑰。

內容索引鍵名稱不區分大小寫。例如，包括 aws:SourceIP 內容索引鍵等同於對 AWS:SourceIp 的測試。內容索引鍵值是否區分大小寫取決於您使用的條件運算子。例如，下列條件包含 StringEquals 運算子，以確保僅 johndoe 所做的請求會相符。使用者指定的 JohnDoe 會受到存取遭拒。

```
"Condition" : { "StringEquals" : { "aws:username" : "johndoe" }}
```

以下條件使用 StringEqualsIgnoreCase 運算子，以符合使用者指定的 johndoe 或 JohnDoe。

```
"Condition" : { "StringEqualsIgnoreCase" : { "aws:username" : "johndoe" }}
```

部分內容索引鍵支援索引鍵值對，可允許您指定部分索引鍵名稱。範例包括多個服務支援的ResourceTag/*tag-key*內容索引鍵 AWS KMS kms:EncryptionContext:*encryption_context_key*、和內容索引鍵。aws:RequestTag/*tag-key*

- 如果您對 Amazon EC2 之類的服務使用 ResourceTag/*tag-key* 內容索引鍵，則必須為 tag-key 指定索引鍵名稱。
- 金鑰名稱不區分大小寫。這表示如果您在政策的條件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，則該條件會符合名為 TagKey1 或 tagkey1 的資源標籤鍵 (但不會同時符合兩者)。
- AWS 支援這些屬性的服務可能允許您建立多個按大小寫不同的金鑰名稱。例如，可能使用 ec2=test1 和 EC2=test2 標記 Amazon EC2 執行個體。當您使用條件 (例如，"aws:ResourceTag/EC2": "test1") 以允許對該資源的存取，則金鑰名稱同時符合兩個標籤，但只有一個值符合。這會導致意外的條件失敗。

> **Important**
>
> 根據最佳實務，確保帳戶成員在命名鍵值對屬性時遵守一致的命名慣例。範例包括標籤或 AWS KMS 加密內容。您可以使用標記的aws:TagKeys內容金鑰或 AWS KMS 加密內容來kms:EncryptionContextKeys強制執行此操作。

- 如需所有條件運算子的清單及其運作方式的描述，請參閱條件運算子。

- 除非另有說明，否則，所有內容索引鍵都可以具有多個值。如需有關如何處理具有多個值的內容索引鍵的描述，請參閱 多值內容索引鍵
- 如需有關所有全域可用內容索引鍵的清單，請參閱 AWS 全域條件內容索引鍵。
- 如需每個服務所定義的條件內容索引鍵，請參閱服務的動作、資源和條件索 AWS 引鍵。

請求內容

當主體向其提出要求時 AWS，會將要求資訊 AWS 收集到要求前後關聯中。此資訊會用來評估和授權請求。您可以使用 JSON 政策的 Condition 元素來針對請求內容測試特定內容索引鍵。例如，您可以建立使用 aws: CurrentTime 上下文金鑰的政策，允許使用者僅在特定日期範圍內執行動作。

提交要求時，會 AWS 評估原則中的每個內容索引鍵，並傳回 true、false、不存在，偶爾為 null (空白資料字串) 的值。請求中不存在的內容索引鍵視為不相符。例如，下列政策允許移除您自己的多重要素驗證 (MFA) 裝置，但僅限您在過去一個小時 (3,600 秒) 中使用 MFA 登入。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Sid": "AllowRemoveMfaOnlyIfRecentMfa",
        "Effect": "Allow",
        "Action": [
            "iam:DeactivateMFADevice"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}",
        "Condition": {
            "NumericLessThanEquals": {"aws:MultiFactorAuthAge": "3600"}
        }
    }
}
```

請求內容可傳回下列值：

- True (真) – 若申請者在過去一個小時或一個小時內使用 MFA 登入，條件便會傳回 true。
- False (偽) – 若申請者在過去超過一個小時中使用 MFA 登入，條件便會傳回 false。
- 不存在 — 如果請求者使用 AWS CLI 或 AWS API 中的 IAM 使用者存取金鑰提出請求，則金鑰不存在。在此情況下，由於索引鍵不存在，因此不會相符。
- Null – 針對使用者定義的內容索引鍵 (例如在請求中傳遞標籤)，您可以包含空白字串。在此情況下，請求內容中的值是 null。在某些情況下，null 值也可能會傳回 true。例如，如果您搭配

ForAllValues 內容索引鍵使用多值 aws:TagKeys 條件運算子，若請求內容傳回 null，您便可能會遭遇未預期的結果。如需詳細資訊，請參閱 aws: TagKeys 和多值內容索引鍵.

條件區塊

以下範例顯示 Condition 元素的基本格式：

```
"Condition": {"StringLike": {"s3:prefix": ["janedoe/*"]}}
```

來自請求的值由內容索引鍵表示，在本例中為 s3:prefix。內容金鑰值會與您指定為常值的值進行比較，例如 janedoe/*。要進行的比較類型由 條件運算子 (在這裡為StringLike) 指定。您可以使用典型的布林比較 (例如，等於、大於和小於) 來建立比較字串、日期、數字等條件。當您使用字串運算子或 ARN 運算子時，您也可以在內容索引鍵值中使用政策變數。下列範例包含 aws:username 變數。

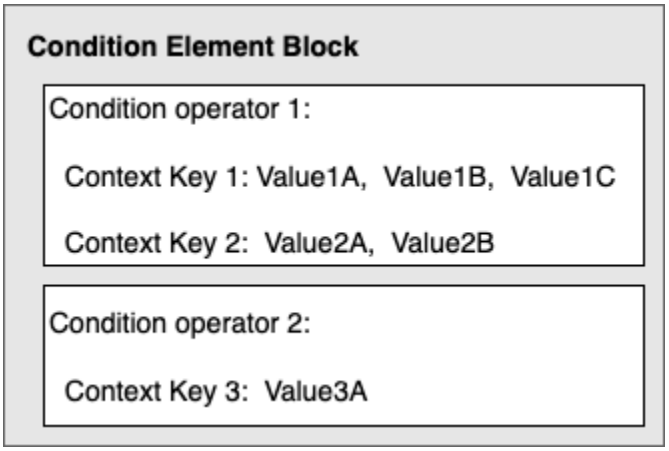
```
"Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
```

在某些情況下，內容索引鍵可以包含多個值。例如，對 Amazon DynamoDB 的請求可能會要求從資料表中傳回或更新多個屬性。用於存取 DynamoDB 資料表的政策可以包括 dynamodb:Attributes 內容索引鍵，其包含請求中列出的所有屬性。您可以使用 Condition 元素中的設定運算子，根據政策中允許的屬性清單測試請求中的多個屬性。如需詳細資訊，請參閱 多值內容索引鍵。

在要求期間評估原則時， AWS 會以要求中對應的值取代金鑰。(在此範例中， AWS 會使用要求的日期和時間。) 負責評估條件，以傳回 true 或 false，然後將該因素考慮為整個政策是否允許或拒絕請求。

條件中的多個值

Condition 元素可以包含多個條件運算子，而且每個條件運算子都可以包含多個內容索引鍵值對。下圖說明了這一點。



如需更多詳細資訊，請參閱 多值內容索引鍵。

IAM JSON 政策元素：條件運算子

在 Condition 元素中使用條件運算子來將政策中的條件鍵和值與請求內容中的值進行比對。如需有關 Condition 元素的詳細資訊，請參閱 IAM JSON 政策元素：Condition。

您可以在政策中使用的條件運算子取決於您選擇的條件索引鍵。您可以選擇全域條件索引鍵或服務限定條件索引鍵。若要了解針對全域條件索引鍵您可以使用的條件運算子，請參閱 AWS 全域條件內容索引鍵。若要瞭解可用於服務特定條件金鑰的條件運算子，請參閱AWS 服務的動作、資源和條件索引鍵，然後選擇您要檢視的服務。

> **Important**
>
> 如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符且條件為 false。如果政策條件要求索引鍵為不相符，例如 StringNotLike 或 ArnNotLike，並且正確的索引鍵不存在，則條件為 true。此邏輯適用於除了... 以外的所有條件運算子 IfExists和空檢查。這些運算子測試索引鍵是否存在於請求內容中。

條件運算子可分成以下幾種類別：

- ：字串
- 數值
- 日期和時間
- ：布林值
- ：二進位
- IP 地址
- Amazon Resource Name (ARN) (僅可用於部分服務。)
- ... IfExists（檢查鍵值是否存在作為另一個檢查的一部分）
- Null 檢查 (檢查鍵值是否做為獨立檢查存在)

字串條件運算子

運用字串條件運算子，您可以建構以索引鍵與字串值的對比為基礎來限制存取的 Condition 元素。

條件運算子	描述
StringEquals	完全相符，區分大小寫
StringNotEquals	否定相符
StringEqualsIgnoreCase	完全相符，不區分大小寫
StringNotEqualsIgnoreCase	否定相符，不區分大小寫
StringLike	大小寫相符。值可以在字串的任何位置包含多字元比對萬用字元 (*) 和單一字元比對萬用字元 (?)。您必須指定萬用字元才能達到部分字串相符。 **Note** 如果索引鍵包含多個值，StringLike 可使用集合運算子限定 ForAllValues:StringLike 和 ForAnyValue:StringLike。如需詳細資訊，請參閱多值內容索引鍵。
StringNotLike	否定大小寫相符。值可以在字串的任何位置包含多字元比對萬用字元 (*) 或單一字元比對萬用字元 (?)。

例如，下列陳述式包含的 Condition 元素使用 aws:PrincipalTag 索引鍵，以指定提出請求的主體必須以 iamuser-admin 任務類別標記。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"StringEquals": {"aws:PrincipalTag/job-category": "iamuser-admin"}}
  }
}
```

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。在此範例中，如果主體使用的 IAM 使用者已連接標籤，則 aws:PrincipalTag/job-category 索引鍵存在於請求內容中。如果主體使用的 IAM 角色有連接的標籤或工作階段標籤，則也包含此索引鍵。如果不具有標籤的使用者嘗試檢視或編輯存取金鑰，此條件會傳回 false，且此陳述式會隱含拒絕請求。

您可以使用政策變數與 String 條件運算子。

下列範例使用 StringLike 條件運算子執行與政策變數的字串配對來建立政策，該政策允許 IAM 使用者使用 Amazon S3 主控台管理其 Amazon S3 儲存貯體中的「主目錄」。該政策允許對 S3 儲存貯體執行指定操作，前提是 s3:prefix 與任一指定模式相符。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::BUCKET-NAME",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}",
        "arn:aws:s3:::BUCKET-NAME/home/${aws:username}/*"
      ]
    }
  ]
}
```

如需說明如何根據應用程式 ID 和 OIDC 聯盟的使用者識別碼，使用Condition元素限制資源存取的原則範例，請參閱。Amazon S3：可讓 Amazon Cognito 使用者存取其儲存貯體中的物件

萬用字元比對

字串條件運算子會執行不強制執行預先定義格式的無模式比對。ARN 和 Date 條件運算子是字串運算子的子集，它們會對條件索引鍵值強制執行一種結構。當您針對 ARN StringLike 或日期的部分字串相符項目使用或 StringNotLike運算子時，相符項目會忽略結構的哪一部分是萬用字元。

例如，下列條件會使用不同的條件運算子來搜尋 ARN 的部分相符項目。

使 ArnLike 用時，ARN 的分區，服務，帳戶 ID，資源類型和部分資源 ID 部分必須與請求內容中的 ARN 完全匹配。只有區域和資源路徑允許部分相符。

```
"Condition": {"ArnLike": {"aws:SourceArn": "arn:aws:cloudtrail:*:111122223333:trail/
*"}}
```

使 StringLike 用代替時 ArnLike，match 會忽略 ARN 結構，並允許部分匹配，而不管是通配符的部分。

```
"Condition": {"StringLike": {"aws:SourceArn": "arn:aws:cloudtrail:*:111122223333:trail/
*"}}
```

ARN	ArnLike	StringLike
arn:aws:cloudtrail:us-west-2:111122223333:trail/finance	匹配	匹配
arn:aws:cloudtrail:us-east-2:111122223333:trail/finance/archive	匹配	匹配
arn:aws:cloudtrail:us-east-2:444455556666:user/111122223333:trail/finance	無相符項目	匹配

數位條件運算子

利用數位條件運算子，您可以建構以索引鍵與整數或小數值的對比來限制存取的 Condition 元素。

條件運算子	描述
NumericEquals	相符
NumericNotEquals	否定相符
NumericLessThan	「小於」相符
NumericLessThanEquals	「小於或等於」相符
NumericGreaterThan	「大於」相符
NumericGreaterThanEquals	「大於或等於」相符

例如，下列聲明包含一個 Condition 元素，該元素使用 NumericLessThanEquals 條件運算子與 s3:max-keys 金鑰來指定申請者一次「最多」可在 example_bucket 內列出 10 個物件。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket",
    "Condition": {"NumericLessThanEquals": {"s3:max-keys": "10"}}
  }
}
```

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。在此範例中，當您執行 s3:max-keys 操作時，ListBucket 索引鍵永遠存在於請求中。如果此政策允許所有 Amazon S3 操作，則只有包含 max-keys 內容索引鍵且值小於或等於 10 的操作，才允許執行。

您不可使用政策變數與 Numeric 條件運算子。

日期條件運算子

運用日期條件運算子，您可以建構以索引鍵與日期/時間值的對比為基礎來限制存取的 Condition 元素。您同時使用這些條件運算子與 aws:CurrentTime 索引鍵或 aws:EpochTime 索引鍵。您必須指定日期/時間值，且其中一個 W3C 實作要採用 ISO 8601 日期格式或 epoch (UNIX) 時間格式。

> **Note**
>
> 日期條件運算子不允許使用萬用字元。

條件運算子	描述
DateEquals	符合特定日期
DateNotEquals	否定相符
DateLessThan	在特定日期與時間前相符
DateLessThanEquals	在特定日期與時間時或之前相符
DateGreaterThan	在特定日期與時間後相符
DateGreaterThanEquals	在特定日期與時間時或之後相符

例如，下列陳述式包含的 Condition 元素使用 DateGreaterThan 條件運算子搭配 aws:TokenIssueTime 索引鍵。此條件指定用來提出請求的臨時安全憑證已於 2020 年發行。此政策可以每天以程式設計方式更新，以確保帳戶成員使用全新的憑證。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"DateGreaterThan": {"aws:TokenIssueTime": "2020-01-01T00:00:01Z"}}
  }
}
```

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。只有在主體使用臨時憑證發出請求時，aws:TokenIssueTime 索引鍵才存在於請求內容中。金鑰不存在於 AWS CLI使用存取金鑰所建立的 AWS API 或 AWS SDK 要求中。在此範例中，如果 IAM 使用者嘗試檢視或編輯存取金鑰，則會拒絕請求。

您不可使用政策變數與 Date 條件運算子。

布林值條件運算子

運用布林值條件，您可以建構以索引鍵與「true」或「false」的對比來限制存取的 Condition 元素。

條件運算子	描述
Bool	布林值相符

例如，如果請求不在 SSL 上，則此身分型政策會使用 Bool 條件運算子搭配 aws:SecureTransport 金鑰，拒絕將物件和物件標籤複寫到目的地儲存貯體及其內容。

> **Important**
>
> 此政策不允許任何動作。將此政策與允許特定動作的其他政策結合使用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BooleanExample",
      "Action": "s3:ReplicateObject",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
        "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
```

```
  ]
}
```

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。aws:SecureTransport 請求內容會傳回 true 或 false。

您可以使用政策變數與 Boolean 條件運算子。

二進位條件運算子

運用 BinaryEquals 條件運算子，您可以建立測試二進位格式索引鍵值的 Condition 元素。它會比較指定金鑰位元組的值與政策中 base-64 編碼表示的二進位值。

```
"Condition" : {
  "BinaryEquals": {
    "key" : "QmluYXJ5VmFsdWVJbkJhc2U2NA=="
  }
}
```

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。

您不可使用政策變數與 Binary 條件運算子。

IP 地址條件運算子

運用 IP 地址條件運算子，您可以建構 Condition 元素，它們會以索引鍵與 IPv4 或 IPv6 地址或 IP 地址範圍的對比來限制存取。您可以搭配 aws:SourceIp 索引鍵來使用運算子。該值必須採用標準的 CIDR 格式 (例如 203.0.113.0/24 或 2001:DB8:1234:5678::/64)。如果您指定的 IP 地址沒有關聯的路由字首，IAM 將使用 /32 做為預設字首值。

一些 AWS 服務支持 IPv6，使用:: 代表 0 的範圍。若要了解某項服務是否支援 IPv6，請參閱該服務的文件。

條件運算子	描述
IpAddress	指定的 IP 地址或範圍
NotIpAddress	除指定 IP 地址或範圍外的所有 IP 地址

例如，下列聲明使用 IpAddress 條件運算子與 aws:SourceIp 索引鍵來指定必須來自 IP 範圍 203.0.113.0 到 203.0.113.255。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*AccessKey*",
    "Resource": "arn:aws:iam::account-id:user/*",
    "Condition": {"IpAddress": {"aws:SourceIp": "203.0.113.0/24"}}
  }
}
```

aws:SourceIp 條件索引鍵解析為發出請求的 IP 地址。如果請求源自 Amazon EC2 執行個體，則 aws:SourceIp 計算為執行個體的公有 IP 地址。

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。aws:SourceIp 索引鍵永遠存在於請求內容中，但請求者使用 VPC 端點提出請求時為例外。在此情況下，此條件會傳回 false，且此陳述式會隱含拒絕請求。

您不可使用政策變數與 IpAddress 條件運算子。

以下範例顯示如何混合使用 IPv4 與 IPv6 地址，以涵蓋您組織的所有有效 IP 地址。建議您使用您的 IPv6 地址範圍以及已擁有的 IPv4 範圍來更新組織的政策，以確保政策在您過渡到 IPv6 時繼續有效。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "someservice:*",
    "Resource": "*",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "203.0.113.0/24",
          "2001:DB8:1234:5678::/64"
        ]
      }
    }
  }
}
```

如果以使用者身分直接呼叫測試的 API，aws:SourceIp 條件索引鍵僅在 JSON 政策中有效。如果改為使用服務代表您呼叫目標服務，則目標服務看到的是進行呼叫的服務的 IP 地址而不是來源使用者的 IP 地址。例如，如果您使用 AWS CloudFormation 呼叫 Amazon EC2 為您建構執行個體，就可能會發

生這種情況。目前，無法透過進行呼叫的服務將來源 IP 地址傳遞給目標服務以在 JSON 政策中進行評估。對於這些服務 API 呼叫類型，請勿使用 aws:SourceIp 條件索引鍵。

Amazon Resource Name (ARN) 條件運算子

運用 Amazon Resource Name (ARN) 條件運算子，您可以建構以索引鍵與 ARN 的對比為基礎來限制存取的 Condition 元素。ARN 被視為一個字串。

條件運算子	描述
ArnEquals ，ArnLike	ARN 大小寫相符。ARN 的六個由冒號分隔開的部分都要單獨檢查，每一個部分都可包括一個多字元比對萬用字元 (*) 或一個單字元比對萬用字元 (?)。ArnEquals 和 ArnLike 條件運算子的行為完全相同。
ArnNotEquals ，ArnNotLike	ARN 否定相符。ArnNotEquals 和 ArnNotLike 條件運算子的行為完全相同。

您可以使用政策變數與 ARN 條件運算子。

下列以資源為基礎的政策範例顯示連接至 Amazon SQS 佇列的政策 (您想要將 SNS 訊息傳送到此佇列)。它授予 Amazon SNS 將訊息傳送到所選佇列的許可，但僅在該服務代表特定 Amazon SNS 主題傳送這些訊息時才授予此許可。您在 Resource 欄位中指定佇列，Amazon SNS 主題則作為 SourceArn 索引鍵的值。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "123456789012"},
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:REGION:123456789012:QUEUE-ID",
    "Condition": {"ArnEquals": {"aws:SourceArn":
 "arn:aws:sns:REGION:123456789012:TOPIC-ID"}}
  }
}
```

如果您在政策條件中指定的索引鍵不存在於請求內容中，則值不相符。只有在資源觸發服務以代替資源擁有者呼叫另一個服務時，aws:SourceArn 索引鍵才存在於請求內容中。如果 IAM 使用者嘗試直接執行此操作，此條件會傳回 false，且此陳述式會隱含拒絕請求。

... IfExists 條件運算子

除 IfExists 條件外，您可在任何條件運算子名稱的尾端加入 Null，例如 StringLikeIfExists。如果您是指「如果請求的內容中存在政策索引鍵，則依照政策所述來處理索引鍵。如果該索引鍵不存在，則評估條件元素為 true。」 陳述式中其他條件因素仍然可以導致不相符，但使用 ...IfExists 檢查時並非使用遺失索引鍵。如果您使用 "Effect": "Deny" 元素搭配否定條件運算子，例如 StringNotEqualsIfExists，即使標籤丟失，請求仍然被拒絕。

使用 **IfExists** 的範例

許多條件索引鍵描述有關特定類型的資源的資訊，僅當存取該類型的資源時才存在。這些條件索引鍵在其他類型的資源上不存在。當政策陳述式僅適用於一種類型的資源時，這不會導致問題。但是，有時單一陳述式可以適用於多種類型的資源，例如當政策陳述式從多個服務引用操作時，或是當服務中的給定操作存取同一服務中的多種不同資源類型時。在這種情況下，在政策陳述式中包含僅適用於一種資源的條件索引鍵可能會導致政策陳述式中的 Condition 元素失敗，從而使陳述式的 "Effect" 不適用。

例如，請設想以下政策範例：

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "THISPOLICYDOESNOTWORK",
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLike": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

上述政策的意圖是讓使用者可以啟動類型為 t1、t2 或 m3 的任何執行個體。但是，啟動執行個體要求存取除了執行個體本身之外的許多資源；例如映射、金鑰對、安全群組等。會針對啟動執行個體所需的每個資源來評估整個陳述式。這些其他資源沒有 ec2:InstanceType 條件金鑰，因此 StringLike 檢查會失敗，並且不會向使用者授予啟動「任何」執行個體類型的能力。

若要解決此問題，請改用 StringLikeIfExists 條件運算子。如此一來，僅有在條件索引鍵存在時才會進行測試。您會讀到以下政策：「如果正在檢查的資源有 "ec2:InstanceType" 條件索引鍵，那

麼只會在鍵值以 t1.、t2. 或 m3. 開頭時才允許該動作。如果正在檢查的資源沒有該條件索引鍵，則無需擔心它。』 與 StringLikeIfExists 條件運算子搭配使用時，條件索引鍵值中的星號 (*) 會被解譯為萬用字元，以實現部分字串相符。DescribeActions 陳述式包含在主控台檢視執行個體所需的動作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RunInstance",
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*",
            "Condition": {
                "StringLikeIfExists": {
                    "ec2:InstanceType": [
                        "t1.*",
                        "t2.*",
                        "m3.*"
             ]}}
        },
        {
            "Sid": "DescribeActions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeVpcs",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        }]
}
```

用於檢查條件索引鍵是否存在的條件運算子

使用 Null 條件運算子檢查授權時是否沒有條件索引鍵。在政策陳述式中使用 true (索引鍵不存在 - 為 null) 或 false (索引鍵存在且值不為 null)。

您不可使用政策變數與 Null 條件運算子。

例如，您可以使用此條件運算子確定使用者使用的是自己的針對該操作的憑證還是臨時憑證。如果使用者使用的是臨時憑證，則索引鍵 aws:TokenIssueTime 存在並具有一個值。以下範例顯示一個條件，該條件說明使用者在使用 Amazon EC2 API 時不能使用臨時憑證 (索引鍵不能存在)。

```
{
  "Version": "2012-10-17",
  "Statement":{
      "Action":"ec2:*",
      "Effect":"Allow",
      "Resource":"*",
      "Condition":{"Null":{"aws:TokenIssueTime":"true"}}
  }
}
```

具有多個內容索引鍵或值的條件

您可以使用政策的 Condition 元素來測試請求中單一內容索引鍵的多個內容索引鍵或多個值。當您以程式設計方式或透過向 AWS提出要求時 AWS Management Console，您的要求會包含有關主參與者、作業、標籤等的資訊。您可以使用內容索引鍵來測試請求中相符的內容索引鍵的值，以及政策條件中指定的內容索引鍵。若要了解請求中包含的資訊和資料，請參閱 請求內容。

主題

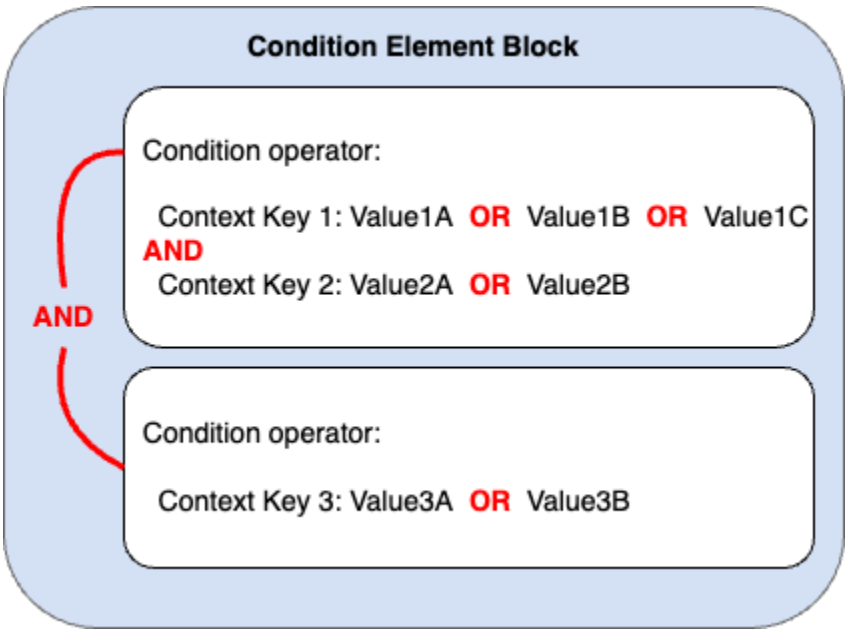
- 多個內容索引鍵或值的評估邏輯
- 否定相符條件運算子的評估邏輯

多個內容索引鍵或值的評估邏輯

Condition 元素可以包含多個條件運算子，而且每個條件運算子都可以包含多個內容索引鍵值對。除非另有指定，否則大多數內容索引鍵都支援使用多個值。

- 如果您的政策陳述式有多個條件運算子，則使用邏輯 AND 評估條件運算子。
- 如果您的政策陳述式已將多個內容索引鍵連接至單一條件運算子，則使用邏輯 AND 評估內容索引鍵。
- 如果單一條件運算子包含一個內容索引鍵的多個值，則使用邏輯 OR 評估這些值。
- 如果單一否定相符條件運算子包含一個內容索引鍵的多個值，則使用邏輯 NOR 評估這些值。

條件元素區塊中的所有內容索引鍵都必須解析為 true 才能調用所需的 Allow 或 Deny 效果。下圖說明具有多個條件運算子和內容索引鍵值對的條件的計算邏輯。



例如，下列 S3 儲存貯體政策說明上圖在政策中的表示方式。條件區塊包含條件運算子 `StringEquals` 和 `ArnLike`，以及內容索引鍵 `aws:PrincipalTag` 和 `aws:PrincipalArn`。若要調用所需的 `Allow` 或 `Deny` 效果，條件區塊中的所有內容索引鍵都必須解析為 true。發出請求的用戶必須同時具有部門和角色這兩個主體標籤索引鍵，其中包含政策中指定的其中一個標籤索引鍵值。此外，發出請求之使用者的主體 ARN 必須符合政策中指定的其中一個 `aws:PrincipalArn` 值才會評估為 true。

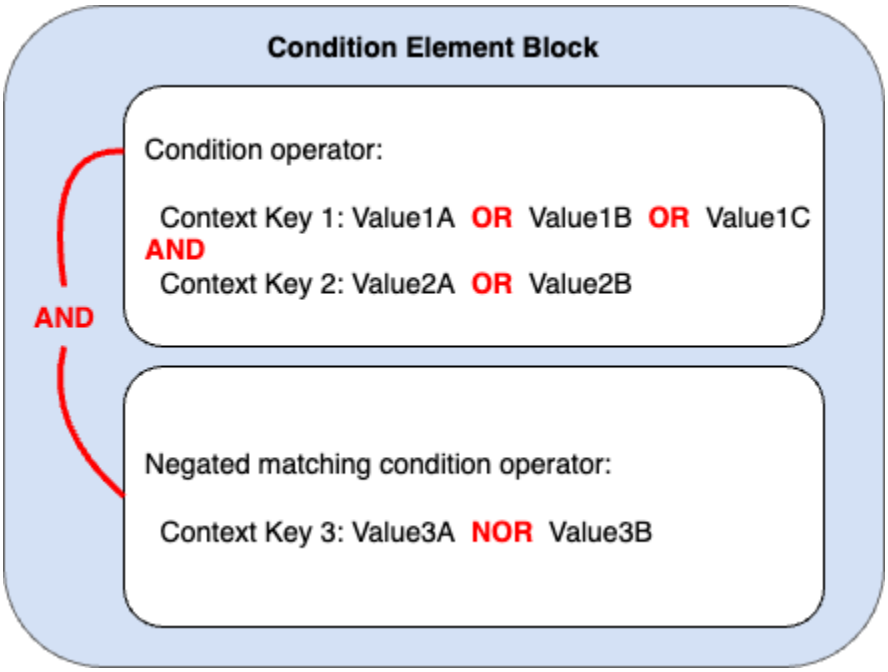
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",
            "hr",
            "legal"
          ],
          "aws:PrincipalTag/role": [
```

```
            "audit",
            "security"
          ]
        },
        "ArnLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:user/Ana",
            "arn:aws:iam::222222222222:user/Mary"
          ]
        }
      }
    }
  ]
}
```

否定相符條件運算子的評估邏輯

部分條件運算子 (例如 StringNotEquals 或 ArnNotLike) 使用否定相符將政策中的內容索引鍵值對與請求中的內容金鑰值對進行比較。使用否定相符條件運算子為政策中的單一內容索引鍵指定多個值時，有效許可的工作方式類似於邏輯 NOR。在否定相符中，僅當所有值都評估為 false 時，邏輯 NOR 或 NOT OR 才傳回 true。

下圖說明具有多個條件運算子和內容索引鍵值對的條件的計算邏輯。此圖包含內容索引鍵 3 的否定相符條件運算子。



例如，下列 S3 儲存貯體政策說明上圖在政策中的表示方式。條件區塊包含條件運算子 StringEquals 和 ArnNotLike，以及內容索引鍵 aws:PrincipalTag 和 aws:PrincipalArn。若要調用所需的 Allow 或 Deny 效果，條件區塊中的所有內容索引鍵都必須解析為 true。發出請求的用戶必須同時具有部門和角色這兩個主體標籤索引鍵，其中包含政策中指定的其中一個標籤索引鍵值。由於 ArnNotLike 條件運算子使用否定相符，因此發出請求的使用者的主體 ARN 必須符合要評估為 true 的政策中指定的任何 aws:PrincipalArn 值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",
            "hr",
            "legal"
          ],
          "aws:PrincipalTag/role": [
            "audit",
            "security"
          ]
        },
        "ArnNotLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:user/Ana",
            "arn:aws:iam::222222222222:user/Mary"
          ]
        }
      }
    }
  ]
}
```

單值與多值的內容索引鍵

單值與多值內容索引鍵之間的差異取決於請求內容中值的數目，而非政策條件中值的數目。

- 單值條件內容索引鍵在請求內容中最多有一個值。例如，您可以在 AWS中標記資源。資源標籤儲存為標籤金鑰/值對。資源標籤鍵可以有單一標籤值。因此，the section called “ResourceTag” 是單值內容索引鍵。請勿使用具有單值內容索引鍵的條件集運算子。
- 多值條件內容索引鍵在請求內容中可以擁有多個值。例如，您可以在中標記資源， AWS 並在請求中包含多個標籤鍵值配對。因此，the section called “TagKeys” 是多值內容索引鍵。多值內容索引鍵需要條件集運算子。

> **Important**
>
> 多值內容索引鍵需要條件集運算子。請勿將條件集運算子 `ForAllValues` 或 `ForAnyValue` 與單值內容索引鍵搭配使用。若要進一步了解條件集運算子，請參閱 多值內容索引鍵。

單值和多值分類包含在 AWS 全域條件內容索引鍵 主題中作為值類型的每個條件內容索引鍵的描述中。服務授權參考對多值內容索引鍵不同的值類型分類，格式如下：`ArrayOf` 字首後接條件運算子類別類型。例如 `ArrayOfString` 或 `ArrayOfARN`。

例如，請求可以來自最多一個 VPC 端點，因此 the section called “SourceVpce” 是單值內容索引鍵。由於服務可以有多個屬於該服務的服務主體名稱，因此 AWS：PrincipalServiceNamesList 是多值內容索引鍵。

您可以使用任何可用的單值內容索引鍵作為政策變數。您無法使用多值內容索引鍵作為政策變數。如需有關政策變數的詳細資訊，請參閱 IAM 政策元素：變數與標籤。

多值內容索引鍵需要條件集運算子 `ForAllValues` 或 `ForAnyValue`。包含索引鍵值對的內容索引鍵 (例如 the section called “RequestTag” 和 the section called “ResourceTag”) 可能會導致混淆，因為可能有多個 *`tag-key`* 值。但是，由於每個 *`tag-key`* 只能有一個值，因此 `aws:RequestTag` 和 `aws:ResourceTag` 都是單值內容索引鍵。使用條件集運算子搭配單值內容索引鍵可能會導致過度寬鬆的政策。

多值內容索引鍵

若要將您的條件內容索引鍵與具有多值的請求內容索引鍵進行比較，您必須使用 `ForAllValues` 或 `ForAnyValue` 集運算子。這些集運算子用於比較兩組值，例如請求中的標籤集和政策條件中的標籤集。

ForAllValues 和 ForAnyValue 限定詞新增集操作功能到條件運算子，以便您可以針對政策條件中的多內容索引鍵值測試具有多值的請求內容索引鍵。此外，如果您在政策中使用萬用字元或變數包含多值字串內容索引鍵，您也必須使用 StringLike 條件運算子。多個條件索引鍵值必須像陣列那樣用方括號括住。例如 "Key2":["Value2A", "Value2B"]。

- ForAllValues – 此限定詞測試請求集每個成員的值，是否為條件內容索引鍵集的子集。如果請求中每個內容索引鍵值至少符合政策中的一個內容索引鍵值，則條件會傳回 true。如果請求中沒有內容索引鍵，或內容索引鍵值解析為 null 資料集 (例如空白字串)，則也會傳回 true。為了防止遺失內容索引鍵或具有空值的內容索引鍵評估為 true，您可以在政策中包含具有 false 值的 Null 條件運算子，以檢查內容索引鍵是否存在且值不為 Null。

  > **Important**
  >
  > 如果將 ForAllValues 與 Allow 效果搭配使用，請務必小心，因為如果請求內容中意外出現遺失內容索引鍵或具有空值的內容索引鍵，則可能過度寬鬆。您可以在政策中包含具有 false 值的 Null 條件運算子，以檢查內容索引鍵是否存在且值不為 Null。如需範例，請參閱根據標籤索引鍵控制存取權限。

- ForAnyValue – 此限定詞測試這組請求內容索引鍵值是否至少有一個成員符合您的政策條件中這組內容索引鍵值的至少一個成員。如果請求中任一內容索引鍵值符合政策中的任一內容索引鍵值，則內容索引鍵會傳回 true。如果沒有相符內容索引鍵或為 null 資料集，則條件會傳回 false。

> **Note**
>
> 單值與多值內容索引鍵之間的差異取決於請求內容中值的數目，而非政策條件中值的數目。

條件政策範例

在 IAM 政策中，您可以為單值和多值內容索引鍵指定多個值，以便與請求內容進行比較。下列一組政策範例示範具有多個內容索引鍵和值的政策條件。

> **Note**
>
> 如果您想提交要包含在本參考指南中的政策，請使用此頁面底部的 Feedback (意見回饋) 按鈕。如需以身分為基礎的 IAM 政策範例，請參閱 以身分為基礎的 IAM 政策範例。

條件政策範例：單值內容索引鍵

- 具有單值內容索引鍵的多個條件區塊。(檢視此範例。)
- 具有多個單值內容索引鍵和值的一個條件區塊。(檢視此範例。)

條件政策範例：多值內容索引鍵

- 使用條件集運算子 ForAllValues 的拒絕政策。(檢視此範例。)
- 使用條件集運算子 ForAnyValue 的拒絕政策。(檢視此範例。)

多值內容索引鍵範例

下列一組政策範例示範如何使用多值內容索引鍵建立政策條件。

範例：使用條件集運算子拒絕策略 ForAllValues

下列以身分為基礎的政策範例會在請求中包含特定標籤索引鍵字首時，拒絕使用 IAM 標記動作。內容索引鍵 aws:TagKeys 的每個值都包含用於部分字串比對的萬用字元 (*)。此政策包含具有內容索引鍵 aws:TagKeys 的 ForAllValues 集運算子，因為請求內容索引鍵可以包含多值。為了使內容索引鍵 aws:TagKeys 傳回 true，請求中的每個值都必須與政策中的至少一個值相符。

如果請求中沒有內容索引鍵，或內容索引鍵值解析為 null 資料集 (例如空白字串)，則 ForAllValues 集運算子也會傳回 true。為了防止遺失內容索引鍵或具有空值的內容索引鍵評估為 true，請在政策中包含具有 false 值的 Null 條件運算子，以檢查請求中的內容索引鍵是否存在且值不為 Null。

> **Important**
>
> 此政策不允許任何動作。將此政策與允許特定動作的其他政策結合使用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyRestrictedTags",
      "Effect": "Deny",
      "Action": [
        "iam:Tag*",
        "iam:UnTag*"
      ],
```

```
      "Resource": [
        "*"
      ],
      "Condition": {
        "Null": {
          "aws:TagKeys": "false"
        },
        "ForAllValues:StringLike": {
          "aws:TagKeys": [
            "key1*",
            "key2*",
            "key3*"
          ]
        }
      }
    }
  ]
}
```

範例：使用條件集運算子拒絕策略 ForAnyValue

如果使用政策中指定的其中一個標籤索引鍵 (environment 或 webserver) 標記任何快照，下列以身分為基礎的政策範例將拒絕建立 EC2 執行個體磁碟區的快照。此政策包含具有內容索引鍵 aws:TagKeys 的 ForAnyValue 集運算子，因為請求內容索引鍵可以包含多值。如果您的標記請求包含政策中指定的任何一個標籤索引鍵值，aws:TagKeys 內容索引鍵將傳回 true 以調用拒絕政策效果。

> **Important**
>
> 此政策不允許任何動作。將此政策與允許特定動作的其他政策結合使用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "ec2:CreateSnapshot",
                "ec2:CreateSnapshots"
            ],
            "Resource": "arn:aws:ec2:us-west-2::snapshot/*",
```

```
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": ["environment", "webserver"]
                }
            }
        }
    ]
}
```

單值內容索引鍵政策範例

下列一組政策範例示範如何使用單值內容索引鍵建立政策條件。

範例：具有單值內容索引鍵的多個條件區塊

當條件區塊具有多個條件時，每個條件都有單一內容索引鍵，所有內容索引鍵都必須解析為 true 才能調用所需的 Allow 或 Deny 效果。當您使用否定相符條件運算子時，會反轉條件值的評估邏輯。

下列範例可讓使用者建立 EC2 磁碟區，並在磁碟區建立期間將標籤套用至磁碟區。請求內容必須包含內容索引鍵 aws:RequestTag/project 的值，以及內容索引鍵 aws:ResourceTag/environment 的值可以是生產以外的任何值。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVolume",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:::volume/*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/project": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
```

```
      "Resource": "arn:aws:ec2:region:account:*/*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/environment": "production"
        }
      }
    }
  ]
}
```

請求內容必須包含專案標籤值，且無法為生產資源建立以調用 Allow 效果。下列 EC2 磁碟區已成功建立，因為專案名稱為 Feature3，資源標籤為 QA。

```
aws ec2 create-volume \
    --availability-zone us-east-1a \
    --volume-type gp2 \
    --size 80 \
    --tag-specifications 'ResourceType=volume,Tags=[{Key=project,Value=Feature3},
{Key=environment,Value=QA}]'
```

範例：具有多個單值內容索引鍵和值的一個條件區塊

當條件區塊包含多個內容索引鍵，且每個內容索引鍵都具有多個值時，每個內容索引鍵都必須解析為 true，以便至少一個金鑰值能夠調用所需的 Allow 或 Deny 效果。當您使用否定相符條件運算子時，會反轉內容索引鍵值的評估邏輯。

下列範例可讓使用者在 Amazon Elastic Container Service 叢集上啟動和執行任務。

- 對於 aws:RequestTag/environment 內容索引鍵 AND，請求內容必須包含 production OR pre-prod。
- ecs:cluster 內容索引鍵確保任務在 default1 OR default2 ARN ECS 叢集上執行。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RunTask",
        "ecs:StartTask"
      ],
```

```
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": [
            "production",
            "prod-backup"
          ]
        },
        "ArnEquals": {
          "ecs:cluster": [
            "arn:aws:ecs:us-east-1:111122223333:cluster/default1",
            "arn:aws:ecs:us-east-1:111122223333:cluster/default2"
          ]
        }
      }
    }
  ]
}
```

IAM 政策元素：變數與標籤

如果您在撰寫政策時不知道資源或條件金鑰的確切值，請使用 AWS Identity and Access Management (IAM) 政策變數做為預留位置。

> **Note**
>
> 如果 AWS 無法解析變數，這可能會導致整個陳述式無效。例如，如果您使用 `aws:TokenIssueTime` 變數，此變數只會在請求者使用臨時憑證驗證時解析值 (IAM 角色)。若要防止變數造成無效陳述式，請使用... IfExists 條件運算符。

主題

- 指定預設值
- 如需詳細資訊

簡介

在 IAM 政策中，許多動作允許您為要控制存取的特定資源提供名稱。例如，下列政策可讓使用者為 marketing 專案列出、讀取和寫入 S3 儲存貯體 DOC-EXAMPLE-BUCKET 中的物件。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
      "Condition": {"StringLike": {"s3:prefix": ["marketing/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/marketing/*"]
    }
  ]
}
```

在某些情況下，您在編寫政策時可能不知道資源的確切名稱。您可能想要一般化政策，以使其適用於許多使用者，而不必為每個使用者製作政策的唯一副本。建議您建立適用於該群組中任何使用者的單一群組政策，而不是為每個使用者建立單獨政策。

在政策中使用變數

您可以使用在政策中設定預留位置的政策變數來定義政策內的動態值。

變數使用 **$** 字首後接一對大括號 (**{ }**) 進行標記，其中包含請求中的值的變數名稱。

評估政策時，政策變數將取代為來自請求中傳遞的條件式內容金鑰的值。變數可用於以身分為基礎的政策、資源政策、服務控制政策、工作階段政策以及 VPC 端點政策。用作許可界限的以身分為基礎的政策也支援政策變數。

全域條件內容索引鍵可以用作跨 AWS 服務要求中的變數。與 AWS 資源互動時，服務特定條件索引鍵也可以用作變數，但只有在針對支援其資源提出請求時才可用。如需每個 AWS 服務和資源可用的內容金鑰清單，請參閱服務授權參考。在某些情況下，您無法使用值填入全域條件內容索引鍵。若要進一步了解每個索引鍵，請參閱 AWS 全域條件內容索引鍵。

> **Important**
>
> - 索引鍵名稱不區分大小寫。例如，aws:CurrentTime 等同於 AWS:currenttime。
> - 您可以使用任何單一值條件鍵作為變數。您無法使用多重值條件鍵做為變數。

下面範例顯示了 IAM 角色或使用者的政策，它用政策變數替換特定資源名稱。您可以利用 aws:PrincipalTag 條件索引鍵來重複使用此政策。此政策被評估時，僅在儲存貯體名稱以 team 主體標籤中的團隊名稱作為結尾時，${aws:PrincipalTag/team} 才會允許該動作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:PrincipalTag/team}/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:PrincipalTag/team}/*"]
    }
  ]
}
```

該變數使用 $ 字首標記，後跟一對大括號 ({ })。在 ${ } 字元內，您可以包含要在政策中想要使用的請求中的值的名稱。您可以使用的值將在本頁稍後討論。

如需有關此全域條件索引鍵的詳細資訊，請參閱全域條件金鑰清單中的 AWS：PrincipalTag/標籤鍵。

> **Note**
>
> 若要使用政策變數，必須在陳述式中包含 Version 元素，並且必須將版本設定為支援政策變數的版本。變數已導入版本 2012-10-17。舊版的政策語言不支援政策變數。如果不包含 Version 元素，並將其設定為適當的版本，則 ${aws:username} 等變數將被視為政策中的常值字串。
>
> Version 政策元素與政策版本不同。Version 政策元素是在政策內使用，並定義政策語言的版本。另一方面，政策版本會在您在 IAM 中變更客戶受管政策時建立。變更的政策不會覆寫現有的政策。IAM 反而會建立新版本的受管政策。若要進一步了解 Version 政策元素，請參閱 the section called “Version”。若要進一步了解政策版本，請參閱 the section called “版本控制 IAM 政策”。

允許主體從 S3 儲存貯體的 /David 路徑取得物件之政策如下所示：

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3::DOC-EXAMPLE-BUCKET/David/*"]
  }]
}
```

如果此政策附加至使用者 David，該使用者會從其自己的 S3 儲存貯體取得物件，但您必須為包含使用者名稱的每個使用者建立獨立政策。您接著會將每個政策連接到個別使用者。

透過使用政策變數，您可以建立可重複使用的政策。如果 aws:PrincipalTag 的標籤鍵值與請求中傳遞的標籤鍵 owner 值相符，下列政策將允許使用者從 Amazon S3 儲存貯體取得物件。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowUnlessOwnedBySomeoneElse",
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["*"],
    "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/owner": "${aws:PrincipalTag/owner}"
```

```
        }
      }
    }
  ]
}
```

像這樣使用政策變數來替代使用者時，您不必為每個單獨使用者設定獨立的政策。在下列範例中，政策會附加至由產品經理使用臨時安全憑證擔任的 IAM 角色。使用者提出新增 Amazon S3 物件的請求時，IAM 會將目前請求中的 `dept` 標籤值替換為 `${aws:PrincipalTag}` 變數，並評估該政策。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowOnlyDeptS3Prefix",
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET/${aws:PrincipalTag/dept}/*"],
    }
  ]
}
```

做為政策變數的標籤

在某些 AWS 服務中，您可以將自己的自訂屬性附加至這些服務所建立的資源。例如，您可以將標籤套用至 Amazon S3 儲存貯體或 IAM 使用者。這些標籤均為鍵值組。您定義標籤鍵名稱以及與該鍵名稱關聯的值。例如，您可以建立一個具有 **`department`** 索引鍵和 **`Human Resources`** 值的標籤。如需有關標記 IAM 實體的詳細資訊，請參閱 標記 IAM 資源。有關標記其他 AWS 服務建立之資源的資訊，請參閱該服務的文件。如需有關使用標籤編輯器的詳細資訊，請參閱 AWS Management Console 使用者指南中的使用標籤編輯器。

您可以標記 IAM 資源以簡化探索、整理和追蹤您的 IAM 資源。您也可以標記 IAM 身分來控制存取資源或標記本身。若要進一步了解有關使用標籤以控制存取的詳細資訊，請參閱使用標籤控制對 IAM 使用者和角色的存取。

在此您可以使用政策變數

您可以在 `Resource` 元素中使用政策變數，也可以在 `Condition` 元素中使用字串比較。

資源元素

您可以在 Resource 元素中使用政策變數，但只能在 ARN 的資源部分中使用政策變數。ARN 的這個部分會出現在第 5 個冒號 (:) 之後。您無法使用變數來取代第 5 個冒號之前的 ARN 部分，例如服務或帳戶。如需有關 ARN 格式的詳細資訊，請參閱 IAM ARN。

若要以標籤值取代 ARN 的一部分，請以 ${ } 包住字首與索引鍵名稱。例如，以下資源元素僅指其名稱與請求之使用者的部門標籤值相同的儲存貯體。

```
"Resource": ["arn:aws::s3:::bucket/${aws:PrincipalTag/department}"]
```

許多 AWS 資源使用包含使用者建立名稱的 ARN。下列 IAM 政策可確保只有具有相符 access-project、access-application 和 access-environment 標籤值的特定使用者才能修改其資源。此外，使用 * 萬用字元比對時，它們能允許自訂資源名稱尾碼。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessBasedOnArnMatching",
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:DeleteTopic"],
      "Resource": ["arn:aws:sns:*:*:${aws:PrincipalTag/access-project}-
${aws:PrincipalTag/access-application}-${aws:PrincipalTag/access-environment}-*"
      ]
    }
  ]
}
```

條件元素

在涉及字串運算子或 ARN 運算子的任何條件下，您都可以將政策變數用於 Condition 值。字串運算子包括 StringEquals、StringLike 和 StringNotLike。ARN 運算子包括 ArnEquals 和 ArnLike。您無法使用政策變數搭配其他運算子，例如 Numeric、Date、Boolean、Binary、IP Address 或 Null 運算子。如需有關條件運算子的詳細資訊，請參閱 IAM JSON 政策元素：條件運算子。

在 Condition 元素表達式中參照標籤時，請使用相關的字首和索引鍵名稱做為條件金鑰。然後，使用您想在條件值中測試的值。

例如，下列政策範例允許完整存取使用者，但僅限在標籤 costCenter 連接至使用者時。此標籤也必須有一個 12345 或 67890 的值。如果此標籤沒有值，或有任何其他值，請求將會失敗。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
          "iam:*user*"
       ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:ResourceTag/costCenter": [ "12345", "67890" ]
        }
      }
    }
  ]
}
```

沒有值的政策變數

當政策變數參考的條件內容索引鍵沒有值或不存在於請求授權內容中時，該值實際上是空值。沒有相等或類似的值。在以下情況下，授權內容中可能不存在條件內容索引鍵：

- 您正在對不支援該條件索引鍵的資源請求中使用服務特定條件內容索引鍵。
- IAM 主體、工作階段、資源或請求的標籤不存在。
- 在 AWS 全域條件內容索引鍵 中列出了每個全域條件內容金鑰的其他情況。

當您在 IAM 政策的條件元素中使用沒有值的變數時 (IAM JSON 政策元素：條件運算子，例如 StringEquals 或 StringLike 不匹配)，政策陳述式不會生效。

反向條件運算子 (例如 StringNotEquals 或 StringNotLike) 確實與空值匹配，因為其正在測試的條件索引鍵的值不等於或不類似於有效空值。

在下列範例中，aws:principaltag/Team 必須等於 s3:ExistingObjectTag/Team 才能允許存取。未設定 aws:principaltag/Team 時，系統會明確拒絕存取。如果在授權內容中沒有值的變數用作政策的 Resource 或 NotResource 元素的一部分，則包含沒有值之政策變數的資源將不會與任何資源相匹配。

```
{
  "Version": "2012-10-17",
  "Statement": [
   {
    "Effect": "Deny",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::/example-bucket/*",
    "Condition": {
      "StringNotEquals": {
        "s3:ExistingObjectTag/Team": "${aws:PrincipalTag/Team}"
       }
      }
    }
  ]
}
```

您可以使用適用於政策變數的請求資訊

您可以使用 JSON 政策的 Condition 元素，來比較請求內容中的金鑰和您在政策中指定的鍵值。當您使用原則變數時，AWS 會從要求內容索引鍵取代原則中的變數值。

主體索引鍵值

aws:username、aws:userid 及 aws:PrincipalType 的值取決於啟動請求之主體的類型。例如，請求可能是使用 IAM 使用者、IAM 角色或 AWS 帳戶根使用者的憑證提出。下列清單顯示不同類型之主體的鍵值。

- AWS 帳戶根使用者
  - aws:username：(不存在)
  - aws:userid: AWS 帳戶 身份證
  - aws:PrincipalType: Account
- IAM 使用者
  - aws:username：*IAM-user-name*
  - aws:userid：唯一 ID
  - aws:PrincipalType: User
- 聯合身分使用者
  - aws:username：(不存在)
  - aws:userid：*account*:*caller-specified-name*

  - `aws:PrincipalType`: `FederatedUser`
- Web 聯合身分使用者和 SAML 聯合身分使用者

> **Note**
>
> 如需使用 OIDC 聯盟時可用之原則金鑰的相關資訊，請參閱。???

  - `aws:username`：(不存在)
  - `aws:userid`：(不存在)
  - `aws:PrincipalType`: `AssumedRole`
- 擔任的角色
  - `aws:username`：(不存在)
  - `aws:userid`：*`role-id`*:*`caller-specified-role-name`*
  - `aws:PrincipalType`: `Assumed role`
- 指派給 Amazon EC2 執行個體的角色
  - `aws:username`：(不存在)
  - `aws:userid`：*`role-id`*:*`ec2-instance-id`*
  - `aws:PrincipalType`: `Assumed role`
- 匿名呼叫者 (僅限 Amazon SQS、Amazon SNS 和 Amazon S3)
  - `aws:username`：(不存在)
  - `aws:userid`：(不存在)
  - `aws:PrincipalType`: `Anonymous`

對於此、清單中的項目，請注意下列事項：

- 「不存在」表示該值不在目前請求資訊中，並且任何符合它的嘗試都會失敗並導致陳述式無效。
- *`role-id`* 是在建立時指派給每個角色的唯一識別符。您可以使用以下 AWS CLI 命令顯示角色 ID：`aws iam get-role --role-name` *`rolename`*
- *`caller-specified-name`* 和 *`caller-specified-role-name`* 是呼叫程序 (例如，應用程式或服務) 在呼叫取得暫時憑證時所傳遞的名稱。

- *ec2-instance-id* 是在啟動時指派給執行個體的值，並顯示在 Amazon EC2 主控台的 Instances (執行個體) 頁面上。您也可以執行下 AWS CLI 列指令來顯示執行個體 ID：`aws ec2 describe-instances`

聯合身分使用者在請求中可用的資訊

聯合身分使用者是使用 IAM 以外的系統進行身分驗證的使用者。例如，一家公司可能有一個在內部使用的應用程序，可以撥打電話 AWS。向使用該應用程式的每個公司使用者提供 IAM 身分可能是不切實際的。相反地，公司可能會使用具有單一 IAM 身分的代理 (中間層) 應用程式，或者公司可能使用 SAML 身分提供者 (IdP)。代理應用程式或 SAML IdP 使用公司網路對個別使用者進行身分驗證。然後，代理應用程式可以使用其 IAM 身分來取得個別使用者的臨時安全憑證。SAML IdP 實際上可以將身分資訊交換為 AWS 臨時安全登入資料。然後，臨時登入資料可用於存取 AWS 資源。

同樣，您可以為應用程式需要存取 AWS 資源的行動裝置建立應用程式。在這種情況下，您可以使用 OIDC 聯合，該應用程序使用知名身份提供商（例如使用亞馬遜，Amazon Cognito，Facebook 或 Google 登錄）對用戶進行身份驗證。然後，應用程式可以使用來自這些提供者的使用者身分驗證資訊來取得用來存取 AWS 資源的臨時安全憑證。

使用 OIDC 聯盟的建議方式是利用 Amazon Cognito 和行動開發套件 AWS 。如需詳細資訊，請參閱下列內容：

- Amazon Cognito 使用者指南
- 暫時性憑證的常見案例

特殊字元

有幾個特殊的預先定義的政策變數具有固定值，使您能夠表示具有特殊含意的字元。如果這些特殊字元是字串的一部分，那麼您正在嘗試比對，並且從字面上插入它們會被轉譯。例如，在字串中插入 * 星號將被轉譯為萬用字元，比對任何字元，而不是做為文字 *。在這些情況下，您可以使用以下預定義的政策變數：

- ${*} - 在需要 * 星號字元的地方使用。
- ${?} - 在需要 ? 問號字元的地方使用。
- ${$} - 在需要 $ 貨幣符號字元的地方使用。

這些預先定義的政策變數可以在可以使用一般政策變數的任何字串中使用。

指定預設值

若要將預設值新增到一個變數，請以單引號 (' ') 括住預設值，並使用逗號和空格 (,) 分隔變數文字和預設值。

例如，如果主體標記為 team=yellow，他們可以存取名為 DOC-EXAMPLE-BUCKET-yellow 的 ExampleCorp's Amazon S3 儲存貯體。具有此資源的政策可讓團隊成員存取其團隊儲存貯體，但不能存取其他團隊的儲存貯體。對於沒有團隊標籤的使用者，它會將 company-wide 的預設值設定為儲存貯體名稱。這些使用者只能存取 DOC-EXAMPLE-BUCKET-company-wide 儲存貯體，其中他們可以檢視廣泛的資訊，例如加入團隊的指示。

```
"Resource":"arn:aws:s3:::DOC-EXAMPLE-BUCKET-${aws:PrincipalTag/team, 'company-wide'}"
```

如需詳細資訊

如需有關政策的詳細資訊，請參閱以下項目：

- IAM 中的政策和許可
- 以身分為基礎的 IAM 政策範例
- IAM JSON 政策元素參考
- 政策評估邏輯
- OIDC 聯盟

IAM JSON 政策元素：支援的資料類型

本部分列出在 JSON 政策中指定值時所支援的資料類型。政策語言不支援每個政策元素的所有類型；有關每個元素的相關資訊，請參閱上一個部分。

- Strings
- 編號 (Ints 和 Floats)
- Boolean
- Null
- 清單
- 地圖
- 結構 (只是巢狀地圖)

下表將每種資料類型對應至序列化。請注意，所有政策必須採用 UTF-8。如需 JSON 資料類型的資訊，請前往 RFC 4627。

類型	JSON
字串	字串
整數	Number
Float	Number
Boolean	true false
Null	null
日期	字串遵守 ISO 8601 的 W3C 描述檔
IpAddress	字串遵守 RFC 4632
列出	Array
物件	物件

政策評估邏輯

當主體嘗試使用 AWS Management Console、 AWS API 或該 AWS CLI主體傳送要求時 AWS。當 AWS 服務收到要求時，請 AWS 完成數個步驟以決定是否允許或拒絕要求。

1. 驗證 — 必要時會 AWS 先驗證發出要求的主體。有少數幾個服務並不需要這個步驟，例如 Amazon S3，它允許匿名使用者的一些請求。
2. 處理請求內容— AWS 處理要求中收集的資訊，以決定要求套用哪些原則。
3. 運用單一帳戶評估政策— 評 AWS 估所有策略類型，這會影響評估策略的順序。
4. 決定是否允許或拒絕帳戶中的請求— AWS 然後針對要求內容處理原則，以判斷要求是允許還是拒絕。

處理請求內容

AWS 會處理要求，將下列資訊收集到要求內容中：

- 動作 (或操作) – 主體想要執行的動作或操作。
- 資源 — 執行動作或作業時所依據的 AWS 資源物件。
- 主體 – IAM 使用者、角色、聯合身分使用者或傳送請求的應用程式。有關主體的資訊，包括與該主體相關聯的政策。
- 環境資料 – IP 地址、使用者代理程式、啟用 SSL 的狀態或一天中時間的相關資訊。
- 資源資料 – 與所請求資源相關的資料。這可以包括諸如 DynamoDB 資料表名稱或 Amazon EC2 執行個體上之標籤的資訊。

AWS 然後使用此資訊來尋找適用於請求前後關聯的原則。

運用單一帳戶評估政策

AWS 評估原則的方式取決於套用至請求前後關聯的原則類型。下面以頻率順序列出的政策類型，可以提供單一 AWS 帳戶使用。如需有關這些政策類型的詳細資訊，請參閱 IAM 中的政策和許可。若要瞭解如何 AWS 評估跨帳戶存取的政策，請參閱跨帳戶政策評估邏輯。

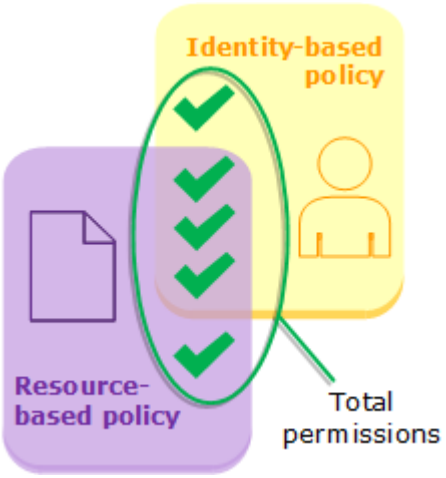
1. 以身分為基礎的政策 – 以身分為基礎的政策會連接到 IAM 身分 (使用者、使用者群組或角色)，並且授予許可給 IAM 實體 (使用者與角色)。如果要求只套用以身分識別為基礎的原則，則至少 AWS 會檢查所有這些原則中的一個原則。`Allow`
2. 資源型政策 – 資源型政策會授予許可給指定為主體的主體 (帳戶、使用者、角色和工作階段主體，例如角色工作階段和 IAM 聯合身分使用者)。這些許可會定義主體可以對政策連接於其中的資源做哪些動作。如果以資源為基礎的政策和以身分識別為基礎的策略都適用於請求，則至少 AWS 會檢查所有策略中的一個。`Allow`評估資源型政策時，政策中指定的主體 ARN 會決定其他政策類型中的隱含拒絕是否適用於最終決定。
3. IAM 許可界限 – 許可界限是一種進階功能，可負責設定以身分為基礎的政策可以授予 IAM 實體 (使用者或角色) 的最大許可。當您為實體設定許可界限時，實體只能執行由以身分為基礎的政策和其許可界限同時允許的動作。在某些情況下，許可界限中的隱含拒絕會限制資源型政策所授予的許可。如需進一步了解，請參閱本主題稍後的 決定是否允許或拒絕帳戶中的請求。
4. AWS Organizations 服務控制策略 (SCP) — 組 Organizations SCP 可指定組織或組織單位 (OU) 的最大權限。SCP 最大值適用於成員帳戶中的主參與者，包括每個主參與者。 AWS 帳戶根使用者如果 SCP 存在，則只有在這些政策和 SCP 允許該動作執行時，以身分為基礎和以資源為基礎的政策才會對成員帳戶中的主體授予許可。如果許可界限和 SCP 兩者都存在，則此界限、SCP 和以身分為基礎的政策都必須允許該動作。
5. 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身分使用者的暫時工作階段時，作為參數傳遞。若要以程式設計方式建立角色工作階段，請使用其中一

種 AssumeRole* API 操作。當您這麼做且傳遞工作階段政策時，所產生工作階段的許可會是 IAM 實體的身分類型政策和工作階段政策的交集。若要建立聯合身分使用者工作階段，您要使用 IAM 使用者存取金鑰，以程式設計的方式來呼叫 GetFederationToken API 操作。以資源為基礎的政策對工作階段政策許可的評估會有不同的效果。效果差異會因使用者或角色的 ARN 或工作階段的 ARN 是否在以資源為基礎的政策中列為主體而有所不同。如需詳細資訊，請參閱 工作階段政策。

請記住，所有這類政策中的明確拒絕都會覆寫該允許。

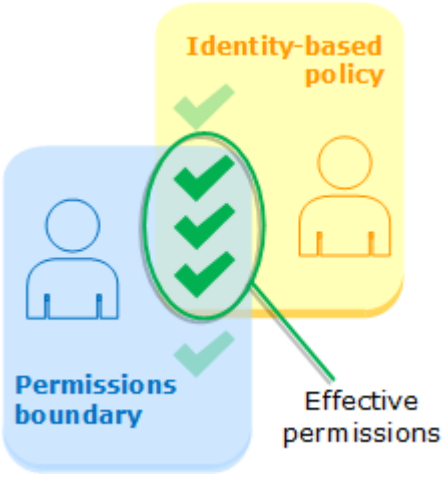
搭配以資源為基礎的政策來評估以身分為基礎的政策

以身分為基礎的政策和以資源為基礎的政策，可為其連接到其中的身分或資源授予許可。當 IAM 實體 (使用者或角色) 要求存取同一帳戶內的資源時， AWS 會評估以身分為基礎和以資源為基礎的政策授予的所有許可。最後得到的許可總括了這兩類許可。如果以身分為基礎的策略、以資源為基礎的策略或兩者都允許動作，則 AWS 允許該動作。在這些政策的明確拒絕會覆寫允許。



搭配許可界限來評估以身分為基礎的政策

AWS 評估使用者以身分識別為基礎的原則和權限界限時，產生的權限就是這兩個類別的交集。這表示，當您新增許可界限到已有現有以身分為基礎之政策的使用者時，您可能會減少使用者可執行的動作。或者，當您移除使用者的許可界限時，您可能會增加他們可以執行的動作。在這些政策的明確拒絕會覆寫允許。若要檢視有關如何搭配許可界限來評估其他政策類型的詳細資訊，請參閱評估含界限的有效許可。



搭配 Organizations SCP 來評估以身分為基礎的政策

當使用者所屬的帳戶是組織的成員，產生的許可會是使用者政策和 SCP 的交集。這就代表該動作必須同時獲得以身分為基礎的政策和 SCP 的允許。在這些政策的明確拒絕會覆寫允許。

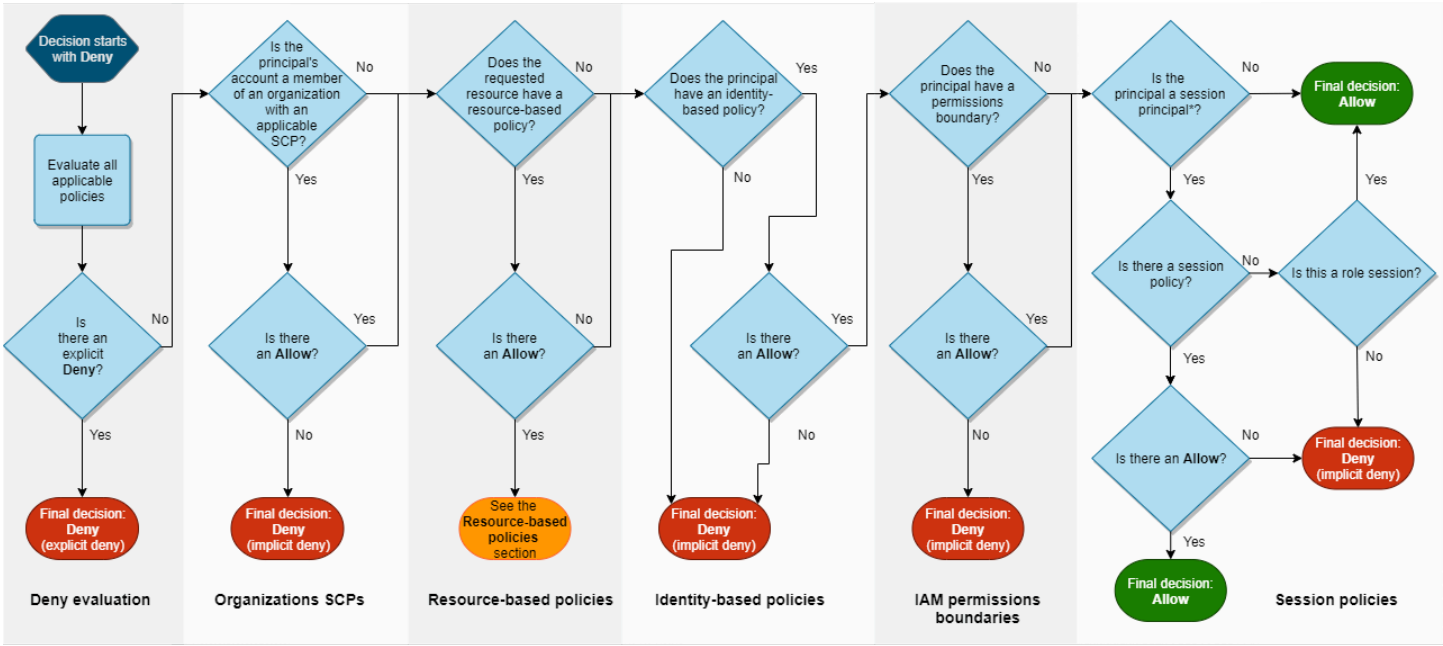
您可以了解您的帳戶是否為 AWS Organizations中組織的成員。組織成員可能會受到 SCP 影響。若要使用 AWS CLI 命令或 AWS API 作業檢視此資料，您必須擁有 Organizations 實體`organizations:DescribeOrganization`動作的權限。您必須擁有其他許可，才能在 Organizations 主控台中執行操作。若要瞭解 SCP 是否拒絕存取特定要求，或是要變更您的有效權限，請聯絡您 AWS Organizations 的系統管理員。

決定是否允許或拒絕帳戶中的請求

假設主參與者傳送要求， AWS 以存取與主參與者實體相同帳戶中的資源。 AWS 強制執行代碼決定應該允許還是拒絕請求。 AWS 評估適用於請求前後關聯的所有原則。以下是單一帳戶內原則 AWS 評估邏輯的摘要。

- 根據預設，所有要求都會隱含拒絕 AWS 帳戶根使用者，但具有完整存取權限的除外。
- 若是以身分為基礎或以資源為基礎的政策，當中的明確允許會覆寫此預設值。
- 如果有許可界限、Organizations SCP 或工作階段政策，則其可能會以隱含拒絕覆寫該允許。
- 任何政策中的明確拒絕會覆寫任何允許。

以下流程圖提供有關如何做出決策的詳細資訊。此流程圖不涵蓋以資源型政策和其他類型政策中隱含拒絕的影響。



*A session principal is either a role session or an IAM federated user session.

1. 拒絕評估 – 根據預設，所有的請求一律拒絕。這稱為隱含拒絕。 AWS 強制執行程式碼會評估帳戶內套用至要求的所有策略。其中包括 AWS Organizations SCP、以資源為基礎的政策、身分型政策、IAM 許可界限和工作階段政策。在所有這些政策中，強制執行程式碼會尋找套用到請求的 Deny 陳述式。此稱為明確拒絕。如果強制執行程式碼找到一個適用的明確拒絕，則程式碼會傳回 拒絕 的最後決定。如果沒有明確拒絕，該強制執行程式碼評估會持續執行。
2. Organ@@ izations SCP — 然後強制執行程式碼會評估適用於要求的 AWS Organizations 服務控制原則 (SCP)。SCP 會套用至連接 SCP 的帳戶之主體。如果強制執行程式碼在 SCP 中找不到任何適用的 `Allow` 陳述式，請求便是被明顯拒絕，即使拒絕是隱含的。強制執行程式碼傳回拒絕的最後決定。如果 SCP 不存在，或是 SCP 允許請求動作，強制執行程式碼評估就會繼續執行。
3. 資源型政策 – 在同一帳戶內，資源型政策會根據存取資源的主體類型以及資源型政策中允許的主體，以不同的方式影響政策評估。根據主體類型，資源型政策中的 `Allow` 可能會導致 `Allow` 的最終決定，即使身分型識別政策、許可界限或工作階段政策中有隱含拒絕。

針對大部分資源，您只需要使用身分型政策或資源型政策明確允許主體，授予存取權即可。IAM 角色信任政策和 KMS 金鑰政策是此邏輯的例外，因為這些政策必須明確允許存取主體。

如果指定的主體是 IAM 使用者、IAM 角色或工作階段主體，資源型政策的邏輯就會與其他政策類型有所不同。工作階段主體包括 IAM角色工作階段或 IAM 聯合身分使用者工作階段。如果資源型政策直接將許可授予 IAM 使用者或提出要求的工作階段主體，則身分識別型政策、許可界限或工作階段政策中的隱含拒絕不會影響最終決定。

下表可協助您瞭解當身分識別型政策、許可界限和工作階段政策中出現隱含拒絕時，資源型政策會針對不同主體類型產生何種影響。

資源型政策和其他政策類型 (相同帳戶) 中的隱含拒絕

主體發出要求	以資源為基礎政策	身分型政策	許可界限	工作階段政策	結果	原因
IAM 角色	不適用	不適用	不適用	不適用	不適用	角色本身無法提出要求。擔任角色之後，系統會使用角色工作階段提出要求。

主體發出要求	以資源為基礎政策	身分型政策	許可界限	工作階段政策	結果	原因
IAM 角色工作階段	允許角色 ARN	隱含拒絕	隱含拒絕	隱含拒絕	拒絕	許可界限和工作階段政策會作為最終決定的一部分進行評估。任一政策中的隱含拒絕會導致「拒絕」決定。
IAM 角色工作階段	允許角色工作階段 ARN	隱含拒絕	隱含拒絕	隱含拒絕	允許	許可會直接授予工作階段。其他政策類型不會影響決定。
IAM 使用者	允許 IAM 使用者 ARN	隱含拒絕	隱含拒絕	不適用	允許	許可直接授予使用者。其他政策類型不會影響決定。

主體發出要求	以資源為基礎政策	身分型政策	許可界限	工作階段政策	結果	原因
IAM 聯合身分使用者 (GetFederationToken)	允許 IAM 使用者 ARN	隱含拒絕	隱含拒絕	隱含拒絕	拒絕	許可界限或工作階段政策中的隱含拒絕會導致「拒絕」。
IAM 聯合身分使用者 (GetFederationToken)	允許 IAM 聯合身分使用者工作階段 ARN	隱含拒絕	隱含拒絕	隱含拒絕	允許	許可會直接授予工作階段。其他政策類型不會影響決定。

主體發出要求	以資源為基礎政策	身分型政策	許可界限	工作階段政策	結果	原因
根使用者	允許根使用者 ARN	不適用	不適用	不適用	允許	根使用者可無限制地全面存取 AWS 帳戶的所有資源。若要了解如何控制對 AWS Organizations中帳戶的根使用者的存取權，請參閱 Organizations 使用者指南中的服務控制政策 (SCP)。
AWS 服務主體	允許 AWS 服務主體	不適用	不適用	不適用	允許	資源型政策直接將許可授予 AWS 服務主體時，其他政策類型不會影響決定。

- IAM 角色 – 將許可授予 IAM 角色 ARN 的資源型政策受限於許可界限或工作階段政策中隱含拒絕的限制。

  角色 ARN 範例

  ```
  arn:aws:iam::111122223333:role/examplerole
  ```

- IAM 角色工作階段 – 在同一帳戶內，將許可授予 IAM 角色工作階段 ARN 的資源型政策直接授予許可給擔任的角色工作階段。直接授予工作階段的許可不會受到身分識別型政策、許可界限或工作階段政策中隱含拒絕的限制。當您擔任角色並提出要求時，提出要求的主體是 IAM 角色工作階段 ARN，而不是角色本身的 ARN。

  角色工作階段 ARN 範例

  ```
  arn:aws:sts::111122223333:assumed-role/examplerole/examplerolesessionname
  ```

- IAM 使用者 – 在同一帳戶內，將許可授予 IAM 使用者 ARN (非聯合身分使用者工作階段) 的資源型政策不受身分識別型政策或許可界限中隱含拒絕的限制。

  IAM 使用者 ARN 範例

  ```
  arn:aws:iam::111122223333:user/exampleuser
  ```

- IAM 聯合身分使用者工作階段 – IAM 聯合身分使用者工作階段是透過呼叫 `GetFederationToken` 建立的工作階段。當聯合身分使用者提出要求時，提出要求的主體是聯合身分使用者 ARN，而不是聯合身分 IAM 使用者的 ARN。在相同的帳戶中，授予許可給聯合身分使用者 ARN 的資源型政策會直接將許可授予工作階段。直接授予工作階段的許可不會受到身分識別型政策、許可界限或工作階段政策中隱含拒絕的限制。

  但是，如果資源型政策將許可授予聯合身分 IAM 使用者的 ARN，則聯合身分使用者在工作階段期間提出的要求會受到許可界限或工作階段政策中隱含拒絕的限制。

  IAM 聯合身分使用者工作階段 ARN 範例

  ```
  arn:aws:sts::111122223333:federated-user/exampleuser
  ```

4. 以身分為基礎的政策 – 程式碼接著會檢查主體的以身分為基礎的政策。若為 IAM 使用者，這些政策會包含使用者政策，以及使用者所屬群組的政策。如果沒有任何身分型政策或其中的任何陳述式允

許該要求動作，則表示要求已遭隱含拒絕，且程式碼會傳回拒絕的最終決定。如果任何適用的身分型政策中有任一陳述式允許該要求動作，則程式碼會繼續執行。

5. IAM 許可界限 – 程式碼接著會檢查主體所使用的 IAM 實體是否有許可界限。如果用來設定許可界限的政策不允許該請求動作，則表示請求已遭隱含拒絕。程式碼傳回拒絕的最後決定。如果許可界限不存在，或是許可界限允許請求動作，程式碼就會繼續執行。
6. 工作階段政策 – 程式碼接著會檢查主體是否為工作階段主體。工作階段主體包括 IAM 角色工作階段或 IAM 聯合身分使用者工作階段。如果主體不是工作階段主體，則強制執行程式碼會傳回允許的最終決定。

   對於工作階段主體，程式碼檢查工作階段政策是否在要求中傳遞。您可以在使用 AWS CLI 或 AWS API 時傳遞工作階段政策，以取得角色或 IAM 聯合身分使用者的臨時登入資料。

   - 如果工作階段政策存在，且不允許該要求動作，則表示要求已遭隱含拒絕。程式碼傳回拒絕的最後決定。
   - 如果工作階段政策不存在，程式碼會檢查主體是否為角色工作階段。如果主體是角色工作階段，則要求為已允許。否則，要求會受到婉拒，程式碼則會傳回拒絕的最終決定。
   - 如果有工作階段政策，且政策允許該要求動作，則強制執行程式碼會傳回允許的最終決定。
7. Er ror — 如果 AWS 強制執行程式碼在評估期間的任何時候遇到錯誤，則會產生例外狀況並關閉。

以身分為基礎和以資源為基礎的政策的範例

最常見的政策類型是以身分為基礎的政策和以資源為基礎的政策。當要求存取資源時， AWS 會評估策略授與至少一個 [允許] 在同一帳號中的所有權限。所有政策中的明確拒絕都會覆寫該允許。

> **Important**
>
> 如果在相同帳戶中，身分型政策和資源型政策中的任意一項政策允許請求而另一項不允許，則請求仍將被允許。

假設 Carlos 有使用者名稱 `carlossalazar`，他嘗試儲存檔案到 `carlossalazar-logs` Amazon S3 儲存貯體。

另外，也假設以下政策已連接到 `carlossalazar` IAM 使用者。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowS3ListRead",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowS3Self",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::carlossalazar/*",
                "arn:aws:s3:::carlossalazar"
            ]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::*log*"
        }
    ]
}
```

此政策中的 AllowS3ListRead 陳述式允許 Carlos 檢視帳戶中的所有儲存貯體的清單。AllowS3Self 陳述式允許 Carlos 完整存取名稱和他的使用者名稱完全相同的儲存貯體。DenyS3Logs 陳述式拒絕 Carlos 存取名稱中包含 log 的任何 S3 儲存貯體。

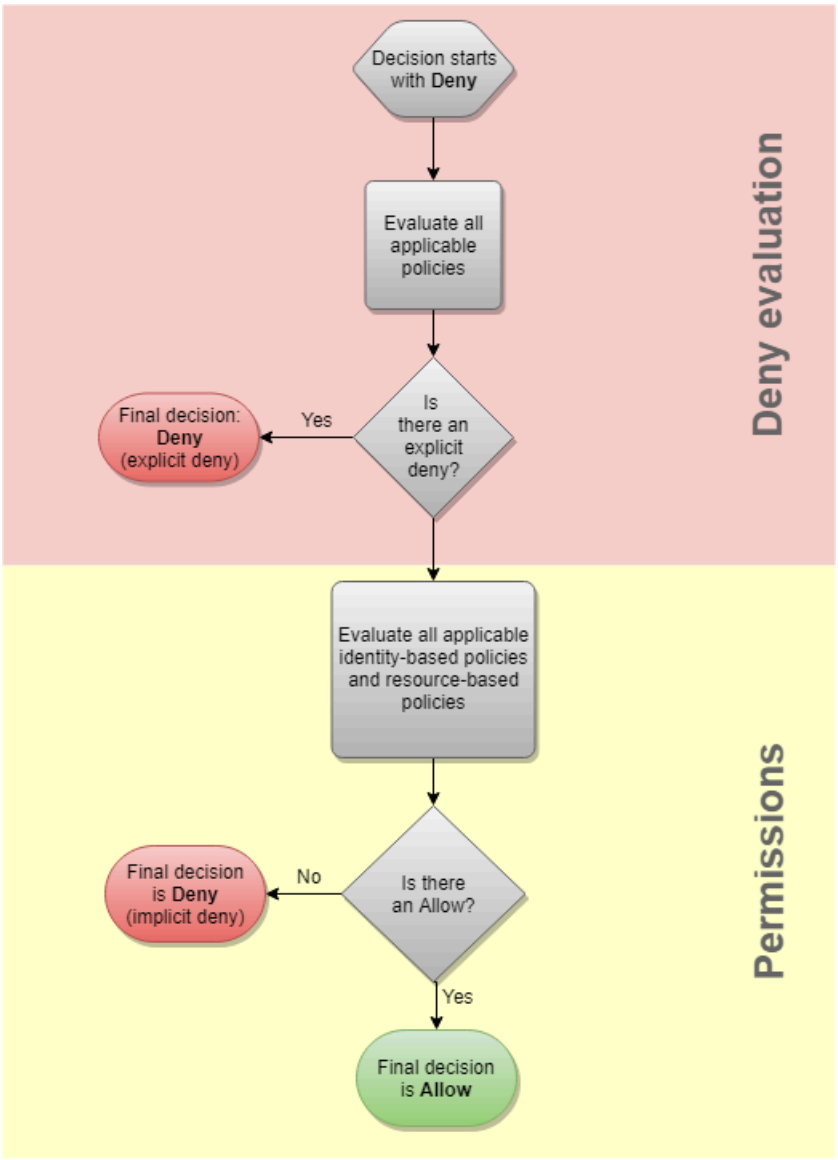
此外，以下以資源為基礎的政策 (稱為儲存貯體政策) 已連接到 carlossalazar 儲存貯體。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/carlossalazar"
            },
```

```
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::carlossalazar/*",
                "arn:aws:s3:::carlossalazar"
            ]
        }
    ]
}
```

此政策指定只有 carlossalazar 使用者可以存取 carlossalazar 儲存貯體。

當 Carlos 提出要求將檔案儲存至carlossalazar-logs值區時， AWS 會決定要求套用哪些原則。在這種情況下，只有以身分為基礎的政策和以資源為基礎的政策適用。這兩者都是許可政策。由於未套用許可界限，評估邏輯會降低到以下邏輯。



AWS 首先檢查適用於請求上下文的Deny語句。它找到一個，因為以身分為基礎的政策明確拒絕 Carlos 存取任何用於記錄的 S3 儲存貯體。Carlos 存取遭拒。

假設他然後意識到自己的錯誤，並試圖將文件保存到存儲carlossalazar桶。 AWS 檢查Deny陳述式，但找不到陳述式。接著檢查許可政策。身分類型政策和資源類型政策都允許請求。因此， AWS 允許請求。如果其中一個元素明確拒絕陳述式、請求會被拒絕。如果其中一種政策類型允許請求，但另一種則不允許，則仍會允許請求。

明確和隱含拒絕之間的差異

如果適用的請求包含 Deny 陳述式，請求會導致明確拒絕。如果套用到請求的政策包含 Allow 陳述式和 Deny 陳述式，則 Deny 陳述式勝過 Allow 陳述式。請求明確遭拒。

如果沒有適用的 Deny 陳述式，也沒有適用的 Allow 陳述式，則會發生暗中拒絕。由於預設拒絕存取 IAM 主體，他們必須明確獲允許執行動作。否則，會暗中拒絕他們存取。

當您設計授權策略時，您必須建立具 Allow 陳述式的政策，讓您的主體成功發出請求。不過，您可以選擇明確和暗中拒絕的任意組合。

例如，您可以建立下列政策，其中包含允許的動作、隱含拒絕的動作以及明確拒絕的動作。此 AllowGetList 陳述式允許對以字首 Get 和 List 開頭的 IAM 動作的唯讀存取。IAM 中的所有其他動作，例如 iam:CreatePolicy，則被隱含拒絕。此 DenyReports 陳述式明確拒絕存取 IAM 報告，方法是拒絕存取包含 Report 尾碼的動作，例如 iam:GetOrganizationsAccessReport。如果有人向此主體新增其他政策，以授予他們對 IAM 報告的存取權，例如 iam:GenerateCredentialReport，則由於存在此明確拒絕，報告相關的要求仍然被拒絕。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGetList",
            "Effect": "Allow",
            "Action": [
                "iam:Get*",
                "iam:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DenyReports",
            "Effect": "Deny",
```

```
            "Action": "iam:*Report",
            "Resource": "*"
        }
    ]
}
```

跨帳戶政策評估邏輯

您可以允許一個帳戶中的主體存取第二個帳戶中的資源。這稱為跨帳戶存取。當您允許跨帳戶存取時，主體所在的帳戶稱為受信任帳戶。資源所在的帳戶是信任帳戶。

若要允許跨帳戶存取，請將以資源為基礎的政策連接至您要共用的資源。您也必須將身分型政策連接至在請求中扮演主體的身分。信任帳戶中的資源型政策，必須指定具有資源存取權的受信任帳戶主體。您可以指定整個帳戶或其 IAM 使用者、聯合身分使用者、IAM 角色或擔任的角色工作階段。您也可以將 AWS 服務指定為主體。如需詳細資訊，請參閱 指定主體。

主體的以身分為基礎的政策，必須允許要求存取信任服務中的資源。作法是指定資源的 ARN，或允許存取所有資源 (*)。

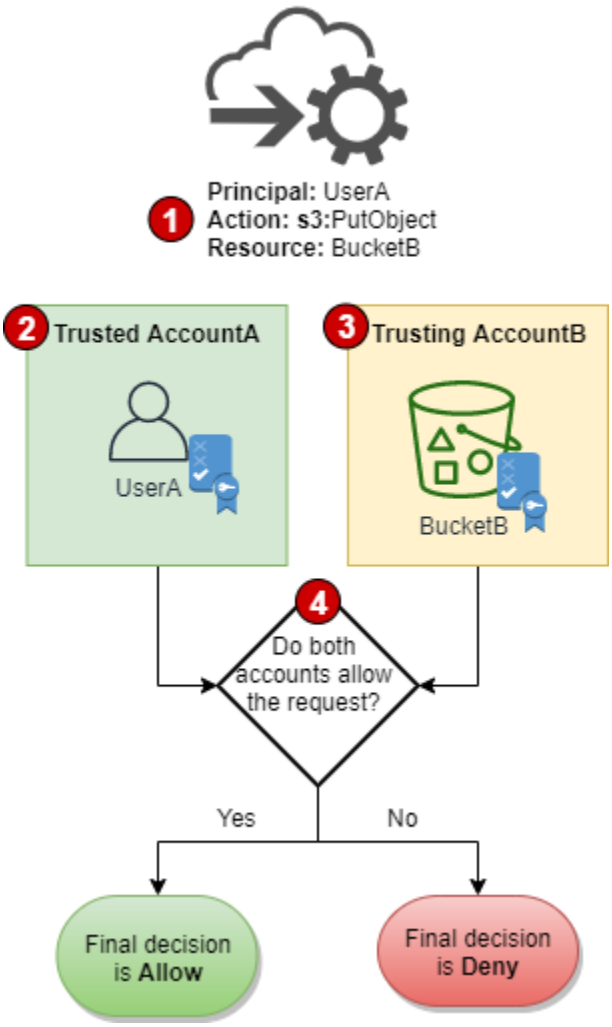
在 IAM 中，您可以將以資源為基礎的政策連接至 IAM 角色，以允許其他帳戶中的主體擔任該角色。角色的以資源為基礎的政策稱為角色信任政策。擔任該角色之後，允許的主體可以使用產生的臨時憑證，以存取您帳戶中的多個資源。此存取是在角色的以身分為基礎的許可政策中定義。若要了解使用角色允許跨帳戶存取，與使用其他以資源為基礎的政策允許跨帳戶存取有何不同，請參閱 IAM 中的跨帳戶資源存取。

> **Important**
>
> 其他服務會影響政策評估邏輯。例如， AWS Organizations 支援可套用至主體一或多個帳戶的服務控制原則。 AWS Resource Access Manager 支援政策片段，這些片段可控制允許主體對與其共用的資源執行哪些動作。

決定是否允許跨帳戶請求

對於跨帳戶請求，受信任 `AccountA` 中的請求者必須具有以身分為基礎的政策。該政策必須允許他們對信任 `AccountB` 中的資源提出請求。此外，`AccountB` 中以資源為基礎的政策，必須允許 `AccountA` 中的請求者存取資源。

當您提出跨帳戶請求時， AWS 會執行兩項評估。 AWS 評估信任帳戶和受信任帳戶中的請求。如需有關如何在單一帳戶內評估請求的詳細資訊，請參閱 決定是否允許或拒絕帳戶中的請求。只有當兩項評估都傳回 `Allow` 決定時，才允許此請求。



1. 當一個帳戶中的主體請求存取另一個帳戶中的資源時，就稱為跨帳戶請求。
2. 提出請求的主體存在於受信任帳戶中 (AccountA)。 AWS 評估此帳戶時會檢查以身分為基礎的政策，以及可限制以身分為基礎的政策的任何政策。如需詳細資訊，請參閱 運用單一帳戶評估政策。
3. 所請求的資源存在於信任帳戶中 (AccountB)。 AWS 評估此帳戶時會檢查連接至所請求資源的以資源為基礎的政策，以及可限制以資源為基礎的政策的任何政策。如需詳細資訊，請參閱 運用單一帳戶評估政策。
4. AWS 只有在兩個帳號策略評估都允許要求時，才允許請求。

跨帳戶政策評估範例

下列範例示範一個帳戶中的使用者由第二個帳戶中以資源為基礎的政策授予許可的案例。

假設 Carlos 是開發人員，而且他在帳戶 111111111111 中是名為 `carlossalazar` 的 IAM 使用者。他想將檔案儲存至帳戶 222222222222 中的 `Production-logs` Amazon S3 儲存貯體。

另外，也假設以下政策已連接到 carlossalazar IAM 使用者。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3ListRead",
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Sid": "AllowS3ProductionObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object*",
            "Resource": "arn:aws:s3:::Production/*"
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::*log*",
                "arn:aws:s3:::*log*/*"
            ]
        }
    ]
}
```

此政策中的 AllowS3ListRead 陳述式允許 Carlos 檢視 Amazon S3 中所有儲存貯體的清單。AllowS3ProductionObjectActions 陳述式允許 Carlos 完整存取 Production 儲存貯體中的物件。DenyS3Logs 陳述式拒絕 Carlos 存取名稱中包含 log 的任何 S3 儲存貯體。也拒絕存取這些儲存貯體中的所有物件。

此外，以下以資源為基礎的政策 (稱為儲存貯體政策) 已連接至帳戶 222222222222 中的 Production 儲存貯體。

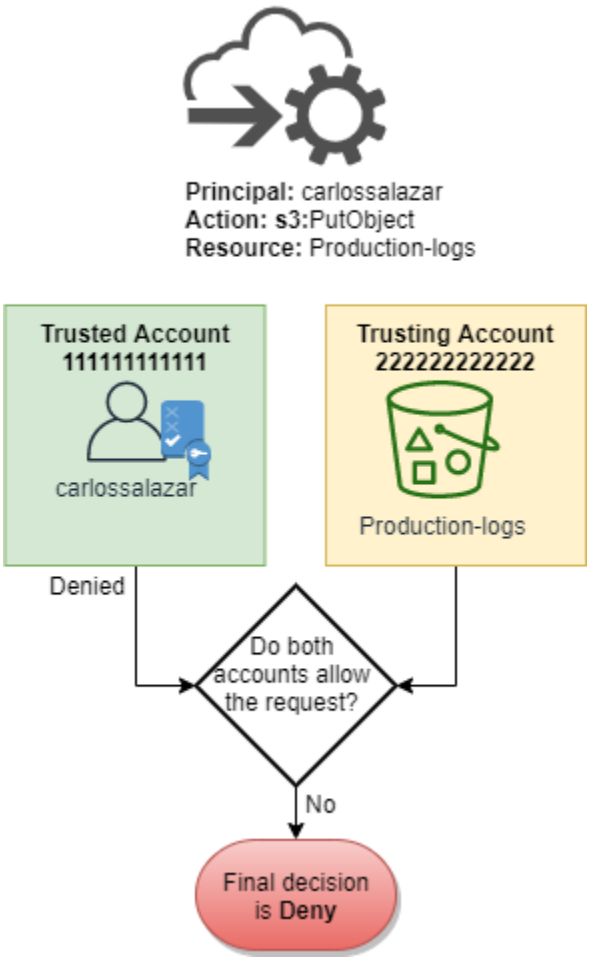
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject*",
                "s3:PutObject*",
                "s3:ReplicateObject",
                "s3:RestoreObject"
            ],
            "Principal": { "AWS": "arn:aws:iam::111111111111:user/carlossalazar" },
            "Resource": "arn:aws:s3:::Production/*"
        }
    ]
}
```

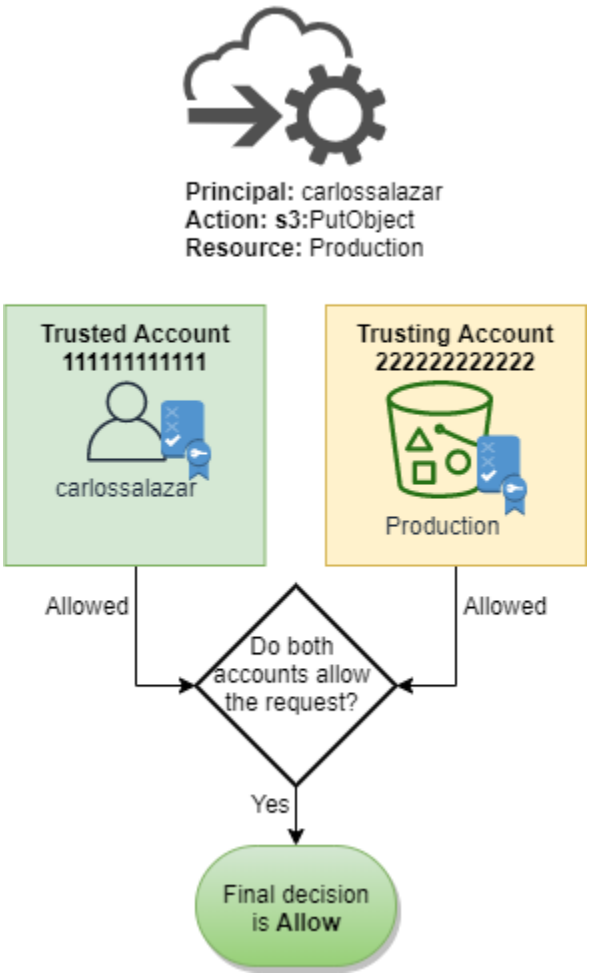
此政策允許 carlossalazar 使用者存取 Production 儲存貯體中的物件。其可建立與編輯，但不能刪除儲存貯體中的物件。其無法管理儲存貯體本身。

當 Carlos 提出請求將檔案儲存到 Production-logs 儲存貯體時， AWS 會決定套用至請求的政策。在此情況下，連接至 carlossalazar 使用者的以身分為基礎的政策，是帳戶 111111111111 中唯一套用的政策。在帳戶 222222222222 中，沒有任何以資源為基礎的政策連接至 Production-logs 儲存貯體。 AWS 評估帳戶 111111111111 時會傳回 Deny 的決定。這是因為以身分為基礎的政策中的 DenyS3Logs 陳述式，明確拒絕存取任何日誌儲存貯體。如需有關如何在單一帳戶內評估請求的詳細資訊，請參閱 決定是否允許或拒絕帳戶中的請求。

因為其中一個帳戶內明確拒絕請求，所以最終決定是拒絕請求。



假設 Carlos 然後意識到他的錯誤，並嘗試將文件保存到存儲Production桶。 AWS 首先檢查帳戶，111111111111以確定是否允許請求。只有以身分為基礎的原則適用，而且允許要求。 AWS 然後檢查帳戶222222222222。只有連接至 Production 儲存貯體的以資源為基礎的政策才會套用，並允許請求。因為這兩個帳戶都允許請求，所以最終決定是允許請求。



IAM JSON 政策語言的文法

此頁面介紹用來在 IAM 中建立 JSON 政策語言的正式文法。我們展示此文法，以便您可以了解如何建構和驗證政策。

如需政策的範例，請參閱下列主題：

- IAM 中的政策和許可
- 以身分為基礎的 IAM 政策範例
- 在 Amazon EC2 主控台中工作的範例政策，以及在 Amazon EC2 使用者指南中使用 CLI、Amazon EC2 CLI 或 AWS SDK 的範例政策。 AWS
- Amazon Simple Storage Service 使用者指南中的儲存貯體政策範例和使用者政策範例。

如需其他 AWS 服務中使用的政策範例，請前往這些服務的說明文件。

主題

政策語言和 JSON

以 JSON 表達的政策。當您建立或編輯 JSON 政策時，IAM 可以執行政策驗證以協助您建立有效的政策。IAM 會識別 JSON 語法錯誤，而 IAM Access Analyzer 會提供額外的政策檢查及建議，協助您進一步改良政策。若要進一步了解政策驗證的資訊，請參閱 驗證 IAM 政策。若要進一步了解 IAM Access Analyzer 政策檢查和可動作的建議，請參閱 IAM Access Analyzer 政策驗證。

在本文件中，我們不提供哪些資料構成有效 JSON 的完整說明。不過，以下是一些基本 JSON 規則：

- 個別實體間允許空格。
- 值以雙引號括住。雙引號對於數字和布林值為選用。
- 許多元素 (例如，`action_string_list` 和 `resource_string_list`) 可將 JSON 陣列當作值。陣列可以使用一個或多個值。如果包含多個值，陣列會在方括號 (`[` 和 `]`) 中並以逗號分隔，如以下範例所示：

  ```
  "Action" : ["ec2:Describe*","ec2:List*"]
  ```
- 基本 JSON 資料類型 (布林值、數字及字串) 是在 RFC 7159 中定義。

此文法中使用的慣例

此文法中使用以下慣例：

- 下列字元是 JSON 權杖，並且「包含」在政策中：

  ```
  { } [ ] " , :
  ```
- 下列字元在文法中為特殊字元，並且「不」包含在政策中：

  ```
  = < > ( ) |
  ```
- 如果某個元素允許多個值，它會使用重複值、逗號分隔符號和省略符號 (`...`) 來表示。範例：

  ```
  [<action_string>, <action_string>, ...]
  ```

  ```
  <principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }
  ```

如果允許多個值，只包含一個值也屬有效。對於僅有一個值，必須省略尾端逗號。如果元素使用陣列 (以 [和] 標註)，但只包含一個值，則括號為選用。範例：

```
"Action": [<action_string>]
```

```
"Action": <action_string>
```

- 某個元素後的問號 (?) 表示該元素為選用。範例：

```
<version_block?>
```

不過，有關選用元素的詳細資訊，請務必參照文法清單後的注意事項。

- 元素之間的垂直線 (|) 表示替代方案。在文法中，括號定義替代方案的範圍。範例：

```
("Principal" | "NotPrincipal")
```

- 必須為文字字串的元素會加上雙引號 (")。範例：

```
<version_block> = "Version" : ("2008-10-17" | "2012-10-17")
```

如需其他注意事項，請參閱文法清單後的政策文法備註。

文法

以下清單說明政策語言文法。如需清單中所使用的慣例，請參閱上一節。如需其他資訊，請參閱後續的注意事項。

> **Note**
>
> 此文法描述以版本 2008-10-17 和 2012-10-17 標記的政策。Version 政策元素與政策版本不同。Version 政策元素是在政策內使用，並定義政策語言的版本。另一方面，政策版本會在您在 IAM 中變更客戶受管政策時建立。變更的政策不會覆寫現有的政策。IAM 反而會建立新版本的受管政策。若要進一步了解 Version 政策元素，請參閱 IAM JSON 政策元素：Version。若要進一步了解政策版本，請參閱 the section called “版本控制 IAM 政策”。

```
policy  = {
     <version_block?>
     <id_block?>
     <statement_block>
}
```

```
<version_block> = "Version" : ("2008-10-17" | "2012-10-17")

<id_block> = "Id" : <policy_id_string>

<statement_block> = "Statement" : [ <statement>, <statement>, ... ]

<statement> = {
    <sid_block?>,
    <principal_block?>,
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block?>
}

<sid_block> = "Sid" : <sid_string>

<effect_block> = "Effect" : ("Allow" | "Deny")

<principal_block> = ("Principal" | "NotPrincipal") : ("*" | <principal_map>)

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }

<principal_map_entry> = ("AWS" | "Federated" | "Service" | "CanonicalUser") :
    [<principal_id_string>, <principal_id_string>, ...]

<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])

<resource_block> = ("Resource" | "NotResource") :
    : ("*" | <resource_string> | [<resource_string>, <resource_string>, ...])

<condition_block> = "Condition" : { <condition_map> }
<condition_map> = {
  <condition_type_string> : { <condition_key_string> : <condition_value_list> },
  <condition_type_string> : { <condition_key_string> : <condition_value_list> }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
<condition_value> = (<condition_value_string> | <condition_value_string> |
 <condition_value_string>)
```

政策文法備註

- 單一政策可包含一系列的陳述式。
- 政策大小上限為 2048 個字元和 10,240 個字元，取決於政策所連接的實體。如需詳細資訊，請參閱 IAM 和 AWS STS 配額。政策大小的計算不包含空格字元。
- 個別元素不得包含相同索引鍵的多個執行個體。例如，您不能在同一個陳述式中包含兩次 `Effect` 區塊。
- 區塊可以任何順序顯示。例如，政策中 `version_block` 可以在 `id_block` 之後。同樣地，`effect_block`、`principal_block`、`action_block` 可以在陳述式內以任何順序顯示。
- `id_block` 在以資源為基礎的政策中為選用。身分類型政策「不」得包含它。
- 以資源為基礎的政策和 IAM 角色的信任政策需要 `principal_block` 元素 (例如，在 Amazon S3 儲存貯體政策中)。身分類型政策「不」得包含它。
- Amazon S3 儲存貯體政策中的 `principal_map` 元素可以包含 `CanonicalUser` ID。大多數以資源為基礎的政策不支援此映射。若要進一步了解有關在儲存貯體政策中使用正式使用者 ID，請參閱 Amazon Simple Storage Service 使用者指南中的在政策中指定主體。
- 每個字串值 (`policy_id_string`、`sid_string`、`principal_id_string`、`action_string`、`resource_stri` 和字串版本的 `condition_value`) 可以有自己的最低和最大長度限制、特定允許的值，或是必要的內部格式。

有關字串值的備註

本節提供有關在政策的不同元素中使用字串值的其他資訊。

`action_string`

包含服務命名空間、冒號和動作的名稱。動作名稱可以包含萬用字元。範例：

```
"Action":"ec2:StartInstances"

"Action":[
  "ec2:StartInstances",
  "ec2:StopInstances"
]

"Action":"cloudformation:*"

"Action":"*"
```

```
"Action":[
  "s3:Get*",
  "s3:List*"
]
```

policy_id_string

提供方式以包含有關政策的資訊。有些服務 (例如 Amazon SQS 和 Amazon SNS) 在預留方式中使用 Id 元素。除非按個別服務進行限制，否則 policy_id_string 可以包含空格。有些服務需要此值在 AWS 帳戶中為唯一。

> **Note**
>
> 以資源為基礎的政策中允許 id_block，但是以身分為基礎的政策不允許。

長度並沒有限制，雖然此字串對政策的整體長度有影響，但是有限的。

```
"Id":"Admin_Policy"

"Id":"cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

sid_string

提供方式以包含有關個別陳述式的資訊。對於 IAM 政策，Sid 值只允許使用基本英數字元 (A-Z、a-z、0-9)。支援資源政策的其他 AWS 服務可能對 Sid 值有其他要求。例如，某些服務要求此值在 a 中是唯一的 AWS 帳戶，而某些服務允許其他字元，例如Sid值中的空格。

```
"Sid":"1"

"Sid": "ThisStatementProvidesPermissionsForConsoleAccess"
```

principal_id_string

提供使用 Amazon 資源名稱 (ARN)、IAM 使用者 AWS 帳戶、IAM 角色、聯合身分使用者或指定角色使用者來指定主體的方法。對於一個 AWS 帳戶，你也可以使用簡短的形式，AWS:*accountnumber*而不是完整的 ARN。如需包含 AWS 服務、擔任角色等等所有選項，請參閱指定主體。

請注意，您只能使用 * 來指定「每個人/匿名」。您不能使用它來指定 ARN 的部分名稱。

resource_string

在大部分情況下，包含 Amazon Resource Name (ARN)。

```
"Resource":"arn:aws:iam::123456789012:user/Bob"

"Resource":"arn:aws:s3:::examplebucket/*"
```

condition_type_string

識別要測試條件的類型，例如 StringEquals、StringLike、NumericLessThan、DateGreaterThanEquals、Bool、BinaryE 等。如需資料類型的完整清單，請參閱 IAM JSON 政策元素：條件運算子。

```
"Condition": {
  "NumericLessThanEquals": {
    "s3:max-keys": "10"
  }
}

"Condition": {
  "Bool": {
    "aws:SecureTransport": "true"
  }
}

"Condition": {
  "StringEquals": {
      "s3:x-amz-server-side-encryption": "AES256"
   }
}
```

condition_key_string

識別將測試其值以判斷是否符合條件的條件索引鍵。 AWS 定義一組可在所有 AWS 服務中使用的條件金鑰aws:PrincipalType，包括aws:SecureTransport、和aws:userid。

如需 AWS 條件索引鍵的清單，請參閱AWS 全域條件內容索引鍵。如需服務特定的條件金鑰，請參閱該服務的文件，例如下列：

- Amazon Simple Storage Service 使用者指南中的在政策中指定條件
- Amazon EC2 使用者指南中適用於 Amazon EC2 的 IAM 政策。

```
"Condition":{
  "Bool": {
      "aws:SecureTransport": "true"
   }
}

"Condition": {
  "StringNotEquals": {
      "s3:x-amz-server-side-encryption": "AES256"
   }
}

"Condition": {
  "StringEquals": {
    "aws:ResourceTag/purpose": "test"
  }
}
```

condition_value_string

識別決定是否符合條件的 condition_key_string 的值。如需條件類型有效值的完整清單，請參閱 IAM JSON 政策元素：條件運算子。

```
"Condition":{
  "ForAnyValue:StringEquals": {
  "dynamodb:Attributes": [
   "ID",
   "PostDateTime"
         ]
  }
}
```

AWS 受管理的工作職能政策

我們建議使用授予最低權限的政策，或僅授予執行任務所需的許可。授予最低權限的最安全方式是撰寫僅具有團隊所需許可的自訂政策。您必須建立程序，以允許您的團隊在必要時要求更多許可。建立 IAM 客戶受管政策需要時間和專業知識，而受管政策可為您的團隊提供其所需的許可。

若要開始將許可新增至您的 IAM 身分 (使用者、使用者群組和角色)，您可以使用AWS 受管理政策。AWS 受管政策涵蓋常見使用案例，並可在您的 AWS 帳戶. AWS 受管理的政策不會授與最低權限。若授予主體的許可超出執行任務所需的許可，您必須考量其相關的安全性風險。

您可以將 AWS 受管政策 (包括工作職能) 附加至任何 IAM 身分。若要切換到最低權限權限，您可以執行 AWS Identity and Access Management 存取分析器來監視具有 AWS 受管理原則的主體。了解他們正在使用哪些許可後，您可以撰寫自訂政策或產生僅具有團隊所需許可的政策。這不太安全，但是當您了解團隊的使用方式時，會提供更大的靈活性 AWS。

AWS 工作職能的管理政策旨在緊密配合 IT 行業的常見工作職能。您可以使用這些政策來授予執行特定任務角色中的某人所預期的任務所需的許可權。這些政策將許多服務的許可整合到單一政策，以便更輕鬆地使用分散在許多政策中的許可。

使用角色來結合服務

部分政策使用 IAM 服務角色來協助您充分利用其他 AWS 服務中的功能。這些策略授與存取權`iam:passrole`，允許具有該策略的使用者將角色傳遞給 AWS 服務。此角色會將 IAM 許可委派給 AWS 服務，以代表您執行動作。

您必須根據您的需求建立角色。例如，網路管理員政策允許具有該政策的使用者將名為「流程記錄-vpc」的角色傳遞給 Amazon 服務。 CloudWatch CloudWatch 使用該角色記錄和擷取使用者建立之 VPC 的 IP 流量。

為了遵守安全性最佳實務，工作職能的政策包含篩選條件，其會限制可傳遞的有效角色的名稱。這有助於避免授予不必要的許可。如果您的使用者確實選用的服務角色，則您必須建立遵照政策中指定的命名慣例的角色。然後，將許可授予角色。完成後，使用者就可以設定服務以使用角色，授予它角色提供的許可。

在以下區段，每個政策的名稱是連到 AWS Management Console中政策詳細資訊頁面的連結。您可以在那裡查看政策文件並檢閱其授予的許可。

管理員任務角色

AWS 受管理的策略名稱：AdministratorAccess

使用案例：此使用者具有完整的存取權，可委派許可給 AWS中的每個服務和資源。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

策略說明：此策略會授與帳號中所有 AWS 服務和所有資源的所有動作。如需有關受管理策略的詳細資訊，請參閱AdministratorAccess受AWS 管理策略參考指南中的。

> **Note**
>
> 您必須先啟用 IAM 使用者和角色存取權限，IAM 使用者或角色才能透過此政策中的許可存取 AWS Billing and Cost Management 主控台。若要執行此操作，請按照有關委派存取權至帳單主控台的教學的步驟 1 中的指示。

帳單任務角色

AWS 受管策略名稱：帳單

使用案例：此使用者需要檢視帳單資訊、設定付款和授權付款。用戶可以監控整個 AWS 服務累積的成本。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

政策描述：此政策授予完整許可，以管理帳單、成本、付款方法、預算和報告。如需其他成本管理政策範例，請參閱AWS Billing and Cost Management 使用者指南中的AWS Billing 政策範例。如需受管政策的詳細資訊，請參閱受AWS 管理政策參考指南中的計費。

> **Note**
>
> 您必須先啟用 IAM 使用者和角色存取權限，IAM 使用者或角色才能透過此政策中的許可存取 AWS Billing and Cost Management 主控台。若要執行此操作，請按照有關委派存取權至帳單主控台的教學的步驟 1 中的指示。

資料庫管理員任務角色

AWS 受管理的策略名稱：DatabaseAdministrator

使用案例：此使用者在 Cloud 中設定、設定和維護資料庫。 AWS

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

政策描述：此政策授予許可，以建立、設定和維護資料庫。它包括對 AWS 資料庫服務的存取，例如 Amazon DynamoDB、Amazon Relational Database Service (RDS) 和 Amazon Redshift。

查看政策以了解此政策支援之資料庫服務的完整清單。如需有關受管理策略的詳細資訊，請參閱DatabaseAdministrator受AWS 管理策略參考指南中的。

此工作職能原則支援將角色傳遞給 AWS 服務的能力。政策只針對下表中所命名的那些角色允許 `iam:PassRole` 動作。如需詳細資訊，請參閱本主題稍後的 建立角色和連接政策 (主控台)。

資料庫管理員工作職能的選用 IAM 任務角色

使用案例	角色名稱 (* 是萬用字元)	要選擇的服務角色類型	選取此 AWS 受管理的策略
允許使用者監控 RDS 資料庫	rds-monitoring-role	增強監控的 Amazon RDS 角色	亞馬遜 RDS 角 EnhancedMonitoring色
允許 AWS Lambda 監控您的數據庫並訪問外部數據庫	rdbms-lambda-access	Amazon EC2	AWSLambda_FullAccess
允許 Lambda 使用 DynamoDB 將檔案上傳至 Amazon S3 和 Amazon Redshift 叢集	lambda_exec_role	AWS Lambda	如 AWS 大數據部落格中所定義建立新的受管政策
允許 Lambda 函數作為您 DynamoDB 表格的觸發條件	lambda-dynamodb-*	AWS Lambda	AWSLambda DynamoDBExecutionRole
允許 Lambda 函數存取 VPC 中的 Amazon RDS	lambda-vpc-execution-role	如 AWS Lambda 開發人員指南 中所定義使用信任政策建立角色	AWSLambda VPCAccessExecutionRole
允許 AWS Data Pipeline 存取您的 AWS 資源	DataPipelineDefaultRole	如 AWS Data Pipeline 開發人員指南 中所定義使用信任政策建立角色	AWS Data Pipeline 文件列出此使用案例所需的權限。請參閱的 IAM 角色，以 AWS Data Pipeline

使用案例	角色名稱 (* 是萬用字元)	要選擇的服務角色類型	選取此 AWS 受管理的策略
允許在 Amazon EC2 執行個體上執行的應用程式存取您的 AWS 資源	DataPipelineDefaultResource角色	如 AWS Data Pipeline 開發人員指南 中所定義使用信任政策建立角色	亚马逊 RoleforDataPipelineRole

資料科學家任務角色

AWS 受管理的策略名稱：DataScientist

使用案例：此使用者執行 Hadoop 工作和查詢。使用者也會存取和分析資料分析和商業智慧的資訊。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

政策說明：此政策授予在 Amazon EMR 叢集上建立、管理和執行查詢的許可，以及使用 Amazon 等工具執行資料分析。 QuickSight該政策包括存取其他資料科學家服務 AWS Data Pipeline，例如 Amazon EC2、Amazon Kinesis、Amazon Machine Learning 和 SageMaker. 查看政策以了解此政策支援之資料科學家服務的完整清單。如需有關受管理策略的詳細資訊，請參閱DataScientist受AWS 管理策略參考指南中的。

此工作職能原則支援將角色傳遞給 AWS 服務的能力。一個語句允許傳遞任何角色 SageMaker。另外一個陳述式只針對下表中所命名的那些角色允許 `iam:PassRole` 動作。如需詳細資訊，請參閱本主題稍後的 建立角色和連接政策 (主控台)。

資料科學家任務職能的選用 IAM 服務角色

使用案例	角色名稱 (* 是萬用字元)	要選擇的服務角色類型	AWS 要選取的受管理原則
允許 Amazon EC2 執行個體存取適合叢集的服務和資源	電腦-EC2_DefaultRole	Amazon EMR for EC2	AmazonElasticMapReducefor角色

使用案例	角色名稱 (* 是萬用字元)	要選擇的服務角色類型	AWS 要選取的受管理原則
允許 Amazon EMR 存取適用於叢集的 Amazon EC2 服務和資源	電磁鐵 _ DefaultRole	Amazon EMR	亞馬遜電子ServicePolicy產品 _v2
允許 Kinesis Managed Service for Apache Flink 存取串流資料來源	kinesis-*	如 AWS 大數據部落格中所定義使用信任政策建立角色。	請參閱 AWS 大數據部落格，其根據您的使用案例概述四個可能的選項
允許 AWS Data Pipeline 存取您的 AWS 資源	DataPipelineDefaultRole	如 AWS Data Pipeline 開發人員指南 中所定義使用信任政策建立角色	AWS Data Pipeline 文件列出此使用案例所需的權限。請參閱的 IAM 角色，以 AWS Data Pipeline
允許在 Amazon EC2 執行個體上執行的應用程式存取您的 AWS 資源	DataPipelineDefaultResource角色	如 AWS Data Pipeline 開發人員指南 中所定義使用信任政策建立角色	亚马逊 RoleforDataPipelineRole

開發人員進階使用者任務角色

AWS 受管理的策略名稱：PowerUser存取

使用案例：此使用者執行應用程式開發工作，並可建立和設定支援 AWS 感知應用程式開發的資源和服務。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

原則說明：此原則的第一個陳述式會使用`NotAction`元素來允許所有 AWS 服務及除 AWS Identity and Access Management、 AWS Organizations和以外的所有資源執行所有動作 AWS Account Management。第二個陳述式則會授予可建立服務連結角色的 IAM 許可。如果某些服務必須存取其他

服務中的資源 (例如 Amazon S3 儲存貯體)，則這是必要的。它也會授予可檢視使用者組織相關資訊的 Organizations 許可，包括管理帳戶電子郵件和組織限制。雖然此政策會限制 IAM、Organizations，但如果 IAM Identity Center 已啟用，則允許使用者執行所有 IAM Identity Center 動作。它也會授與帳戶管理權限，以檢視帳戶已啟用或停用哪些 AWS 區域。

網路管理員任務角色

AWS 受管理的策略名稱：NetworkAdministrator

使用案例：此使用者的任務是設定和維護 AWS 網路資源。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

政策說明：此政策授予在 Auto Scaling、Amazon EC2、路線 53、亞馬遜 AWS Direct Connect、Elastic Load Balancing CloudFront、Amazon SNS、 CloudWatch 日誌、Amazon S3 AWS Elastic Beanstalk、IAM 和 Amazon Virtual Private Cloud 中建立和維護網路資源的許可。 CloudWatch 如需有關受管理策略的詳細資訊，請參閱NetworkAdministrator受AWS 管理策略參考指南中的。

此工作功能需要將角色傳遞給 AWS 服務的能力。政策只針對下表中所命名的那些角色授予 `iam:GetRole` 和 `iam:PassRole`：如需詳細資訊，請參閱本主題後面部分的 建立角色和連接政策 (主控台)。

網路管理員工作職能的選用 IAM 任務角色

使用案例	角色名稱 (* 是萬用字元)	要選擇的服務角色類型	AWS 要選取的受管理原則
允許 Amazon VPC 代表使用者在日 CloudWatch 誌中建立和管理日誌，以監控進出 VPC 的 IP 流量	flow-logs-*	如 Amazon VPC 使用者指南 中所定義使用信任政策建立角色	此使用案例沒有現有的 AWS 受管理策略，但文件會列出必要的權限。請參閱 Amazon VPC 使用者指南。

唯讀存取

AWS 受管理的策略名稱：ReadOnly存取

使用案例:此使用者需要對 AWS 帳戶中的每個資源進行唯讀存取。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

政策說明：此政策會授予列出、取得、描述及檢視資源及其屬性的許可。不包括像建立或刪除變動職能。此原則確實包含對安全性相關 AWS 服務的唯讀存取權，例如 AWS Identity and Access Management 和。 AWS Billing and Cost Management查看政策以了解此政策支援之服務和動作的完整清單。

安全稽核人員任務角色

AWS 受管理的策略名稱：SecurityAudit

使用案例：此使用者監控帳戶是否符合安全需求。這個使用者可以存取日誌和事件，以調查潛在安全漏洞或潛在惡意活動。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

原則說明：此原則授與檢視許多 AWS 服務之組態資料及檢閱其記錄檔的權限。如需有關受管理策略的詳細資訊，請參閱SecurityAudit受AWS 管理策略參考指南中的。

支援使用者任務角色

AWS 受管理的策略名稱：SupportUser

使用案例：此使用者連絡 Sup AWS port 部門、建立支援案例，並檢視現有案例的狀態。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

原則說明：此原則授與建立和更新 AWS Support 案例的權限。如需有關受管理策略的詳細資訊，請參閱SupportUser受AWS 管理策略參考指南中的。

系統管理員任務角色

AWS 受管理的策略名稱：SystemAdministrator

使用案例：此使用者為開發操作設定和維護資源。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

政策說明：此政策授予跨多種 AWS 服務 (包括 Amazon、、、、、、、 AWS CloudTrail、、、CloudWatch、、 AWS CodeCommit、 AWS CodeDeploy、、 AWS Config、 AWS Directory Service、、、、、 AWS Identity and Access Management、、 AWS Key Management Service、AWS Lambda、、、、、、Amazon RDS、路線 53、Amazon S3、Amazon SQS 和 Amazon VPC) 的許可。 AWS Trusted Advisor如需有關受管理策略的詳細資訊，請參閱SystemAdministrator受AWS 管理策略參考指南中的。

此工作功能需要將角色傳遞給 AWS 服務的能力。政策只針對下表中所命名的那些角色授予 `iam:GetRole` 和 `iam:PassRole`：如需詳細資訊，請參閱本主題後面部分的 建立角色和連接政策 (主控台)。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

系統管理員工作職能的選用 IAM 任務角色

使用案例	角色名稱 (* 是萬用字元)	要選擇的服務角色類型	AWS 要選取的受管理原則
允許在 Amazon ECS 叢集的 EC2 執行個體中執行的應用程式存取 Amazon ECS	ecr-sysadmin-*	EC2 容器服務的 Amazon EC2 角色	亚马逊角ContainerServicefor色
允許使用者監控資料庫	rds-monitoring-role	增強監控的 Amazon RDS 角色	亞馬遜 RDS 角 EnhancedMonitoring色
允許在 EC2 執行個體中執行的應用程式存取 AWS 資源。	ec2-sysadmin-*	Amazon EC2	授予 S3 儲存貯體存取權的角色範例政策，如 Amazon EC2 使用者指南所示；視需要自訂
允許 Lambda 讀取 DynamoDB 資料流並寫入日誌 CloudWatch	lambda-sysadmin-*	AWS Lambda	AWSLambda DynamoDBExecutionRole

僅限檢視使用者任務角色

AWS 受管理的策略名稱：ViewOnly存取

使用案例：此使用者可以跨服務檢視帳戶中的 AWS 資源清單和基本中繼資料。使用者無法讀取超出配額的資源內容或中繼資料，以及列出資源的資訊。

策略更新： AWS 維護和更新此策略。如需此政策的變更記錄，請在 IAM 主控台中檢視政策，然後選擇 Policy versions (政策版本) 索引標籤。如需有關任務角色政策更新的詳細資訊，請參閱 更新工作職能的 AWS 受管理原則。

原則說明：此原則會授`List*`與 AWS 服務資源`Get*`、`View*`、、和`Lookup*`存取權。`Describe*`若要查看此原則針對每個服務包含哪些動作，請參閱ViewOnly存取。如需受管理原則的詳細資訊，請參閱AWS 受管理原則參考指南中的ViewOnly存取。

更新工作職能的 AWS 受管理原則

這些政策都由維護 AWS 並保持最新狀態，以包括對新服務和新功能的支持，因為它們是由 AWS 服務添加。這些政策無法由客戶修改。您可以製作政策的副本，然後修改副本，但該副本不會隨著 AWS 引入新的服務和 API 操作而自動更新。

對於任務角色政策，您可以在 IAM 主控台中檢視版本歷程記錄，以及每次更新的時間和日期。若要執行此操作，請使用此頁面上的連結來檢視政策詳細資訊。然後選擇 Policy versions (政策版本) 索引標籤以檢視版本。此頁面會顯示政策的最近 25 個版本。若要檢視原則的所有版本，請呼叫取得原則版本 AWS CLI 命令或版本 API 作業。GetPolicy

> **Note**
>
> 您最多可以有五個版本的客戶管理策略，但 AWS 保留受 AWS 管策略的完整版本歷史記錄。

建立角色和連接政策 (主控台)

先前列出的幾個原則授與使用角色來設定 AWS 服務的能力，讓這些服務代表您執行作業。工作職能政策會指定確切的角色名稱，您必須使用或至少包含字首，其指定可用名稱的第一個部分。若要建立這些角色之一，請執行下列程序中的步驟。

若要建立 AWS 服務 (IAM 主控台) 的角色

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

2. 在 IAM 主控台的導覽窗格中，選擇角色，然後選擇建立角色。
3. 對於 Trusted entity type (信任的實體類型)，請選擇 AWS 服務。
4. 對於服務或使用案例，請選擇服務，然後選擇使用案例。服務會定義使用案例，以包含服務所需的信任政策。
5. 選擇下一步。
6. 對於權限原則，選項取決於您選取的使用案例：
   - 如果服務定義了角色的權限，您就無法選取權限原則。
   - 從一組有限的權限原則中進行選取。
   - 從所有權限原則中選取。
   - 選取 [無權限原則]，建立角色後建立原則，然後將原則附加至角色。
7. (選用) 設定許可界限。這是進階功能，可用於服務角色，而不是服務連結的角色。
   a. 開啟 [設定權限界限] 區段，然後選擇 [使用權限界限] 控制最大角色權限。

      IAM 在您的帳戶中包含受 AWS 管政策和客戶管理政策的清單。
   b. 選取用於許可界限的政策。
8. 選擇下一步。
9. 對於角色名稱，選項取決於服務：
   - 如果服務定義了角色名稱，您就無法編輯角色名稱。
   - 如果服務定義了角色名稱的前置詞，您可以輸入選擇性的尾碼。
   - 如果服務未定義角色名稱，您可以命名角色。

   > **Important**
   >
   > 命名角色時，請注意下列事項：
   >
   > - 角色名稱在您的內部必須是唯一的 AWS 帳戶，並且不能根據大小寫將其唯一。
   >
   >   例如，請勿建立同時命名為**PRODROLE**和的角色**prodrole**。當角色名稱用於策略中或作為 ARN 的一部分時，角色名稱會區分大小寫，但是當主控台中的客戶 (例如在登入程序期間) 顯示角色名稱時，角色名稱不區分大小寫。
   > - 您無法在建立角色之後編輯該角色的名稱，因為其他實體可能會參照該角色。

10. (選擇性) 在說明中，輸入角色的說明。

11. (選擇性) 若要編輯角色的使用案例和權限，請在步驟 1：選取受信任的實體或步驟 2：新增權限區段中，選擇編輯。
12. (選擇性) 若要協助識別、組織或搜尋角色，請將標籤新增為鍵值配對。如需有關在 IAM 中使用標籤的詳細資訊，請參閱《IAM 使用者指南》中的標記 IAM 資源。
13. 檢閱角色，然後選擇 Create role (建立角色)。

範例 1：設定使用者做為資料庫管理員 (主控台)

此範例顯示設定 IAM 使用者 Alice 作為資料庫管理員所需的步驟。您使用該區段中表格第一列的資訊，並允許使用者啟用 Amazon RDS 監控。您可以將DatabaseAdministrator政策附加到愛麗絲的 IAM 使用者，以便他們可以管理 Amazon 資料庫服務。該政策也允許 Alice 將稱為 `rds-monitoring-role` 的角色傳遞到 Amazon RDS 服務，讓該服務代表他們來監控 Amazon RDS 資料庫。

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 選擇政策，在搜尋方塊中輸入 **`database`**，然後按下 Enter 鍵。
3. 選取DatabaseAdministrator原則的圓鈕，選擇 [動作]，然後選擇 [附加]。
4. 在實體清單中，選取 Alice，然後選擇連接政策。愛麗絲現在可以管理 AWS 數據庫。不過，為允許 Alice 來監控這些資料庫，您必須設定服務角色。
5. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
6. 選擇 AWS Service (服務) 角色類型，然後選擇 Amazon RDS。
7. 選擇 Amazon RDS Role for Enhanced Monitoring (增強監控的 Amazon RDS 角色) 使用案例。
8. Amazon RDS 定義角色的許可。選擇 Next: Review (下一步：檢閱) 以繼續進行。
9. 角色名稱必須是 Alice 現在擁有的 DatabaseAdministrator 原則所指定的名稱之一。其中一個是 **`rds-monitoring-role`**。為 Role name (角色名稱) 輸入該名稱。
10. (選用) 在 Role description (角色說明) 中，輸入新角色的說明。
11. 檢閱詳細資訊之後，選擇 Create role (建立角色)。
12. Alice 現在可以在 Amazon RDS 主控台的 Monitoring (監控) 區段中啟用 RDS Enhanced Monitoring (RDS 增強型監控)。例如，他們在建立資料庫執行個體、建立讀取複本或修改資料庫執行個體時，他們可能執行此操作。將 [啟用增強型監視] 設為 [是] 時，必須在 [監視角色] 方塊中輸入自己建立的角色名稱 (rds-monitoring-role)。

範例 2：設定使用者做為網路管理員 (主控台)

此範例顯示設定 IAM 使用者 Jorge 作為網路管理員所需的步驟。它使用該區段中表格的資訊，允許 Jorge 監控進出 VPC 的 IP 流量。它還允許豪爾赫 (Jorge) 在日 CloudWatch 誌中捕獲該信息。您可以將NetworkAdministrator政策附加到 Jorge 的 IAM 使用者，以便他們可以設定 AWS 網路資源。該政策也會在您建立流程日誌時，讓 Jorge 將名稱開頭為 `flow-logs*` 的角色傳遞到 Amazon EC2。在這個案例中，不像範例 1，沒有預先定義的服務角色類型，所以您必須分別地執行幾個步驟。

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇政策，然後在搜尋方塊中輸入 **`network`**，再按下 Enter。
3. 選取NetworkAdministrator策略旁邊的圓鈕，選擇 [動作]，然後選擇 [附加]。
4. 在使用者清單中，選取 Jorge 旁的核取方塊，然後選擇 Attach policy (連接政策)。豪爾赫現在可以管理 AWS 網路資源。不過，為啟用監控 VPC 中的 IP 流量，您必須設定服務角色。
5. 由於您需要建立的服務角色，沒有預先定義的受管政策，您必須先建立它。在導覽窗格中，選擇 Policies (政策)，然後選擇 Create policy (建立政策)。
6. 在政策編輯器區段中，選擇 JSON 選項，並從下列 JSON 政策文件複製文字。將此文字貼上至 JSON 文字方框中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

7. 解決政策驗證期間產生的任何安全性警告、錯誤或一般性警告，然後選擇 Next (下一步)。

> **Note**
>
> 您可以隨時切換視覺化與 JSON 編輯器選項。不過，如果您進行變更或在視覺化編輯器中選擇下一步，IAM 就可能會調整您的政策結構，以便針對視覺化編輯器進行最佳化。如需詳細資訊，請參閱 政策結構調整。

8. 在檢視與建立頁面上，針對政策名稱輸入 **`vpc-flow-logs-policy-for-service-role`**。檢視此政策中定義的許可以查看政策授與的許可，然後選擇建立政策來儲存您的工作。

   新的政策會出現在受管政策清單中，並且已準備好連接。

9. 在 IAM 主控台的導覽窗格中，選擇 Roles (角色)，然後選擇 Create role (建立角色)。
10. 選擇 AWS Service (服務) 角色類型，然後選擇 Amazon EC2。
11. 選擇 Amazon EC2 使用案例。
12. 在 [附加權限原則] 頁面上，選擇您先前建立的原則、vpc-flow-logs-policy-for-service-role，然後選擇 [下一步：檢閱]。
13. 角色名稱必須由豪爾赫現在擁有的 NetworkAdministrator 策略允許。任何以 `flow-logs-` 開頭的名稱均允許。在此範例中，為 Role name (角色名稱) 輸入 **`flow-logs-for-jorge`**。
14. (選用) 在 Role description (角色說明) 中，輸入新角色的說明。
15. 檢閱詳細資訊之後，選擇 Create role (建立角色)。
16. 現在您可以設定此案例所需的信任政策。在 [角色] 頁面上，選擇flow-logs-for-jorge角色 (名稱，而不是核取方塊)。針對您的新角色，在詳細資訊頁面上，選擇 Trust relationships (信任關係) 標籤，然後選擇 Edit trust relationship (編輯信任關係)。
17. 將「Service (服務)」行讀取為如下所示，取代 `ec2.amazonaws.com` 的項目：

    ```
            "Service": "vpc-flow-logs.amazonaws.com"
    ```

18. Jorge 現在可在 Amazon EC2 主控台中為 VPC 或子網路建立流程日誌。建立流程記錄檔時，請指定flow-logs-for-jorge角色。該角色具有許可，以建立日誌和寫入資料。

AWS 全域條件內容索引鍵

當主體向其提出要求時 AWS，會將要求資訊 AWS 收集到要求前後關聯中。您可以使用 JSON 政策的 `Condition` 元素，來比較請求內容中的鍵和您在政策中指定的鍵值。請求信息由不同的來源提供，包括發出請求的主體，提出請求的資源以及有關請求本身的元數據。

全域條件金鑰可用於所有 AWS 服務。雖然這些條件索引鍵可用於所有原則，但並非每個要求內容中都可使用該金鑰。例如，只有當AWS 服務主體直接呼叫資源時，aws:SourceAccount條件索引鍵才可用。若要深入瞭解要求內容中包含全域金鑰的情況，請參閱每個索引鍵的可用性資訊。

某些個別服務會建立自己的條件金鑰，這些金鑰可在其他服務的要求內容中使用。跨服務條件索引鍵是一種全域條件金鑰類型，其中包含與服務名稱相符的前置詞，例如ec2:或lambda:，但可跨其他服務使用。

服務特定的條件金鑰已定義為與個別 AWS 服務搭配使用。例如，Amazon S3 可讓您使用s3:VersionId條件金鑰撰寫政策，以限制對特定版本的 Amazon S3 物件的存取。此條件金鑰對於服務而言是唯一的，這表示僅適用於 Amazon S3 服務的請求。如需服務特定的條件金鑰，請參閱AWS 服務的動作、資源和條件金鑰，然後選擇您要檢視其金鑰的服務。

> **Note**
>
> 如果您使用僅在某些情況下可用的條件鍵，則可以使用條件運算子的IfExists版本。如果請求內容中缺少條件鍵，則政策可能會讓評估失敗。例如，搭配使用以下條件區塊與 ...IfExists 運算子，以在請求來自特定 IP 範圍或特定 VPC 時進行比對。若其中一個鍵，或兩者都不包含在請求內容中，條件仍會傳回 true。只有在請求內容中包含指定鍵時，才會檢查值。如需當其他運算子沒有索引鍵時如何評估原則的詳細資訊，請參閱條件運算子。
>
> ```
> "Condition": {
>     "IpAddressIfExists": {"aws:SourceIp" : ["xxx"] },
>     "StringEqualsIfExists" : {"aws:SourceVpc" : ["yyy"]}
> }
> ```

> **Important**
>
> 若要將您的條件與具有多個鍵值的請求內容進行比較，您必須使用 ForAllValues 或 ForAnyValue 設定運算子。只能將集合運算子與多值條件鍵搭配使用。請勿將集合運算子與單一值條件鍵搭配使用。如需詳細資訊，請參閱 多值內容索引鍵。

主體的屬性	角色工作階段的屬性	網絡的屬性	資源的屬性	請求的屬性
AWS：PrincipalArn AWS：PrincipalAccount AWS：PrincipalOrg路徑 AWS：PrincipalOrg身份證 AWS：PrincipalTag/標籤鍵 AWS：PrincipalIsAWSService AWS：PrincipalService名稱 AWS：PrincipalServiceNamesList AWS：PrincipalType aws:userid aws:username	AWS：FederatedProvider AWS：TokenIssue時間 AWS：MultiFactorAuthAge AWS：MultiFactorAuthPresent AWS: EC2虛擬私InstanceSource人雲端 AWS: EC2 私InstanceSource有 IPv4 AWS：SourceIdentity ec2：RoleDelivery ec2：SourceInstance阿恩 膠水：RoleAssumed通過 膠水：CredentialIssuing服務	AWS：SourceIp AWS：SourceVpc AWS：SourceVpce AWS：VpcSourceIP	AWS：ResourceAccount AWS：ResourceOrg路徑 AWS：ResourceOrg身份證 AWS：ResourceTag/標籤鍵	AWS：CalledVia AWS：CalledVia第一 AWS：CalledVia最後 AWS: 透過AWSService AWS：CurrentTime AWS：EpochTime aws:Referer AWS：RequestedRegion AWS：RequestTag/標籤鍵 AWS：TagKeys AWS：SecureTransport AWS：SourceArn AWS：SourceAccount

主體的屬性	角色工作階段的屬性	網絡的屬性	資源的屬性	請求的屬性
	拉姆達：SourceFunction阿恩 SSM：阿恩SourceInstance 身份存儲：UserId			AWS：SourceOrg路徑 AWS：SourceOrg身份證 AWS：UserAgent

主體的屬性

使用下列條件索引鍵，將提出要求之主體的詳細資訊與您在原則中指定的主參與者特性進行比較。如需可提出要求的主參與者清單，請參閱指定主體。

內容

- AWS：PrincipalArn
- AWS：PrincipalAccount
- AWS：PrincipalOrg路徑
- AWS：PrincipalOrg身份證
- AWS：PrincipalTag/標籤鍵
- AWS：PrincipalIsAWSService
- AWS：PrincipalService名稱
- AWS：PrincipalServiceNamesList
- AWS：PrincipalType
- aws:userid
- aws:username

AWS：PrincipalArn

使用此鍵來將提出請求主體的 Amazon Resource Name (ARN) 與您在政策中所指定的 ARN 進行比較。針對 IAM 角色，請求內容會傳回角色的 ARN，而非取得角色使用者的 ARN。

- 可用性 – 此鍵會包含在所有簽章請求的請求內容中。匿名請求不包含此鍵。您可以在此條件鍵中指定以下類型的主體：
  - IAM 角色
  - IAM 使用者
  - AWS STS 同盟使用者工作階段
  - AWS 帳戶 根使用者
- 數據類型-ARN，字符串

  AWS 建議您在比較 ARN 時使用 ARN 運算子而非字串運算子。
- 值類型 - 單一值
- 範例值下列清單顯示針對您可在`aws:PrincipalArn`條件索引鍵中指定的不同主參與者類型所傳回的請求前後關聯值：
  - IAM 角色 – 請求內容包含條件鍵 `aws:PrincipalArn` 的下列值。請勿將擔任角色工作階段 ARN 指定為此條件鍵的值。如需有關擔任角色工作階段主體的詳細資訊，請參閱 角色工作階段主體。

    ```
    arn:aws:iam::123456789012:role/role-name
    ```
  - IAM 使用者 – 請求內容包含條件鍵 `aws:PrincipalArn` 的下列值。

    ```
    arn:aws:iam::123456789012:user/user-name
    ```
  - AWS STS 聯合使用者工作階段 — 請求內容包含下列條件索引鍵`aws:PrincipalArn`值。

    ```
    arn:aws:sts::123456789012:federated-user/user-name
    ```
  - AWS 帳戶 root 使用者 — 請求內容包含下列條件索引鍵值`aws:PrincipalArn`。將根使用者 ARN 指定為 `aws:PrincipalArn` 條件鍵的值時，僅會限制 AWS 帳戶根使用者的許可。這不同於在資源型政策的主體元素中指定根使用者 ARN，後者將授權委派給 AWS 帳戶。如需在資源型政策的主體元素中指定根使用者 ARN 的相關資訊，請參閱 AWS 帳戶 校長。

    ```
    arn:aws:iam::123456789012:root
    ```

您可以指定根使用者 ARN 作為 AWS Organizations 服務控制策略 (SCP) `aws:PrincipalArn` 中條件索引鍵的值。SCP 是一種組織政策，用於管理您組織中的許可，並且僅會影響組織中的成員帳戶。SCP 會限制 IAM 使用者和成員帳戶中 IAM 使用者和角色的許可，包括成員帳戶的根使用者。如需 SCP 對許可的影響的詳細資訊，請參閱 Organizations 使用者指南中的 SCP 對許可的影響。

AWS：PrincipalAccount

使用此鍵來將請求主體所屬的帳戶與您在政策中指定的帳戶識別碼進行比較。對於匿名請求，請求內容會返回 anonymous。

- 可用性 – 此金鑰會包含在所有請求的請求內容中，包括匿名請求。
- 數據類型-字符串
- 值類型 - 單一值

以下範例中，除了帳號為 123456789012 的主體之外，一律拒絕存取。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:region:accountID:resource"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

AWS：PrincipalOrg路徑

使用此索引鍵可比較要求原則中 AWS Organizations 路徑的主參與者路徑的路徑。該主體可以是 IAM 使用者、IAM 角色、聯合身分使用者或 AWS 帳戶根使用者. 在政策中，這項條件鍵會確保申請者是在指定組織根或 AWS Organizations中組織單位 (OU) 內的帳戶成員。 AWS Organizations 路徑是組 Organizations 實體結構的文字表示。如需有關使用和了解路徑的詳細資訊，請參閱 了解 AWS Organizations 實體路徑。

- 可用性 – 只有在主體是組織的成員時，請求內容中才會包含此鍵。匿名請求不包含此鍵。
- 數據類型-字符串（列表）
- 值類型 - 多重值

> **Note**
>
> 組織 ID 是全域唯一，但 OU ID 和根 ID 只有在組織內是唯一。這表示沒有兩個組織共用相同的組織 ID。不過，另一個組織的 OU 或根可能與您的 ID 相同。我們建議您在指定 OU 或根時，一律包含組織 ID。

例如，以下條件會針對帳戶中的主體傳回 `true`。這些帳戶是直接連接到 `ou-ab12-22222222` OU，但並非其子 OU。

```
"Condition" : { "ForAnyValue:StringEquals" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

以下條件會針對帳戶中的主體傳回 `true`。該帳戶是直接連接到 OU 或其任何的子 OU。當您包含萬用字元時，您必須使用 `StringLike` 條件運算子。

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
*"]
}}
```

對於直接連接到任何子 OU 但不直接連接到父 OU 的帳戶中的主體，以下條件將傳回 `true`。先前的條件適用於 OU 或任何子系。以下條件僅適用於子系 (以及這些子系的任何子系)。

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
ou-*"]
}}
```

以下條件會允許存取 `o-a1b2c3d4e5` 組織中的任何主體，無論其父 OU 為何。

```
"Condition" : { "ForAnyValue:StringLike" : {
```

```
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/*"]
}}
```

aws:PrincipalOrgPaths 是多重值條件鍵。多重值金鑰在請求內容中可以擁有多個值。當您搭配 ForAnyValue 條件運算子使用多個值時，主體的路徑必須符合政策列出的其中一個路徑。如需有關多重值條件鍵的詳細資訊，請參閱 多值內容索引鍵。

```
    "Condition": {
        "ForAnyValue:StringLike": {
            "aws:PrincipalOrgPaths": [
                "o-a1b2c3d4e5/r-ab12/ou-ab12-33333333/*",
                "o-a1b2c3d4e5/r-ab12/ou-ab12-22222222/*"
            ]
        }
    }
```

AWS：PrincipalOrg身份證

使用此索引鍵可將要求主參與者所屬組織的 AWS Organizations 識別碼與原則中指定的識別碼進行比較。

- 可用性 – 只有在主體是組織的成員時，請求內容中才會包含此鍵。匿名請求不包含此鍵。
- 數據類型-字符串
- 值類型 - 單一值

此全域鍵提供了列出組織中所有 AWS 帳戶的所有帳戶 ID 的替代方法。您可以使用此條件鍵來簡化在資源型政策中指定 Principal 元素。您可以在條件元素中指定組織 ID。當您新增和移除帳戶時，包含 aws:PrincipalOrgID 鍵的政策會自動包含正確的帳戶，不需要手動更新。

例如，下列 Amazon S3 儲存貯體政策允許 o-xxxxxxxxxxx 組織中任何帳戶的成員將物件新增至 policy-ninja-dev 儲存貯體。

```
{
 "Version": "2012-10-17",
 "Statement": {
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:PutObject",
```

```
    "Resource": "arn:aws:s3:::policy-ninja-dev/*",
    "Condition": {"StringEquals":
      {"aws:PrincipalOrgID":"o-xxxxxxxxxxx"}
    }
  }
}
```

> **Note**
>
> 此全域條件也適用於 AWS 組織的管理帳戶。此政策可防止指定組織以外的所有主體存取 Amazon S3 儲存貯體。這包括任何與內部資源互動的 AWS 服務，例如將日誌資料 AWS CloudTrail 傳送到 Amazon S3 儲存貯體。若要瞭解如何安全地授與 AWS 服務存取權，請參閱AWS：PrincipalIsAWSService。

如需有關的詳細資訊 AWS Organizations，請參閱什麼是 AWS Organizations？ 在《AWS Organizations 使用者指南》中。

AWS：PrincipalTag/標籤鍵

使用此鍵來將連接至提出請求主體的標籤，與您在政策中所指定的標籤進行比較。若主體連接的標籤超過一個，請求內容會針對每個連接的標籤鍵包含一個 `aws:PrincipalTag` 鍵。

- 可用性 – 若主體搭配連接標籤使用 IAM 使用者，此鍵會包含在請求內容中。其會針對主體，使用具備連接標籤或工作階段標籤的 IAM 角色包含在其中。匿名請求不包含此鍵。
- 數據類型-字符串
- 值類型 - 單一值

您可以將自訂屬性以鍵值對的形式新增至使用者或角色。如需有關 IAM 標籤的詳細資訊，請參閱 標記 IAM 資源。您可以使用 `aws:PrincipalTag` 對 AWS 主體進行控制存取。

此範例會示範如何建立身分型政策，允許具有 **`department=hr`** 標籤的使用者管理 IAM 使用者、群組或角色。若要使用此政策，請將範例政策中的`########`取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": "hr"
        }
      }
    }
  ]
}
```

AWS：PrincipalIsAWSService

使用此索引鍵可檢查是否正由 AWS 服務主體直接呼叫資源。例如： AWS CloudTrail 使用服務主體 `cloudtrail.amazonaws.com` 將日誌寫入至 Amazon S3 儲存貯體。當服務使用服務主體對您的資源執行直接動作時，請求內容鍵會設定為 true (真)。服務代表 IAM 主體使用主體的憑證來提出請求時，內容鍵設定為 false。若服務使用服務角色或服務連結角色來代表主體呼叫，則也會設定為 false (偽)。

- 可用性 – 對於所有使用 AWS 憑證的已簽署 API 請求，此鍵會出現在請求內容中。匿名請求不包含此鍵。
- 資料類型 — 布林
- 值類型 - 單一值

您可以使用此條件金鑰來限制對受信任身分識別和預期網路位置的存取，同時安全地授予 AWS 服務存取權。

在下列 Amazon S3 儲存貯體政策範例中，除非請求來自服務主體`vpc-111bbb22`或來自服務主體 (例如)，否則對儲存貯體的存取將受到限制。 CloudTrail

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Expected-network+service-principal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/AWS Logs/AccountNumber/*",
```

```
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22"
        },
        "BoolIfExists": {
          "aws:PrincipalIsAWSService": "false"
        }
      }
    }
  ]
}
```

在下列影片中，進一步了解如何在政策中使用 aws:PrincipalIsAWSService 條件鍵。

安全地將存取權授予授權的使用者、預期的網路位置和 AWS 服務。

AWS：PrincipalService名稱

使用此鍵將政策中的服務主體名稱，以及向您的資源提出請求的服務主體進行比較。您可以使用這個鍵來檢查這個呼叫是否由特定的服務主體進行比較。當服務主體直接向您的資源要求時，aws:PrincipalServiceName 鍵會包含服務主體的名稱。例如， AWS CloudTrail 服務主體名稱為cloudtrail.amazonaws.com。

- 可用性 — 當 AWS 服務主體進行呼叫時，此金鑰會出現在要求中。此鍵在任何其他情況下都不存在，包括以下情況：
  - 若服務使用服務角色或服務連結角色來代表主體呼叫。
  - 服務使用 IAM 主體的憑證代表主體提出請求。
  - 如果呼叫是由 IAM 主體直接進行。
  - 如果呼叫由匿名請求者發出。
- 數據類型-字符串
- 值類型 - 單一值

您可以使用此條件金鑰來限制對受信任身分識別和預期網路位置的存取，同時安全地授與 AWS 服務的存取權。

在下列 Amazon S3 儲存貯體政策範例中，除非請求來自服務主體vpc-111bbb22或來自服務主體 (例如)，否則對儲存貯體的存取將受到限制。 CloudTrail

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "expected-network+service-principal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/AWS Logs/AccountNumber/*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22",
          "aws:PrincipalServiceName": "cloudtrail.amazonaws.com"
        }
      }
    }
  ]
}
```

AWS：PrincipalServiceNamesList

此鍵提供屬於該服務的所有服務主體名稱清單。這是進階條件鍵。您可以用其來限制服務僅從特定區域存取您的資源。某些服務可能會建立區域服務主體，以指示特定區域內服務的特定執行個體。您可以將資源的存取限制為服務的特定執行個體。當服務主體直接向您的資源要求時，aws:PrincipalServiceNamesList 包含與服務區域執行個體相關聯的所有服務主體名稱的無序清單。

- 可用性 — 當 AWS 服務主體進行呼叫時，此金鑰會出現在要求中。此鍵在任何其他情況下都不存在，包括以下情況：
  - 若服務使用服務角色或服務連結角色來代表主體呼叫。
  - 服務使用 IAM 主體的憑證代表主體提出請求。
  - 如果呼叫是由 IAM 主體直接進行。
  - 如果呼叫由匿名請求者發出。
- 數據類型-字符串（列表）
- 值類型 - 多重值

aws:PrincipalServiceNamesList 是多重值條件鍵。多值鍵在請求內容中可以擁有多個值。使用此鍵時，您必須使用 ForAnyValue 或 ForAllValues 設定運算子搭配字串條件運算子。如需有關多重值條件鍵的詳細資訊，請參閱 多值內容索引鍵。

AWS：PrincipalType

使用此鍵來將發出請求的主體類型與您在政策中所指定的主體類型進行比較。如需詳細資訊，請參閱指定主體。如需 principal 鍵值的具體範例，請參閱 主體索引鍵值。

- 可用性 – 此鍵會包含在所有請求的請求內容中，包括匿名請求。
- 數據類型-字符串
- 值類型 - 單一值

aws:userid

使用此鍵來將申請者的主體識別碼與您在政策中所指定的 ID 進行比較。針對 IAM 使用者，請求內容值是使用者 ID。針對 IAM 角色，此值的格式可能會不同。如需針對不同主體資訊顯示方式的詳細資訊，請參閱 指定主體。如需 principal 鍵值的具體範例，請參閱 主體索引鍵值。

- 可用性 – 此鍵會包含在所有請求的請求內容中，包括匿名請求。
- 數據類型-字符串
- 值類型 - 單一值

aws:username

使用此鍵將申請者的使用者名稱與您在政策中所指定的使用者名稱進行比較。如需針對不同主體資訊顯示方式的詳細資訊，請參閱 指定主體。如需 principal 鍵值的具體範例，請參閱 主體索引鍵值。

- 可用性 – 此鍵一律會包含在 IAM 使用者的請求內容中。使用 AWS 帳戶根使用者 或 IAM 角色提出的匿名請求和請求不包含此金鑰。使用 IAM Identity Center 憑證提出的請求不會在內容中包含此鍵。
- 數據類型-字符串
- 值類型 - 單一值

角色工作階段的屬性

使用下列條件索引鍵來比較產生工作階段時角色工作階段的屬性。只有當具有角色工作階段或同盟使用者認證的主參與者提出要求時，才能使用這些條件索引鍵。這些條件索引鍵的值會內嵌在角色的工作階段 Token 中。

角色是一種主參與者類型。您也可以使用主體的屬性區段中的條件索引鍵，在角色發出要求時評估角色的屬性。

內容

AWS：FederatedProvider

使用此鍵來將主體的發行身分提供者 (IdP) 與您在政策中所指定的 IdP 進行比較。這意味著使用`AssumeRoleWithWebIdentity` AWS STS 操作假定了 IAM 角色。使用產生角色工作階段的暫時憑證提出要求時，要求內容會識別驗證原始聯合身分的 IdP。

- 可用性：如果主體是角色工作階段主體，且該工作階段是使用 `AssumeRoleWithWebIdentity` 擔任角色時發出，則會出現此鍵。
- 數據類型-字符串
- 值類型 - 單一值

例如，如果使用者透過 Amazon Cognito 進行身分驗證，則要求內容包含值 `cognito-identity.amazonaws.com`。同樣，如果使用者透過 Login with Amazon 進行身分驗證，則要求內容包含值 `www.amazon.com`。

您可以使用任何單一值條件鍵作為變數。以下範例資源型政策會使用 `aws:FederatedProvider` 鍵作為資源的 ARN 中的政策變數。此政策允許使用 IdP 進行身分驗證的任何主體從 Amazon S3 儲存貯體中取得物件，並使用特定於發行身分提供者的路徑。

AWS：TokenIssue時間

使用此鍵來將發行暫時性安全憑證的日期和時間與您在政策中所指定的日期和時間進行比較。

- 可用性 – 只有在主體使用暫時憑證發出請求時，請求內容中才會包含此鍵。金鑰不存在於 AWS CLI 使用存取金鑰所建立的 AWS API 或 AWS SDK 要求中。
- 資料類型 — 日期
- 值類型 - 單一值

若要了解哪些服務支援使用暫時憑證，請參閱AWS 與 IAM 搭配使用的服務。

AWS：MultiFactorAuthAge

使用此鍵來將請求主體獲得授權使用 MFA 以來的秒數，與您在政策中所指定的數字進行比較。如需有關 MFA 的詳細資訊，請參閱 在中使用多因素身份驗證（MFA）AWS。

> **Important**
>
> 對於聯合身分識別或使用存取金鑰簽署 AWS CLI、 AWS API 或 AWS SDK 要求所發出的請求，此條件金鑰不存在。若要進一步瞭解如何使用臨時安全登入資料將 MFA 保護新增至 API 作業，請參閱設定受 MFA 保護的 API 存取。
> 若要檢查 MFA 是否用於驗證 IAM 聯合身分，您可以將身分識別提供者的身分驗證方法作 AWS 為工作階段標記傳遞給。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。若要針對 IAM 身分識別強制執行 MFA，您可以啟用存取控制的屬性，以透過身分識別提供者的身分驗證方法將 SAML 宣告宣告傳遞至 IAM 身分中心。

- 可用性 — 只有當主體使用暫時安全登入資料來提出要求時，此金鑰才會包含在要求前後關聯中。具有 MFA 條件的原則可附加至：
  - IAM 使用者或群組
  - 資源，例如 Amazon S3 儲存貯體、Amazon SQS 佇列或者 Amazon SNS 主題
  - 可由使用者擔任的 IAM 角色的信任政策
- 數據類型-數字
- 值類型 - 單一值

AWS：MultiFactorAuthPresent

使用此金鑰可檢查是否使用多重要素驗證 (MFA) 來驗證提出要求的暫時安全登入資料。

> **Important**
>
> 對於聯合身分識別或使用存取金鑰簽署 AWS CLI、 AWS API 或 AWS SDK 要求所發出的請求，此條件金鑰不存在。若要進一步瞭解如何使用臨時安全登入資料將 MFA 保護新增至 API 作業，請參閱設定受 MFA 保護的 API 存取。
>
> 若要檢查 MFA 是否用於驗證 IAM 聯合身分，您可以將身分識別提供者的身分驗證方法作 AWS 為工作階段標記傳遞給。如需詳細資訊，請參閱 傳遞工作階段標籤 AWS STS。若要針對 IAM 身分識別強制執行 MFA，您可以啟用存取控制的屬性，以透過身分識別提供者的身分驗證方法將 SAML 宣告宣告傳遞至 IAM 身分中心。

- 可用性 – 只有在主體使用暫時憑證發出請求時，請求內容中才會包含此鍵。具有 MFA 條件的原則可附加至：
  - IAM 使用者或群組
  - 資源，例如 Amazon S3 儲存貯體、Amazon SQS 佇列或者 Amazon SNS 主題
  - 可由使用者擔任的 IAM 角色的信任政策
- 資料類型 — 布林
- 值類型 - 單一值

臨時登入資料可用來驗證 IAM 角色和 IAM 使用者，使用來自AssumeRole或 GetSessionToken 的臨時權杖，以及 AWS Management Console.

IAM 使用者存取金鑰是長期登入資料，但在某些情況下， AWS 會代表 IAM 使用者建立臨時登入資料以執行作業。在這些情況下，`aws:MultiFactorAuthPresent` 鍵會出現在請求中，並設為 `false` 值。在兩種常見的情況下可能會發生此情況：

- 在 AWS Management Console 不知情的情況下使用臨時登入資料的 IAM 使用者。使用者會使用他們的使用者名稱和密碼 (長期憑證) 登入主控台。不過，主控台會在背景中代表使用者產生臨時憑證。
- 如果 IAM 使用者呼叫 AWS 服務，服務會重複使用該使用者的登入資料，向其他服務發出另一個要求。例如，在呼叫 Athena 存取 Amazon S3 儲存貯體時，或使 AWS CloudFormation 用建立 Amazon EC2 執行個體時。對於後續要求，請 AWS 使用臨時認證。

若要了解哪些服務支援使用暫時憑證，請參閱AWS 與 IAM 搭配使用的服務。

aws:MultiFactorAuthPresent 鍵在使用長期憑證 (例如使用者存取金鑰對) 呼叫 API 或 CLI 命令時，便不會出現。因此，我們建議您在檢查此鍵時使用條件運算子的 ...IfExists 版本。

請務必了解，下列 Condition 元素不是檢查是否使用 MFA 針對請求進行身分驗證的可靠方法。

```
#####   WARNING: NOT RECOMMENDED   #####
"Effect" : "Deny",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "false" } }
```

這個 Deny 效果、Bool 元素與 false 值的組合會拒絕可使用 MFA 驗證但卻未驗證的請求。這僅適用於支援使用 MFA 的暫時憑證。此陳述式不會拒絕存取使用長期憑證所提出的請求，或是使用 MFA 進行身分驗證的請求。請小心使用本範例，因為其邏輯十分複雜，而且未測試是否實際使用 MFA 身分驗證。

此外，請不要使用 Deny 效果、Null 元素與 true 的組合，因為其行為相同，而邏輯更為複雜。

建議的組合

我們建議您使用 BoolIfExists 運算子來檢查是否使用 MFA 驗證請求。

```
"Effect" : "Deny",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "false" } }
```

這個 Deny、BoolIfExists 與 false 的組合會拒絕未使用 MFA 驗證的請求。特別地是，它會拒絕來自未包含 MFA 之暫時憑證的請求。它也會拒絕使用長期認證 (例如使用存取金鑰進行的 AWS API 作業) 所 AWS CLI 提出的要求。*IfExists 運算子會檢查 aws:MultiFactorAuthPresent 鍵是否存在以及它是否可能存在，如其存在所指出。當您想要拒絕任何未使用 MFA 驗證的請求時，請使用此項目。這比較安全，但可能會破壞任何使用存取金鑰存取或 AWS API 的程式碼 AWS CLI 或指令碼。

替代組合

您也可以使用BoolIfExists運算子允許使用長期認證進行 MFA 驗證的要求和/ AWS CLI 或 AWS API 要求。

```
"Effect" : "Allow",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "true" } }
```

此條件會比對鍵是否存在，或者鍵是否不存在。這個 Allow、BoolIfExists 與 true 的組合允許使用 MFA 驗證的請求，或允許無法使用 MFA 驗證的請求。這意 AWS CLI味著當請求者使用其長期訪問

密鑰時，允許 AWS API 和 AWS SDK 操作。此組合不允許來自暫時憑證，且能夠包含 MFA 的請求但未這麼做的請求。

當您使用 IAM 主控台視覺化編輯器建立政策並選擇 MFA required (需要 MFA) 時，即會套用此組合。此設定需要 MFA 來存取主控台，但允許不使用 MFA 進行程式設計存取。

或者，您可以使用 Bool 運算子，僅在使用 MFA 驗證時允許程式設計和主控台請求。

```
"Effect" : "Allow",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "true" } }
```

這個 Allow、Bool 與 true 的組合只允許 MFA 驗證的請求。這僅適用於支援使用 MFA 的暫時憑證。此陳述式不允許存取使用長期存取鍵所提出的請求，或是使用暫時憑證但未使用 MFA 所提出的請求。

請不要使用與下列內容類似的政策建構來檢查是否存在 MFA 鍵：

```
#####   WARNING: USE WITH CAUTION   #####

"Effect" : "Allow",
"Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : "false" } }
```

這個 Allow 效果、Null 元素與 false 值的組合只允許可使用 MFA 驗證的請求，不論是否實際驗證請求都一樣。這會允許所有使用暫時憑證所提出的請求，並拒絕長期憑證的存取。請小心使用本範例，因為未測試是否實際使用 MFA 身分驗證。

AWS: EC2 虛擬私InstanceSource人雲端

此鍵可識別要接收 Amazon EC2 IAM 角色憑證的 VPC。您可以在具有 aws:SourceVPC 全域鍵的政策中使用此鍵，以檢查是否從與接收憑證之 VPC (aws:Ec2InstanceSourceVpc) 相符的 VPC (aws:SourceVPC) 進行呼叫。

- 可用性 – 每當請求者使用 Amazon EC2 角色憑證簽署請求時，請求內容中會包含此鍵。它可用於 IAM 政策、服務控制政策、VPC 端點政策和資源政策。
- 數據類型-字符串
- 值類型 - 單一值

此鍵可與 VPC 識別符值搭配使用，但是當與 aws:SourceVpc 內容鍵一起用作變數時最有用。只有在請求者使用 VPC 端點提出請求時，請求內容中才會包含 aws:SourceVpc 內容

鍵。搭配使用 aws:Ec2InstanceSourceVpc 與 aws:SourceVpc 可讓您更廣泛地使用 aws:Ec2InstanceSourceVpc，因為其會比較通常一起變更的值。

> **Note**
>
> 此條件鍵在 EC2-Classic 中不可用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireSameVPC",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
            "aws:SourceVpc": "${aws:Ec2InstanceSourceVpc}"
        },
        "Null": {
          "ec2:SourceInstanceARN": "false"
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

在上面的範例中，如果 aws:SourceVpc 值不等於 aws:Ec2InstanceSourceVpc 值，則會拒絕存取。政策聲明僅限於透過測試 ec2:SourceInstanceARN 條件鍵是否存在，作為 Amazon EC2 執行個體角色使用的角色。

該政策用aws:ViaAWSService AWS 於在代表 Amazon EC2 執行個體角色發出請求時授權請求。例如，當您從 Amazon EC2 執行個體向加密的 Amazon S3 儲存貯體提出請求時，Amazon S3 會代表您撥 AWS KMS 打電話給。提出要求時，某些金鑰不存在 AWS KMS。

AWS: EC2 私InstanceSource有 IPv4

此鍵可識別接收 Amazon EC2 IAM 角色憑證之主要彈性網路介面的私有 IPv4 地址。您必須搭配使用此條件鍵與其配套鍵 `aws:Ec2InstanceSourceVpc`，以確保您擁有 VPC ID 和來源私有 IP 的全域唯一組合。搭配使用此鍵與 `aws:Ec2InstanceSourceVpc`，來確保從接收憑證的相同私有 IP 地址提出請求。

- 可用性 – 每當請求者使用 Amazon EC2 角色憑證簽署請求時，請求內容中會包含此鍵。它可用於 IAM 政策、服務控制政策、VPC 端點政策和資源政策。
- 資料類型 — IP 位址
- 值類型 - 單一值

> **Important**
>
> 此鍵不應該單獨在 `Allow` 陳述式中使用。私有 IP 地址根據定義不是全域唯一的。每次使用 `aws:Ec2InstanceSourcePrivateIPv4` 鍵指定可從中使用 Amazon EC2 執行個體憑證的 VPC 時，都應該使用 `aws:Ec2InstanceSourceVpc` 鍵。

> **Note**
>
> 此條件鍵在 EC2-Classic 中不可用。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action":  "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:Ec2InstanceSourceVpc": "${aws:SourceVpc}"
                },
                "Null": {
                    "ec2:SourceInstanceARN": "false"
                },
                "BoolIfExists": {
```

```
                    "aws:ViaAWSService": "false"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action":  "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:Ec2InstanceSourcePrivateIPv4": "${aws:VpcSourceIp}"
                },
                "Null": {
                    "ec2:SourceInstanceARN": "false"
                },
                "BoolIfExists": {
                    "aws:ViaAWSService": "false"
                }
            }
        }
    ]
}
```

AWS：SourceIdentity

使用此鍵來將主體所設定的來源身分與您在政策中所指定的來源身分進行比較。

- 可用性 — 當使用任何 AWS STS assume-role CLI 命令或 AWS STS `AssumeRole` API 作業假設角色時，在設定來源身分識別之後，此金鑰會包含在要求內容中。
- 數據類型-字符串
- 值類型 - 單一值

您可以在原則中使用此機碼，允許主參與 AWS 者在擔任角色時設定來源識別的動作。AWS CloudTrail 中會顯示角色指定來源身分的活動。這可讓系統管理員更容易判斷使用中的角色執行動作的人員或哪些動作 AWS。

與 `sts:RoleSessionName` 不同，在設定來源身分之後，就無法變更值。這會出現在由角色所採取的所有動作的請求內容中。當您使用工作階段憑證來擔任另一個角色時，此值仍然存在於後續角色工作階段。從另一個角色取得及擔任角色稱為角色鏈結。

當主體最初設定來源識別，同時使用任何 AWS STS assume-role CLI 命令或 AWS STS AssumeRole API 作業假設角色時，索sts:SourceIdentity引鍵會出現在要求中。針對具有來源身分集的角色工作階段所採取的任何動作，aws:SourceIdentity 鍵會出現在請求中。

下列在帳戶 111122223333 中的 CriticalRole 角色信任政策包含針對 aws:SourceIdentity 的條件，可防止沒有設定為 Saanvi 或 Diego 的來源身分的主體擔任角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeRoleIfSourceIdentity",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::123456789012:role/CriticalRole"},
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringLike": {
                    "aws:SourceIdentity": ["Saanvi","Diego"]
                }
            }
        }
    ]
}
```

若要進一步了解有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

ec2：RoleDelivery

使用此金鑰可將已簽署請求中的執行個體中繼資料服務版本與 Amazon EC2 的 IAM 角色登入資料進行比較。執行個體中繼資料服務會依據任何指定請求 (PUT 或 GET 標頭) 來區分 IMDSv1 及 IMDSv2 請求，這些對 IMDSv2 而言是唯一的內容，會出現在該請求中。

- 可用性 — 每當 Amazon EC2 執行個體建立角色工作階段時，此金鑰都會包含在請求內容中。
- 數據類型-數字
- 值類型 - 單一值
- 範例值 — 1.0、2.0

您可以在每個執行個體上設定執行個體中繼資料服務 (IMDS)，此類本機程式碼或使用者必須使用 IMDSv2。當您指定必須使用該 IMDSv2 時，IMDSv1 則無法繼續運作。

- 執行個體中繼資料服務第 1 版 (IMDSv1) — 請求/回應方法
- 執行個體中繼資料服務第 2 版 (IMDSv2) – 工作階段導向方法

如需如何設定執行個體使用 IMDSv2 的相關資訊，請參閱設定執行個體中繼資料選項。

在下列範例中，如果要求內容中的 ec2: RoleDelivery 值為 1.0 (IMDSv1)，則會拒絕存取。通常可以套用此政策陳述式，因為如果請求未由 Amazon EC2 角色登入資料簽署，則無效。

```
{
    "Version": "2012-10-17",
    "Statement": [
               {
            "Sid": "RequireAllEc2RolesToUseV2",
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "NumericLessThan": {
                    "ec2:RoleDelivery": "2.0"
                }
            }
        }
    ]
}
```

如需詳細資訊，請參閱使用執行個體中繼資料的範例原則。

ec2：SourceInstance阿恩

使用此索引鍵可比較產生角色工作階段之執行個體的 ARN。

- 可用性 — 每當 Amazon EC2 執行個體建立角色工作階段時，此金鑰都會包含在請求內容中。
- 資料類型 — ARN
- 值類型 - 單一值
- 範例值 — Arn：awn：EC：美國西部-2：11111111：執行個體/執行個體識別碼

如需政策範例，請參閱允許特定執行個體檢視其他 AWS 服務中的資源。

膠水：RoleAssumed通過

此 AWS Glue 服務會為每個 AWS API 要求設定此條件金鑰，其 AWS Glue 中代表客戶使用服務角色發出要求 (不是由工作或開發人員端點，而是直接由 AWS Glue 服務提出)。使用此鍵可驗證對 AWS 資源的呼叫是否來自 AWS Glue 服務。

- 可用性 — 代表客戶使用服務角色提出請求時 AWS Glue ，此金鑰會包含在請求前後關聯中。
- 數據類型-字符串
- 值類型 - 單一值
- 範例值 — 此機碼一律設定為`glue.amazonaws.com`。

下列範例會新增條件，以允許 AWS Glue 服務從 Amazon S3 儲存貯體取得物件。

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:RoleAssumedBy": "glue.amazonaws.com"
        }
    }
}
```

膠水：CredentialIssuing服務

該 AWS Glue 服務使用來自作業或開發人員端點的服務角色為每個 AWS API 請求設置此密鑰。使用此金鑰可驗證對 AWS 資源的呼叫是來自 AWS Glue 工作或開發人員端點。

- 可用性 — 當發出來自工作或開發人員端點 AWS Glue 的請求時，此金鑰會包含在要求內容中。
- 數據類型-字符串
- 值類型 - 單一值
- 範例值 — 此機碼一律設定為`glue.amazonaws.com`。

下列範例會新增附加至 AWS Glue 工作所使用之 IAM 角色的條件。這可確保根據角色階段作業是否用於工作執行階段環境而允許/拒絕某些動 AWS Glue 作。

```
{
```

```
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

拉姆達：SourceFunction阿恩

使用此金鑰可識別 IAM 角色登入資料傳送至的 Lambda 函數 ARN。Lambda 服務會為來自函數執行環境的每個 AWS API 要求設定此金鑰。使用此鍵可驗證對 AWS 資源的呼叫是否來自特定 Lambda 函數的程式碼。Lambda 也會為來自執行環境之外的某些要求設定此金鑰，例如將記錄寫入 CloudWatch 和傳送追蹤至 X-Ray。

- 可用性 — 每當叫用 Lambda 函數程式碼時，此金鑰都會包含在要求內容中。
- 資料類型 — ARN
- 值類型 - 單一值
- 示例值-ARN：AWN：羊：美國東-1：123456789012：功能：TestFunction

下列範例允許一個特定的 Lambda 函數`s3:PutObject`存取指定的值區。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleSourceFunctionArn",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "ArnEquals": {
                    "lambda:SourceFunctionArn": "arn:aws:lambda:us-
east-1:123456789012:function:source_lambda"
                }
            }
        }
    ]
```

```
}
```

如需詳細資訊，請參閱AWS Lambda 發人員指南中的使用 Lambda 執行環境認證。

SSM：阿恩 SourceInstance

使用此金鑰可識別要交付 IAM 角色登入資料的 AWS Systems Manager 代管執行個體 ARN。當請求來自具有 IAM 角色與 Amazon EC2 執行個體設定檔相關聯的受管執行個體時，不會出現此條件金鑰。

- 可用性 — 每當角色認證傳送至 AWS Systems Manager 代管執行個體時，此金鑰都會包含在要求內容中。
- 資料類型 — ARN
- 值類型 - 單一值
- 範例值 — Arn：awn：EC：美國西部-2：11111111：執行個體/執行個體識別碼

身份存儲：UserId

使用此金鑰可將已簽署請求中的 IAM 身分中心員工身分與政策中指定的身分進行比較。

- 可用性 — 當請求的呼叫者是 IAM 身分中心的使用者時，會包含此金鑰。
- 數據類型-字符串
- 值類型 - 單一值
- 範例值 —

您可以使用 UserId AWS CLI、 AWS API 或 AWS SDK 向 I GetUserd API 發出請求，在 IAM 身分中心找到使用者的資訊。

網絡的屬性

您可以使用下列條件索引鍵，比較要求來源或透過您在原則中指定之網路內容傳遞之網路的詳細資料。

內容

- AWS：SourceIp
- AWS：SourceVpc
- AWS：SourceVpce
- AWS：VpcSourceIP

AWS：SourceIp

使用此鍵來將申請者的 IP 地址和您在政策中所指定的 IP 地址進行比較。aws:SourceIp 條件鍵僅可用於公有 IP 地址範圍。

- 可用性 – 此鍵會包含在請求內容中，但當申請者使用 VPC 端點提出請求時則除外。
- 資料類型 — IP 位址
- 值類型 - 單一值

aws:SourceIp 條件鍵可用在政策中，只允許主體在指定的 IP 範圍內提出請求。

> **Note**
>
> aws:SourceIp 同時支援 IPv4 和 IPv6 IP 地址或者 IP 地址範圍。如需支援 IPv6 AWS 服務的清單，請參閱AWS 服務 Amazon VPC 使用者指南中的支援 IPv6。

例如，您可以將以下身分型政策連接至 IAM 角色。如果使用者從指定的 IPv4 地址範圍進行呼叫，則此政策允許使用者將物件放入 DOC-EXAMPLE-BUCKET3 Amazon S3 儲存貯體。此原則也允許使用代表轉送存取工作階段您執行此作業的 AWS 服務。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalPutObjectIfIpAddress",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET3/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        }
    ]
}
```

如果您需要限制來自同時支援 IPv4 和 IPv6 定址的網路存取，您可以在 IAM 政策條件中包含 IPv4 和 IPv6 地址或 IP 地址範圍。如果使用者從指定的 IPv4 或 IPv6 地址範圍進行呼叫，則以下身分型政策允

許使用者將物件放入 DOC-EXAMPLE-BUCKET3 Amazon S3 儲存貯體。在 IAM 政策中包含 IPv6 位址範圍之前，請確認您使用的是支援 IPv6。 AWS 服務 如需支援 IPv6 AWS 服務 的清單，請參閱AWS 服務 Amazon VPC 使用者指南中的支援 IPv6。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalPutObjectIfIpAddress",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET3/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                }
            }
        }
    ]
}
```

如果請求來自使用 Amazon VPC 端點的主機，則 aws:SourceIp 鍵不可用。您應該改用 VPC 人雲端專用金鑰，例如 aws: VpcSource IP。如需有關使用 VPC 端點的詳細資訊，請參閱 AWS PrivateLink 指南中的 VPC 端點和 VPC 端點服務的身分與存取管理。

AWS：SourceVpc

使用此金鑰可檢查要求是否透過 VPC 端點所連接的 VPC 傳遞。在政策中，您可以使用此鍵，只允許存取特定 VPC。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的限制特定 VPC 的存取。

- 可用性 – 只有在申請者使用 VPC 端點提出請求時，請求內容中才會包含此鍵。
- 數據類型-字符串
- 值類型 - 單一值

AWS：SourceVpce

使用此鍵來將請求的 VPC 端點識別碼與您在政策中所指定的端點 ID 進行比較。在政策中，您可以使用此鍵來將存取限制在特定 VPC 端點。如需詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的限制特定 VPC 端點的存取。

- 可用性 – 只有在申請者使用 VPC 端點提出請求時，請求內容中才會包含此金鑰。
- 數據類型-字符串
- 值類型 - 單一值

AWS：VpcSourceIP

使用此鍵來將提出請求的 IP 地址與您在政策中所指定的 IP 地址進行比較。在政策中，只有在請求來自指定 IP 地址並且透過 VPC 端點時，此鍵才會相符。

- 可用性 – 只有在請求是使用 VPC 端點提出時，請求內容中才會包含此鍵。
- 資料類型 — IP 位址
- 值類型 - 單一值

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。

> **Note**
>
> `aws:VpcSourceIp` 同時支援 IPv4 和 IPv6 IP 地址或者 IP 地址範圍。如需支援 IPv6 AWS 服務 的清單，請參閱AWS 服務 Amazon VPC 使用者指南中的支援 IPv6。

資源的屬性

您可以使用下列條件索引鍵，比較屬於要求目標之資源的詳細資訊與您在策略中指定的資源特性。

內容

- AWS：ResourceAccount
- AWS：ResourceOrg路徑
- AWS：ResourceOrg身份證
- AWS：ResourceTag/標籤鍵

AWS：ResourceAccount

使用此鍵來比較請求的資源所有者的 AWS 帳戶 ID 與政策中的資源帳戶。然後，根據擁有該資源的帳戶來允許或拒絕對該資源的存取。

- 可用性 – 此鍵一律會包含在大部分服務動作的請求內容中。下列動作不支援此鍵：
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store – 所有動作
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`
    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVolume`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
    - `ec2:DeleteTransitGatewayPeeringAttachment`
    - `ec2:DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`
    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents`— 如果該事件匯流排在 2023 年 3 月 2 日之前設定為跨帳戶 EventBridge 目標，則 EventBridge`PutEvents`呼叫另一個帳戶中的事件匯流排。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的授予許可以允許來自其他 AWS 帳戶的事件。
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`

- Amazon Macie
  - `macie2:AcceptInvitation`
- Amazon OpenSearch 服務
  - `es:AcceptInboundConnection`
  - `es:CreateOutboundConnection`
- Amazon Route 53
  - `route53:AssociateVpcWithHostedZone`
  - `route53:CreateVPCAssociationAuthorization`
  - `route53:DeleteVPCAssociationAuthorization`
  - `route53:DisassociateVPCFromHostedZone`
  - `route53:ListHostedZonesByVPC`
- AWS Security Hub
  - `securityhub:AcceptAdministratorInvitation`
- 數據類型-字符串
- 值類型 - 單一值

> **Note**
>
> 如需上述不受支援動作的其他考量事項，請參閱資料周邊政策範例儲存庫。

此索引鍵等於在請求中評估之資源的帳號 AWS 帳戶 ID。

對於您帳戶中的大部分資源，ARN 包含該資源的擁有者帳戶 ID。對於某些資源 (例如 Amazon S3 儲存貯體)，資源 ARN 不包含帳戶 ID。以下兩個範例顯示在 ARN 中具有帳戶 ID 的資源與沒有帳戶 ID 的 Amazon S3 ARN 之間的差異：

- `arn:aws:iam::123456789012:role/AWSExampleRole` – 在帳戶 123456789012 中建立和擁有的 IAM 角色。
- `arn:aws:s3:::DOC-EXAMPLE-BUCKET2` – 在帳戶 111122223333 中建立和擁有 Amazon S3 儲存貯體，不會顯示在 ARN 中。

使用 AWS 主控台或 API 或 CLI 來尋找所有資源和對應的 ARN。

您撰寫的政策會根據資源擁有者的帳戶 ID 拒絕資源許可。例如，如果指定資源不屬於指定帳戶，下列身分型政策會拒絕存取這些資源。

若要使用此政策，請將斜體預留位置文字取代為您的帳戶資訊。

> **Important**
>
> 此政策不允許任何動作。相反，它使用 Deny 效果，明確拒絕存取陳述式中列出的不屬於所列帳戶的所有資源。將此政策與允許存取特定資源的其他政策結合使用。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyInteractionWithResourcesNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:region:account:*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "account"
          ]
        }
      }
    }
  ]
}
```

除非指定的擁有資源，否則此原則會拒絕存取特定 AWS 服務的所 AWS 帳戶 有資源。

> **Note**
>
> 有些資源 AWS 服務 需要存取在另一個中託管的 AWS 擁有資源 AWS 帳戶。在身分型政策中使用 `aws:ResourceAccount`，可能會影響您的身分存取這些資源。

某些 AWS 服務 (例如) 依賴於您以外的資源存取 AWS Data Exchange，以執 AWS 帳戶 行正常作業。如果在政策中使用元素 `aws:ResourceAccount`，請包含其他陳述式來建立這些服務的豁免。範例政策 AWS：拒絕存取您帳戶外部的 Amazon S3 資源，除非 AWS Data Exchange 示範如何在定義服務所擁有資源的例外狀況時，根據資源帳戶拒絕存取。

使用此政策範例作為建立您自己的自訂政策的範本。如需詳細資訊，請參閱您的服務文件。

AWS：ResourceOrg路徑

使用此索引鍵可將存取資源的「Organ AWS izations」路徑與策略中的路徑進行比較。在策略中，此條件金鑰可確保資源屬於組織中指定組織根目錄或組織單位 (OU) 內的 AWS 帳號成員。「 AWS 組 Organizations」路徑是「組織」實體結 Organizations 的文字表示。如需有關使用和了解路徑的詳細資訊，請參閱 了解 AWS Organizations 實體路徑。

- 可用性 – 只有在擁有資源的帳戶是組織成員時，請求內容中才會包含此鍵。此全域條件鍵不支援下列動作：
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store – 所有動作
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`
    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVolume`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
    - `ec2:DeleteTransitGatewayPeeringAttachment`
    - `ec2:DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`

    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents`— 如果該事件匯流排在 2023 年 3 月 2 日之前設定為跨帳戶 EventBridge 目標，則 EventBridge`PutEvents`呼叫另一個帳戶中的事件匯流排。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的授予許可以允許來自其他 AWS 帳戶的事件。
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`
  - Amazon Macie
    - `macie2:AcceptInvitation`
  - Amazon OpenSearch 服務
    - `es:AcceptInboundConnection`
    - `es:CreateOutboundConnection`
  - Amazon Route 53
    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
    - `route53:DeleteVPCAssociationAuthorization`
    - `route53:DisassociateVPCFromHostedZone`
    - `route53:ListHostedZonesByVPC`
  - AWS Security Hub
    - `securityhub:AcceptAdministratorInvitation`
- 數據類型-字符串（列表）
- 值類型 - 多重值

> **Note**
>
> 如需上述不受支援動作的其他考量事項，請參閱資料周邊政策範例儲存庫。

`aws:ResourceOrgPaths` 是多重值條件鍵。多值鍵在請求內容中可以擁有多個值。使用此鍵時，您必須使用 `ForAnyValue` 或 `ForAllValues` 設定運算子搭配字串條件運算子。如需有關多重值條件鍵的詳細資訊，請參閱 多值內容索引鍵。

例如，下列條件會對屬於組織 o-a1b2c3d4e5 的資源傳回 True。當您包含萬用字元時，必須使用StringLike條件運算子。

```
"Condition": {
      "ForAnyValue:StringLike": {
            "aws:ResourceOrgPaths":["o-a1b2c3d4e5/*"]
   }
}
```

針對具有 OU ID ou-ab12-11111111 的資源，下列條件會傳回 True。該條件會比對連接至 OU ou-ab12-11111111 或任何子 OU 的帳戶所擁有的資源。

```
"Condition": { "ForAnyValue:StringLike" : {
     "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/*"]
}}
```

下列條件會對直接連接至 OU ID ou-ab12-22222222 但未連接至子 OU 的帳戶所擁有的資源傳回 True。下列範例會使用StringEquals條件運算子來指定 OU ID 的完全相符需求，而非萬用字元相符項目。

```
"Condition": { "ForAnyValue:StringEquals" : {
     "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

> **Note**
>
> 有些資源 AWS 服務 需要存取在另一個中託管的 AWS 擁有資源 AWS 帳戶。在身分型政策中使用 aws:ResourceOrgPaths，可能會影響您的身分存取這些資源。

某些 AWS 服務 (例如) 依賴於您以外的資源存取 AWS Data Exchange，以執 AWS 帳戶 行正常作業。如果在政策中使用 aws:ResourceOrgPaths 鍵，請包含其他陳述式來建立這些服務的豁免。範例政策 AWS：拒絕存取您帳戶外部的 Amazon S3 資源，除非 AWS Data Exchange 示範如何在定義服務所擁有資源的例外狀況時，根據資源帳戶拒絕存取。您可以建立類似的政策，使用 aws:ResourceOrgPaths 鍵來限制組織單位 (OU) 內資源的存取，同時考量服務擁有的資源。

使用此政策範例作為建立您自己的自訂政策的範本。如需詳細資訊，請參閱您的服務文件。

AWS：ResourceOrg身份證

使用此索引鍵可將要求資源所屬組織中 AWS 組織的識別碼與策略中指定的識別碼進行比較。

- 可用性 – 只有在擁有資源的帳戶是組織成員時，請求內容中才會包含此鍵。此全域條件鍵不支援下列動作：
  - AWS Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Elastic Block Store – 所有動作
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CopyImage`
    - `ec2:CopySnapshot`
    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVolume`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
    - `ec2:DeleteTransitGatewayPeeringAttachment`
    - `ec2:DeleteVpcPeeringConnection`
    - `ec2:RejectTransitGatewayPeeringAttachment`
    - `ec2:RejectVpcEndpointConnections`
    - `ec2:RejectVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents`— 如果該事件匯流排在 2023 年 3 月 2 日之前設定為跨帳戶 EventBridge 目標，則 EventBridge`PutEvents`呼叫另一個帳戶中的事件匯流排。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南中的授予許可以允許來自其他 AWS 帳戶的事件。
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`

  - Amazon Macie
    - `macie2:AcceptInvitation`
  - Amazon OpenSearch 服務
    - `es:AcceptInboundConnection`
    - `es:CreateOutboundConnection`
  - Amazon Route 53
    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
    - `route53:DeleteVPCAssociationAuthorization`
    - `route53:DisassociateVPCFromHostedZone`
    - `route53:ListHostedZonesByVPC`
  - AWS Security Hub
    - `securityhub:AcceptAdministratorInvitation`
- 數據類型-字符串
- 值類型 - 單一值

> **Note**
>
> 如需上述不受支援動作的其他考量事項，請參閱資料周邊政策範例儲存庫。

此全域鍵會傳回指定請求的資源組織 ID。它可讓您建立規則，這些規則適用於身分型政策的 `Resource` 元素中指定的組織中的所有資源。您可以在條件元素中指定組織 ID。當您新增和移除帳戶時，包含 `aws:ResourceOrgID` 鍵的政策會自動包含正確的帳戶，您無需手動更新它。

例如，下列政策可避免主體將物件新增至 `policy-genius-dev` 資源，除非 Amazon S3 資源屬於與提出請求的主體相同的組織。

> **Important**
>
> 此政策不允許任何動作。相反，它使用 `Deny` 效果，明確拒絕存取陳述式中列出的不屬於所列帳戶的所有資源。將此政策與允許存取特定資源的其他政策結合使用。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "DenyPutObjectToS3ResourcesOutsideMyOrganization",
    "Effect": "Deny",
    "Action": "s3:PutObject",
    "Resource": "arn:partition:s3:::policy-genius-dev/*",
    "Condition": {
      "StringNotEquals": {
        "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
      }
    }
  }
}
```

> **Note**
>
> 有些資源 AWS 服務 需要存取在另一個中託管的 AWS 擁有資源 AWS 帳戶。在身分型政策中使用 `aws:ResourceOrgID`，可能會影響您的身分存取這些資源。

某些 AWS 服務 (例如) 依賴於您以外的資源存取 AWS Data Exchange，以執 AWS 帳戶 行正常作業。如果在政策中使用 `aws:ResourceOrgID` 鍵，請包含其他陳述式來建立這些服務的豁免。範例政策 AWS：拒絕存取您帳戶外部的 Amazon S3 資源，除非 AWS Data Exchange 示範如何在定義服務所擁有資源的例外狀況時，根據資源帳戶拒絕存取。您可以建立類似的政策，使用 `aws:ResourceOrgID` 鍵來限制組織內資源的存取，同時考量服務擁有的資源。

使用此政策範例作為建立您自己的自訂政策的範本。如需詳細資訊，請參閱您的服務文件。

在下列影片中，進一步了解如何在政策中使用 `aws:ResourceOrgID` 條件鍵。

確保身分和網路只能用來存取受信任的資源。

AWS：ResourceTag/標籤鍵

使用此鍵來將您在政策中所指定的標籤鍵值對與連接到資源的鍵值對進行比較。例如，您可以要求只在資源擁有連接標籤鍵 `"Dept"` 和值 `"Marketing"` 時才允許資源的存取。如需詳細資訊，請參閱 控制 AWS 資源的存取許可。

- 可用性 – 當所請求資源已連接標籤時，此鍵包含在請求內容中，或者包含在使用所連接標籤建立資源的請求中。只有在資源支援以標籤為基礎的授權時，才會傳回此鍵。每個標籤鍵值對都會有一個內容鍵。
- 數據類型-字符串
- 值類型 - 單一值

此內容鍵的格式為 "aws:ResourceTag/*tag-key*":"*tag-value*"，其中 *tag-key* 和 *tag-value* 是標籤鍵值組。標籤鍵與值皆不區分大小寫。這表示如果您在政策的條件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，則該條件會符合名為 TagKey1 或 tagkey1 的資源標籤鍵 (但不會同時符合兩者)。

如需使用 aws:ResourceTag 鍵來控制 IAM 資源存取的範例，請參閱 控制 AWS 資源的存取許可。

如需使用aws:ResourceTag金鑰控制對其他 AWS 資源存取權的範例，請參閱使用標籤控制 AWS 資源的存取。

如需有關使用屬性型存取控制 (ABAC) 之 aws:ResourceTag 條件鍵的教程，請參閱 IAM 教學課程：根據標籤定義存取 AWS 資源的許可。

請求的屬性

使用下列條件索引鍵，將要求本身和要求內容的詳細資訊與您在原則中指定的要求特性進行比較。

內容

- AWS：CalledVia
- AWS：CalledVia第一
- AWS：CalledVia最後
- AWS: 透過 AWSService
- AWS：CurrentTime
- AWS：EpochTime
- aws:Referer
- AWS：RequestedRegion
- AWS：RequestTag/標籤鍵
- AWS：TagKeys
- AWS：SecureTransport
- AWS：SourceArn

- AWS：SourceAccount
- AWS：SourceOrg路徑
- AWS：SourceOrg身份證
- AWS：UserAgent

AWS：CalledVia

使用此鍵可將政策中的服務與代表 IAM 主體 (使用者或角色) 提出請求的服務進行比較。當主體向服務發出要求時，該 AWS 服務可能會使用主體的認證，向其他服務發出後續要求。`aws:CalledVia` 鍵包含代表主體提出請求的鏈結中，每個服務的排序清單。

例如，您可以使用從 Amazon DynamoDB 表格 AWS CloudFormation 進行讀取和寫入。然後 DynamoDB 會使用由 AWS Key Management Service ()AWS KMS提供的加密。

- 可用性 – 支援 `aws:CalledVia` 的服務使用 IAM 主體的憑證向其他服務提出請求時，此鍵即出現在請求中。若服務使用服務角色或服務連結角色來代表主體呼叫，則此鍵不存在。若主體直接進行呼叫，此鍵亦不存在。
- 數據類型-字符串（列表）
- 值類型 - 多重值

若要在原則中使用`aws:CalledVia`條件索引鍵，您必須提供服務主體來允許或拒絕 AWS 服務要求。AWS 支援搭配使用下列服務主體`aws:CalledVia`。

服務主體
aoss.amazonaws.com
athena.amazonaws.com
backup.amazonaws.com
cloud9.amazonaws.com
cloudformation.amazonaws.com
databrew.amazonaws.com

服務主體
dataexchange.amazonaws.com
dynamodb.amazonaws.com
imagebuilder.amazonaws.com
kms.amazonaws.com
mgn.amazonaws.com
nimble.amazonaws.com
omics.amazonaws.com
ram.amazonaws.com
robomaker.amazonaws.com
servicecatalog-appregistry.amazonaws.com
sqlworkbench.amazonaws.com
ssm-guiconnect.amazonaws.com

若要在任何服務使用主體憑證提出請求時，允許或拒絕存取，請使用 AWS: ## AWSService 條件鍵。該條件鍵支持 AWS 服務。

aws:CalledVia 鍵是多值鍵。但是，您無法在條件中使用此鍵強制排序。使用上面的例子，使用者 1 向 AWS CloudFormation發出請求，其中 AWS CloudFormation會呼叫 DynamoDB，而 DynamoDB 呼叫 AWS KMS。這是三個獨立的請求。的最後呼叫 AWS KMS 是由使用者 1 透過再透過 AWS CloudFormation DynamoDB 執行。

在此情況下，請求環境中的 aws:CalledVia 鍵會依該排序包括 cloudformation.amazonaws.com 和 dynamodb.amazonaws.com。如果您只在意呼叫是透過請求鏈結中的某處的 DynamoDB 進行，則可以在政策中使用此條件鍵。

例如，下列原則允許管理名為的 AWS KMS 金鑰my-example-key，但僅當 DynamoDB 是要求的服務之一時才允許管理。ForAnyValue:StringEquals 條件運算子確保 DynamoDB 為呼叫服務之一。若主體直接呼叫 AWS KMS ，則條件會傳回 false，且此政策不允許該請求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KmsActionsIfCalledViaDynamodb",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": ["dynamodb.amazonaws.com"]
                }
            }
        }
    ]
}
```

如果您想強制執行鏈結中進行第一個或最後一個呼叫的服務，可以使用 aws:CalledViaFirst 和 aws:CalledViaLast 鍵。例如，下列原則允許管理my-example-key中名為的金鑰 AWS KMS。只有在鏈結中包含多個請求時，才允許這些 AWS KMS 操作。第一個請求必須透過 AWS CloudFormation 執行，而最後一個須透過 DynamoDB 進行。如果其他服務在鏈結中間發出請求，則仍允許該操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "KmsActionsIfCalledViaChain",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
```

```
                "kms:ReEncrypt*",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:region:111122223333:key/my-example-key",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "cloudformation.amazonaws.com",
                    "aws:CalledViaLast": "dynamodb.amazonaws.com"
                }
            }
        }
    ]
}
```

當服務使用 IAM 主體憑證來呼叫另一個服務時，aws:CalledViaFirst 和 aws:CalledViaLast 鍵即會出現在請求中。它們會指出在請求鏈中進行呼叫的第一個和最後一個服務。例如，假設 AWS CloudFormation 呼叫另一個名為的服務X Service，該服務呼叫 DynamoDB，然後呼叫該服務。AWS KMS的最後一個呼叫 AWS KMS 是User 1透過 AWS CloudFormation、然後再*X Service*透過 DynamoDB 來執行。它首先是通過調用，最 AWS CloudFormation 後通過 DynamoDB 調用。

AWS：CalledVia第一

使用此鍵可將政策中的服務與代表 IAM 主體 (使用者或角色) 提出請求的第一個服務進行比較。如需詳細資訊，請參閱 aws:CalledVia。

- 可用性 – 服務使用 IAM 主體的憑證向不同服務提出至少一個其他請求時，此鍵即出現在請求中。若服務使用服務角色或服務連結角色來代表主體呼叫，則此鍵不存在。若主體直接進行呼叫，此鍵亦不存在。
- 數據類型-字符串
- 值類型 - 單一值

AWS：CalledVia最後

使用此鍵可將政策中的服務與代表 IAM 主體 (使用者或角色) 提出請求的最後一個服務進行比較。如需詳細資訊，請參閱 aws:CalledVia。

- 可用性 – 服務使用 IAM 主體的憑證向不同服務提出至少一個其他請求時，此鍵即出現在請求中。若服務使用服務角色或服務連結角色來代表主體呼叫，則此鍵不存在。若主體直接進行呼叫，此鍵亦不存在。

- 數據類型-字符串
- 值類型 - 單一值

AWS: 透過 AWSService

使用此金鑰可檢查 AWS 服務是否代表您向其他服務發出要求。

服務代表 IAM 主體使用主體的憑證來提出請求時，請求環境鍵即傳回 `true`。若服務使用服務角色或服務連結角色來代表主體呼叫，則環境鍵即傳回 `false`。主體直接進行呼叫時，請求環境鍵也會傳回 `false`。

- 可用性 – 此金鑰一律會包含在請求內容中。
- 資料類型 — 布林
- 值類型 - 單一值

您可以使用此條件鍵，根據請求是否由服務提出，允許或拒絕存取。

AWS：CurrentTime

使用此鍵來將請求的日期和時間及您在政策中所指定的日期與時間進行比較。如要檢視使用此條件鍵的範例政策，請參閱 AWS：允許根據日期和時間進行存取。

- 可用性 – 此金鑰一律會包含在請求內容中。
- 資料類型 — 日期
- 值類型 - 單一值

AWS：EpochTime

使用此鍵來將請求的日期和時間 (epoch 或 Unix 時間) 與您在政策中所指定的日期與時間進行比較。此鍵也接受自 1970 年 1 月 1 日以來的秒數。

- 可用性 – 此金鑰一律會包含在請求內容中。
- 數據類型-日期，數字
- 值類型 - 單一值

aws:Referer

使用此鍵來將在用戶端瀏覽器中推薦請求的人員，與您在政策中所指定的推薦者進行比較。aws:referer 請求內容的值會由呼叫者在 HTTP 標頭中提供。當您選取網頁上的連結時，Referer 標頭會包含在 Web 瀏覽器請求中。Referer 標頭包含選取連結的網頁 URL。

- 可用性 — 只有當透過從瀏覽器中的網頁 URL 連結呼叫 AWS 資源的要求時，此機碼才會包含在要求前後關聯中。程式設計請求不會包含此鍵，因為它不會使用瀏覽器連結來存取 AWS 資源。
- 數據類型-字符串
- 值類型 - 單一值

例如，您可以直接使用 URL 或使用直接 API 呼叫來存取 Amazon S3 物件。如需詳細資訊，請參閱直接使用 Web 瀏覽器的 Amazon S3 API 操作。當您從存在於網頁中的 URL 存取 Amazon S3 物件時，會在 aws:referer 中使用來源網頁的 URL。當您在瀏覽器中輸入 URL 來存取 Amazon S3 物件時，aws:referer 不存在。當您直接呼叫 API 時，aws:referer 也不存在。您可以使用政策中的 aws:referer 條件鍵來允許特定參照者提出的請求，例如公司網域中網頁上的連結。

> **Warning**
>
> 應該小心使用此鍵。包含公開已知推薦者標頭值相當危險。未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 aws:referer 值。因此，不aws:referer應用於防止未經授權的人士提出直接 AWS 要求。它僅用於允許客戶保護其數位內容 (例如存放在 Amazon S3 中的內容)，使其不被未經授權的第三方網站參考。

AWS：RequestedRegion

使用此索引鍵可將要求中呼叫的 [AWS 區域] 與您在原則中指定的 [區域] 進行比較。您可以使用此全域條件鍵來控制可接受請求的區域。若要檢視每個服務的 AWS 區域，請參閱 Amazon Web Services 一般參考.

- 可用性 – 此鍵一律會包含在請求內容中。
- 數據類型-字符串
- 值類型 - 單一值

有些全域服務 (例如 IAM) 只會有單一端點。因為此端點實際位於美國東部 (維吉尼亞北部) 區域，所以一律會對 us-east-1 區域提出 IAM 呼叫。例如，如果您建立的政策拒絕存取所有服務 (如果請求的區域

不是 us-west-2)，則 IAM 呼叫一律會失敗。若要檢視如何解決此問題的範例，請參閱 NotAction 「拒絕」。

> **Note**
>
> aws:RequestedRegion 條件鍵可讓您控制哪個端點叫用的服務，但不控制操作的影響。有些服務具有跨區域影響。
> 例如，Amazon S3 具有延及其他區域的 API 操作。
>
> - 您可以在一個區域 (受 s3:PutBucketReplication 條件鍵影響) 中叫用aws:RequestedRegion，但其他區域會根據複寫組態設定而受到影響。
> - 您可以調用 s3:CreateBucket 以在其他區域建立儲存貯體，並使用 s3:LocationConstraint 條件鍵控制適用的區域。

您可以使用此內容索引鍵來限制對指定區域集內 AWS 服務的存取。例如，下列政策可讓使用者在 AWS Management Console中檢視所有 Amazon EC2 執行個體。不過，只允許他們對愛爾蘭 (eu-west-1)、倫敦 (eu-west-2) 或巴黎 (eu-west-3) 的執行個體進行變更。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InstanceConsoleReadOnly",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:Export*",
                "ec2:Get*",
                "ec2:Search*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "InstanceWriteRegionRestricted",
            "Effect": "Allow",
            "Action": [
                "ec2:Associate*",
                "ec2:Import*",
                "ec2:Modify*",
                "ec2:Monitor*",
```

```
                "ec2:Reset*",
                "ec2:Run*",
                "ec2:Start*",
                "ec2:Stop*",
                "ec2:Terminate*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": [
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

AWS：RequestTag/標籤鍵

使用此鍵來將請求中傳遞的標籤鍵/值對與您在政策中所指定的標籤對進行比較。例如，您可以檢查請求是否包含標籤鍵 "Dept" 並且其具有值 "Accounting"。如需詳細資訊，請參閱 在 AWS 請求期間控制存取許可。

- 可用性 – 在請求中傳遞標籤鍵/值對時，鍵會包含在請求內容中。在請求中傳遞多個標籤時，每個標籤鍵值對都會有一個內容鍵。
- 數據類型-字符串
- 值類型 - 單一值

此內容鍵的格式為 "aws:RequestTag/*tag-key*":"*tag-value*"，其中 *tag-key* 和 *tag-value* 是標籤鍵值組。標籤鍵與值皆不區分大小寫。這表示如果您在政策的條件元素中指定 "aws:RequestTag/TagKey1": "Value1"，則該條件會符合名為 TagKey1 或 tagkey1 的要求標籤鍵 (但不會同時符合兩者)。

此範例顯示雖然鍵為單一值，但如果鍵不同，您仍然可以在請求中使用多個鍵/值對。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:::instance/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/environment": [
          "preprod",
          "production"
        ],
        "aws:RequestTag/team": [
          "engineering"
        ]
      }
    }
  }
}
```

AWS：TagKeys

使用此鍵來將請求中的標籤鍵與您在政策中所指定的鍵進行比較。當使用政策來控制使用標籤的存取時，建議您使用 aws:TagKeys 條件鍵來定義允許的標籤鍵。如需範例政策和詳細資訊，請參閱 the section called “根據標籤索引鍵控制存取權限”。

- 可用性 – 如果操作支援在請求中傳遞標籤，則此鍵會包含在請求內容中。
- 數據類型-字符串（列表）
- 值類型 - 多重值

此內容鍵的格式為 "aws:TagKeys":"*tag-key*"，其中 *tag-key* 是沒有值的標籤鍵清單 (例如，["Dept","Cost-Center"])。

由於您可以在請求中包含多個標籤鍵值對，因此請求內容可能會是多值請求。在這種情況下，您必須使用 ForAllValues 或 ForAnyValue 設定運算子。如需詳細資訊，請參閱 多值內容索引鍵。

有些服務支援標記搭配資源操作，如建立、修改或刪除資源。若要允許標記和操作作為單一呼叫，您必須建立一個政策，同時包括標記動作和資源修改動作。然後，您可以使用 aws:TagKeys 條件鍵以強制執行在請求中使用特定的標籤鍵。例如，若要在某人建立 Amazon EC2 快照時限制標籤，您必須在政策中包含 ec2:CreateSnapshot 建立動作和 ec2:CreateTags 標記動作。若要檢視此使用案例的政策aws:TagKeys，請參閱 Amazon EC2 使用者指南中的使用標籤建立快照。

AWS：SecureTransport

使用此鍵來檢查請求是否使用 SSL 進行傳送。請求內容會傳回 `true` 或 `false`。在政策中，您可以允許只有在請求使用 SSL 傳送時，才能進行特定動作。

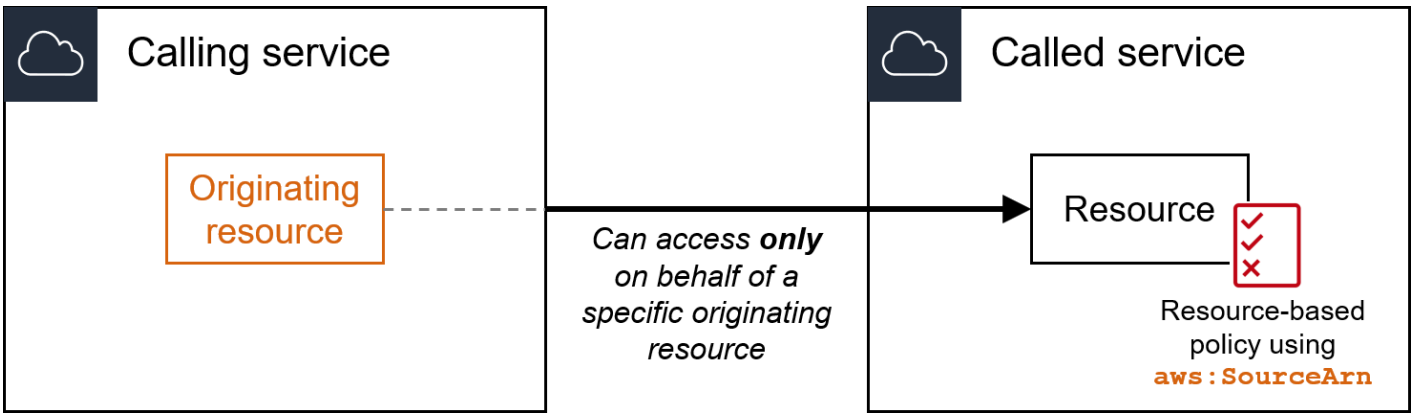
- 可用性 – 此金鑰一律會包含在請求內容中。
- 資料類型 — 布林
- 值類型 - 單一值

AWS：SourceArn

使用此 service-to-service 索引鍵可將發出請求的資源的 Amazon 資源名稱 (ARN) 與您在政策中指定的 ARN 進行比較，但只有在請求是由 AWS 服務主體發出時才會進行比較。如果來源的 ARN 包括帳戶 ID 時，則不需要搭配 `aws:SourceAccount` 使用 `aws:SourceArn`。

此鍵不適用提出請求的主體 ARN。請改用 `AWS#PrincipalArn`。

- 可用性 — 只有當AWS 服務主體代表組態觸發要求的資源直接呼叫您的資源時，此索引鍵才會包含在 service-to-service 要求內容中。發出呼叫的服務會將原始資源的 ARN 傳遞給被呼叫的服務。



下列服務整合不支援此全域條件鍵：

發出呼叫的服務 (服務主體)	被呼叫的服務 (資源型政策)	描述
logdelivery.elb.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄

發出呼叫的服務 (服務主體)	被呼叫的服務 (資源型政策)	描述
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄

> **Note**
>
> 並非所有與 AWS Security Token Service (AWS STS) 和 AWS Key Management Service (AWS KMS) 的服務整合都受到支援。如需詳細資訊，請參閱發出呼叫的服務的文件。針對 AWS 服務 透過 KMS 金鑰授權使用的`aws:SourceArn`金鑰使用 KMS 金鑰原則，可能會導致非預期的行為。

- 數據類型-ARN，字符串

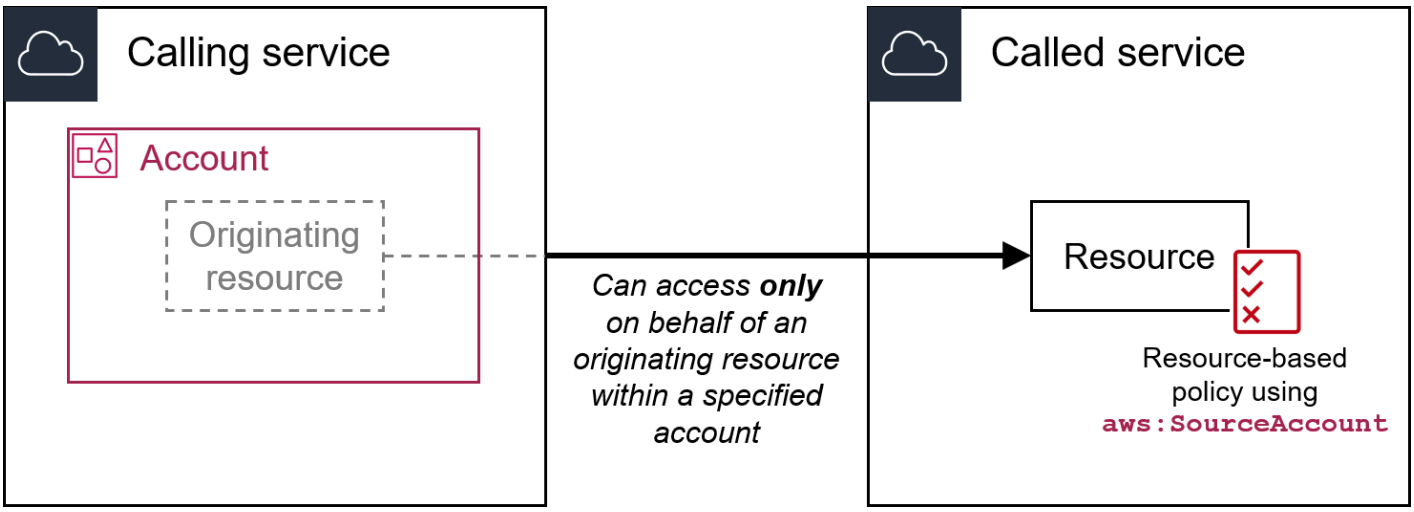
  AWS 建議您在比較 ARN 時使用 ARN 運算子而非字串運算子。
- 值類型 - 單一值

您可以使用此條件金鑰來防止在 AWS 服務之間的交易期間將服務用作混淆的副手。僅在以資源為基礎的策略中使用此機碼，其中`Principal`為 AWS 服務 主體。將此條件鍵的值設定為請求中資源的 ARN。例如，當 Amazon S3 儲存貯體更新觸發 Amazon SNS 主題發布時，Amazon S3 服務會調用 `sns:Publish` API 操作。在允許 `sns:Publish` 操作的主題政策中，將條件鍵的值設定為 Amazon S3 儲存貯體的 ARN。如需建議使用此條件金鑰的方式和時機的詳細資訊，請參閱您使用之 AWS 服務的說明文件。

AWS：SourceAccount

使用此索引鍵可將提出 service-to-service要求之資源的帳號 ID 與您在策略中指定的帳號 ID 進行比較，但只有在要求是由 AWS 服務主體發出時才會進行比較。

- 可用性 — 只有當AWS 服務主體代表組態觸發要求的資源直接呼叫您的資源時，此索引鍵才會包含在 service-to-service 要求內容中。發出呼叫服務會將原始資源的帳戶 ID 傳遞給被呼叫的服務。



下列服務整合不支援此全域條件鍵：

發出呼叫的服務 (服務主體)	被呼叫的服務 (資源型政策)	描述
logdelivery.elb.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄

> **Note**
>
> 並非所有與 AWS Security Token Service (AWS STS) 和 AWS Key Management Service (AWS KMS) 的服務整合都受到支援。如需詳細資訊，請參閱發出呼叫的服務的文件。針對 AWS 服務 透過 KMS 金鑰授權使用的`aws:SourceAccount`金鑰使用 KMS 金鑰原則，可能會導致非預期的行為。

- 數據類型-字符串
- 值類型 - 單一值

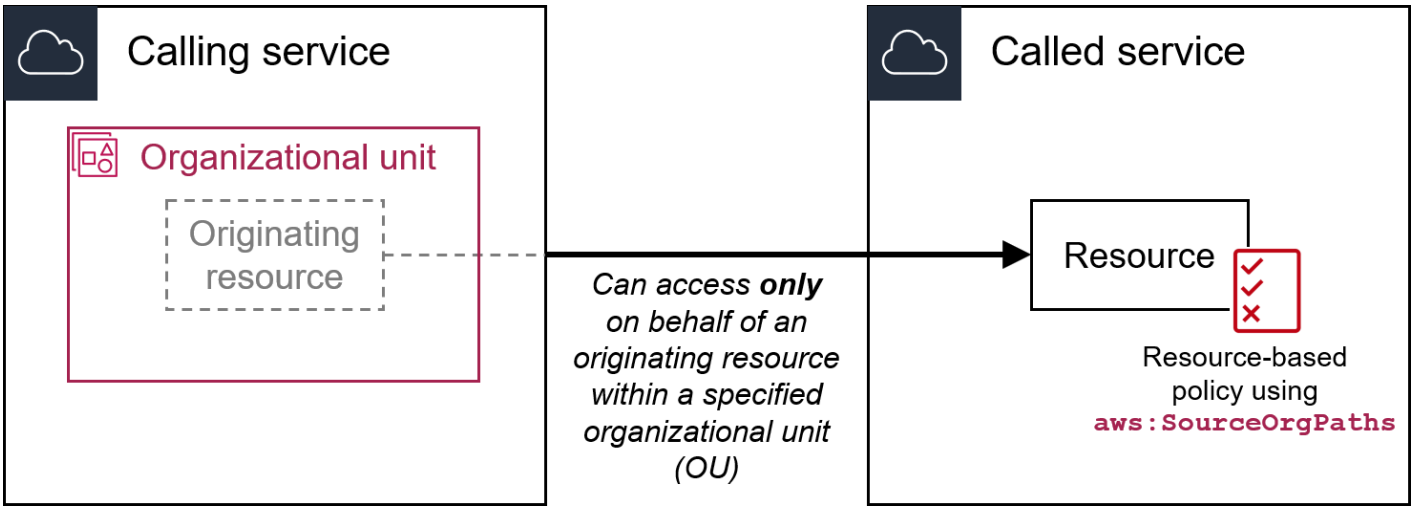
您可以使用此條件金鑰來防止在 AWS 服務之間的交易期間將服務用作混淆的副手。僅在以資源為基礎的策略中使用此機碼，其中`Principal`為 AWS 服務 主體。將此條件鍵的值設定為請求中資源的帳

戶 ID。例如，當 Amazon S3 儲存貯體更新觸發 Amazon SNS 主題發布時，Amazon S3 服務會調用 `sns:Publish` API 操作。在允許 `sns:Publish` 操作的主題政策中，將條件鍵的值設定為 Amazon S3 儲存貯體的帳戶 ID。如需建議如何及何時建議使用此條件金鑰的資訊，請參閱您使用之 AWS 服務的說明文件。

AWS：SourceOrg路徑

使用此索引鍵可將提出 service-to-service 要求的資源 AWS Organizations 路徑與您在策略中指定的組織路徑進行比較，但只有在要求是由 AWS 服務主體發出時才可進行比較。 Organizations 路徑是 Organizations 實體結構的文字表示。如需有關使用和了解路徑的詳細資訊，請參閱了解 AWS Organizations 實體路徑。

- 可用性 – 只有當 AWS 服務主體代表組織成員帳戶所擁有的資源直接呼叫您的資源時，此鍵才會包含在請求內容中。發出呼叫的服務會將原始資源的組織路徑傳遞給被呼叫的服務。



下列服務整合不支援此全域條件鍵：

發出呼叫的服務 (服務主體)	被呼叫的服務 (資源型政策)	描述
logdelivery.elb.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄

發出呼叫的服務 (服務主體)	被呼叫的服務 (資源型政策)	描述
所有服務主體	Amazon Lex 機器人	允許 AWS 服務 使用 Amazon Lex 機器人

> **Note**
>
> 並非所有與 AWS Security Token Service (AWS STS) 和 AWS Key Management Service (AWS KMS) 的服務整合都受到支援。如需詳細資訊，請參閱發出呼叫的服務的文件。針對 AWS 服務 透過 KMS 金鑰授權使用的`aws:SourceOrgPaths`金鑰使用 KMS 金鑰原則，可能會導致非預期的行為。

- 數據類型-字符串（列表）
- 值類型 - 多重值

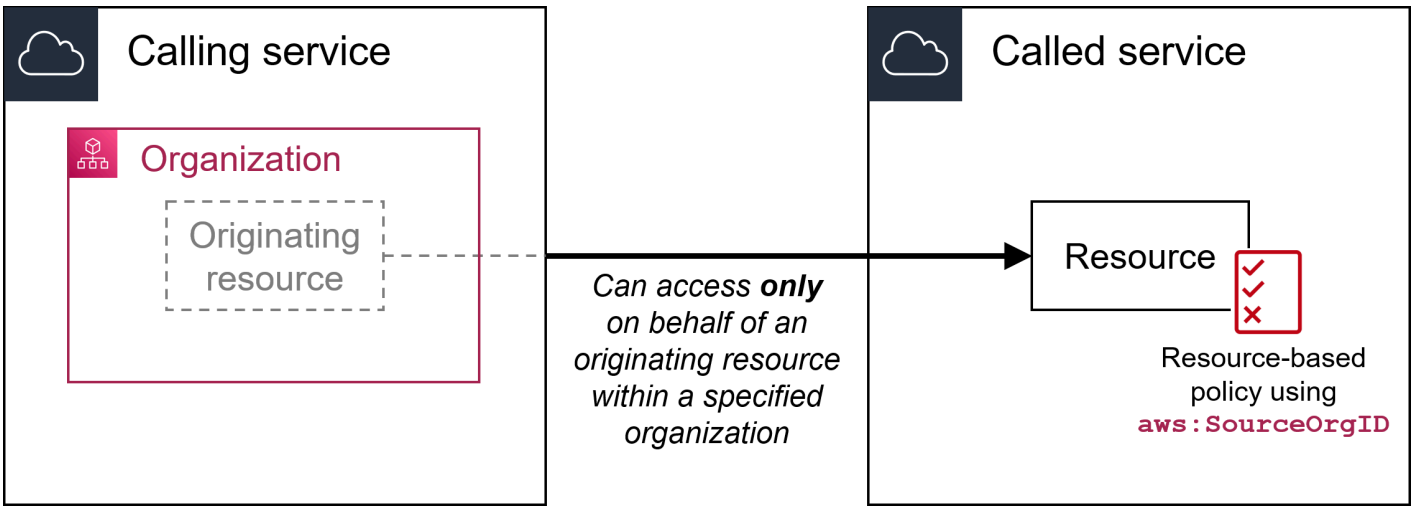
您可以使用此條件金鑰來防止在 AWS 服務之間的交易期間將服務用作混淆的副手。僅在以資源為基礎的策略中使用此機碼，其中`Principal`為 AWS 服務 主體。將此條件鍵的值設定為請求中資源的組織路徑。例如，當 Amazon S3 儲存貯體更新觸發 Amazon SNS 主題發布時，Amazon S3 服務會調用 `sns:Publish` API 操作。在允許 `sns:Publish` 操作的主題政策中，將條件鍵的值設定為 Amazon S3 儲存貯體的組織路徑。如需建議使用此條件金鑰的方式和時機的詳細資訊，請參閱您使用之 AWS 服務的說明文件。

`aws:SourceOrgPaths` 是多重值條件鍵。多值鍵在請求內容中可以擁有多個值。使用此鍵時，您必須使用 `ForAnyValue` 或 `ForAllValues` 設定運算子搭配字串條件運算子。如需有關多重值條件鍵的詳細資訊，請參閱 多值內容索引鍵。

AWS：SourceOrg身份證

使用此索引鍵可將提出 service-to-service 要求之資源的組織識別碼與您在策略中指定的組織識別碼進行比較，但只有當請求是由 AWS 服務主體提出時才會進行比較。當您對 AWS Organizations中的組織新增和移除帳戶時，包含 `aws:SourceOrgID` 鍵的政策會自動包含正確的帳戶，您無需手動更新政策。

- 可用性 – 只有當 AWS 服務主體代表組織成員帳戶所擁有的資源直接呼叫您的資源時，此鍵才會包含在請求內容中。發出呼叫的服務會將原始資源的組織 ID 傳遞給被呼叫的服務。



下列服務整合不支援此全域條件鍵：

發出呼叫的服務 (服務主體)	被呼叫的服務 (資源型政策)	描述
logdelivery.elb.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄
logdelivery.elasticloadbalancing.amazonaws.com	Amazon S3 儲存貯體	在 Amazon S3 儲存貯體中啟用 Elastic Load Balancing 存取記錄
所有服務主體	Amazon Lex 機器人	允許 AWS 服務 使用 Amazon Lex 機器人

> **Note**
>
> 並非所有與 AWS Security Token Service (AWS STS) 和 AWS Key Management Service (AWS KMS) 的服務整合都受到支援。如需詳細資訊，請參閱發出呼叫的服務的文件。針對 AWS 服務 透過 KMS 金鑰授權使用的`aws:SourceOrgID`金鑰使用 KMS 金鑰原則，可能會導致非預期的行為。

- 數據類型-字符串
- 值類型 - 單一值

您可以使用此條件金鑰來防止在 AWS 服務之間的交易期間將服務用作混淆的副手。僅在以資源為基礎的策略中使用此機碼，其中Principal為 AWS 服務 主體。將此條件鍵的值設定為請求中資源的組織 ID。例如，當 Amazon S3 儲存貯體更新觸發 Amazon SNS 主題發布時，Amazon S3 服務會調用 sns:Publish API 操作。在允許 sns:Publish 操作的主題政策中，將條件鍵的值設定為 Amazon S3 儲存貯體的組織 ID。如需建議使用此條件金鑰的方式和時機的詳細資訊，請參閱您使用之 AWS 服務的說明文件。

AWS：UserAgent

使用此鍵來將申請者的用戶端應用程式與您在政策中指定的應用程式進行比較。

- 可用性 – 此鍵一律會包含在請求內容中。
- 數據類型-字符串
- 值類型 - 單一值

> **Warning**
>
> 應該小心使用此鍵。由於aws:UserAgent 值由 HTTP 標頭中的呼叫者提供，未授權方可以使用修改的或自訂瀏覽器來提供他們選擇的任何 aws:UserAgent 值。因此，不aws:UserAgent應用於防止未經授權的人士提出直接 AWS 要求。您可以使用它僅允許特定用戶端應用程式，而且只在測試政策之後才能使用。

其他跨服務條件鍵

AWS STS 支援 OIDC 聯合的 SAML 型聯合條件金鑰和跨服務條件金鑰。當使用 SAML 聯合的使用者在其他服務中執行 AWS 作業時，可以使用這些金鑰。

IAM 和 AWS STS 條件內容金鑰

您可以使用 JSON 政策中的Condition元素來測試包含在所有要求之要 AWS 求內容中的索引鍵值。這些鍵提供有關請求本身或請求參考的資源的資訊。在允許使用者請求的動作之前，您可以檢查鍵是否有指定的值。這可讓您可以在 JSON 政策陳述式符合或不符合傳入請求時進行精細控制。如需有關如何 JSON 政策中使用 Condition 元素的資訊，請參閱 IAM JSON 政策元素：Condition。

本主題說明 IAM 服務 (帶有前置詞) 和 () 服務 (含iam:前置詞AWS STS) 所定義和提供的sts:金鑰。AWS Security Token Service 其他一些 AWS 服務也提供與該服務定義的動作和資源相關的服務特定金鑰。如需詳細資訊，請參閱AWS 服務的動作、資源和條件金鑰。支援條件鍵的服務文件通常包含其

他資訊。例如，如需可在 Amazon S3 資源政策中使用之金鑰的資訊，請參閱 Amazon Simple Storage Service 使用者指南中的 Amazon S3 政策鍵。

主題

- IAM 的可用鍵
- AWS OIDC 聯盟的可用金鑰
- SAML AWS STS 聯合身分的可用鍵
- 跨服務 SAML 型聯合內容索引鍵 AWS STS
- 可用的按鍵 AWS STS

IAM 的可用鍵

您可以在控制存取 IAM 資源的政策中使用以下條件鍵：

IAM：AssociatedResourceArn

使用 ARN 運算子。

指定將在目標服務與此角色建立關聯之資源的 ARN。資源通常屬於主體者傳遞角色的目標服務。有時候，資源也可能屬於第三個服務。例如，您可以將角色傳遞給他們在 Amazon EC2 執行個體上使用的 Amazon EC2 Auto Scaling。在這種情況下，條件可能會符合 Amazon EC2 執行個體的 ARN。

此條件索引鍵僅適用於策略中的PassRole動作。不能用來限制任何其他動作。

在政策中使用此條件鍵以允許實體傳遞角色，但只有在該角色與指定的資源相關聯時才行。您可以使用萬用字元 (*)，允許對特定類型的資源執行操作，而不限制區域或資源 ID。例如，您可以允許 IAM 使用者或角色將任何角色傳遞給 Amazon EC2 服務，此服務會搭配區域 `us-east-1` 或 `us-west-1` 中的執行個體使用。不允許 IAM 使用者或角色將角色傳遞給其他服務。此外，它不允許 Amazon EC2 將該角色搭配其他區域中的執行個體使用。

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
        "StringEquals": {"iam:PassedToService": "ec2.amazonaws.com"},
        "ArnLike": {
            "iam:AssociatedResourceARN": [
```

```
                "arn:aws:ec2:us-east-1:111122223333:instance/*",
                "arn:aws:ec2:us-west-1:111122223333:instance/*"
            ]
        }
    }
}
```

> **Note**
>
> AWS 支援 iam: 的服務PassedToService也支援此條件金鑰。

IAM：AWSServiceName

使用字串運算子。

指定此角色所附加的 AWS 服務。

在此範例中，若服務名稱為 access-analyzer.amazonaws.com，您可允許實體建立服務連結角色。

```
{
   "Version": "2012-10-17",
   "Statement": [{
       "Effect": "Allow",
       "Action": "iam:CreateServiceLinkedRole",
       "Resource": "*",
       "Condition": {
         "StringLike": {
           "iam:AWSServiceName": "access-analyzer.amazonaws.com"
         }
       }
     }]
 }
```

iam:FIDO-certification

使用字串運算子。

在註冊 FIDO 安全金鑰時檢查 MFA 裝置 FIDO 認證等級。從 FIDO Alliance Metadata Service (MDS) 中擷取裝置認證。如果 FIDO 安全金鑰的認證狀態或等級發生變更，除非裝置已取消註冊並再次註冊以取得更新的認證資訊，否則其將不會更新。

L1、L1plus、L2、L2plus、L3、L3plus 的可能值

在此範例中，您註冊安全金鑰並擷取您裝置的 FIDO Level 1 plus 認證。

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-certification": "L1plus"
                }
            }
        }
    ]

 }
```

iam:FIDO-FIPS-140-2-certification

使用字串運算子。

在註冊 FIDO 安全金鑰時檢查 MFA 裝置 FIPS-140-2 驗證認證等級。從 FIDO Alliance Metadata Service (MDS) 中擷取裝置認證。如果 FIDO 安全金鑰的認證狀態或等級發生變更，除非裝置已取消註冊並再次註冊以取得更新的認證資訊，否則其將不會更新。

L1、L2、L3、L4 的可能值

在此範例中，您註冊安全金鑰並擷取您裝置的 FIPS-140-2 Level 2 認證。

```
{
```

```
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        }
    ]

}
```

iam:FIDO-FIPS-140-3-certification

使用字串運算子。

在註冊 FIDO 安全金鑰時檢查 MFA 裝置 FIPS-140-3 驗證認證等級。從 FIDO Alliance Metadata Service (MDS) 中擷取裝置認證。如果 FIDO 安全金鑰的認證狀態或等級發生變更，除非裝置已取消註冊並再次註冊以取得更新的認證資訊，否則其將不會更新。

L1、L2、L3、L4 的可能值

在此範例中，您註冊安全金鑰並擷取裝置的 FIPS-140-3 Level 3 認證。

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L3"
                }
            }
        }
    ]

}
```

IAM：RegisterSecurityKey

使用字串運算子。

檢查 MFA 裝置啟用的目前狀態。

可能值為 `Create` 或 `Activate`。

在此範例中，您註冊安全金鑰並擷取裝置的 FIPS-140-3 Level 1 認證。

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
```

```
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L1"
                }
            }
        }
    ]

}
```

IAM：OrganizationsPolicyId

使用字串運算子。

檢查具有指定 AWS Organizations ID 的策略是否符合請求中使用的策略。如要檢視使用此條件鍵的範例 IAM 政策，請參閱 IAM：檢視 Organizations 政策上次存取的服務資訊。

IAM：PassedToService

使用字串運算子。

指定可以將角色傳遞到的服務的服務主體。此條件索引鍵僅適用於策略中的PassRole動作。不能用來限制任何其他動作。

當您在政策中使用此條件鍵時，請使用服務主體指定服務。服務主體是可以在政策的 `Principal` 元素中指定的服務名稱。這是通常的格式：`SERVICE_NAME_URL.amazonaws.com`。

您可以使用 `iam:PassedToService` 限制使用者，以便他們只能將角色傳遞給特定服務。例如，使用者可以建立信任代表他們 CloudWatch 將日誌資料寫入 Amazon S3 儲存貯體的服務角色。然後，使用者必須將許可政策和信任政策連接到新的服務角色。在這種情況下，信任政策必須在 `cloudwatch.amazonaws.com` 元素中指定 `Principal`。若要檢視允許使用者將角色傳遞給的策略 CloudWatch，請參閱IAM：將 IAM 角色傳遞給特定 AWS 服務。

透過使用此條件鍵，您可以確保使用者僅為您指定的服務建立服務角色。例如，如果具有上述政策的使用者嘗試為 Amazon EC2 建立服務角色，操作將會失敗。發生失敗的原因是使用者沒有將角色傳遞至 Amazon EC2 的許可。

有時您會將角色傳遞給服務，然後將角色傳遞給不同服務。`iam:PassedToService` 僅包含擔任角色的最終服務，而不是傳遞角色的中繼服務。

> **Note**
>
> 有些服務不支援此條件鍵。

IAM：PermissionsBoundary

使用 ARN 運算子。

檢查指定的政策連接為 IAM 主體資源上的許可界限。如需詳細資訊，請參閱IAM 實體的許可界限

iam:PolicyARN

使用 ARN 運算子。

在涉及受管政策的請求中檢查託管政策的 Amazon Resource Name (ARN) 。如需詳細資訊，請參閱 控制對政策的存取。

iam:ResourceTag/#名

使用字串運算子。

檢查連接至身分資源 (使用者或角色) 的標籤是否符合指定的鍵名稱和值。

> **Note**
>
> IAM 和 AWS STS 支援 `iam:ResourceTag` IAM 條件金鑰和`aws:ResourceTag`全域條件金鑰。

您可以將自訂屬性以鍵/值組的形式新增至 IAM 資源。如需有關 IAM 資源標籤的詳細資訊，請參閱 the section called “標記 IAM 資源”。您可以使用 `ResourceTag` 控制對 AWS 資源的存取，包括 IAM 資源。但是，由於 IAM 不支援群組的標籤，因此您無法使用標籤來控制群組的存取。

此範例會示範如何建立身分型政策，允許刪除具有 **`status=terminated`** 標籤的使用者。若要使用此政策，請將範例政策中的`########`取代為您自己的資訊。然後，遵循建立政策或編輯政策中的指示進行操作。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:DeleteUser",
        "Resource": "*",
        "Condition": {"StringEquals": {"iam:ResourceTag/status": "terminated"}}
    }]
}
```

AWS OIDC 聯盟的可用金鑰

您可以使用 OIDC 聯盟，將臨時安全登入資料提供給已透過 OpenID Connect 相容身分提供者 (IdP) 驗證的使用者提供給您帳戶中的 IAM OpenID Connect (OIDC) 身分提供者。 AWS 這些供應商的例子包括 GitHub, Amazon Cognito, Login with Amazon, 和谷歌. 您可以使用來自您自己 IdP 的身分權杖和存取權杖，以及授予 Amazon 彈性 Kubernetes 服務工作負載的服務帳戶權杖。

您可以使用 AWS OIDC 條件內容金鑰來撰寫原則，以限制聯合身分使用者存取與特定提供者、應用程式或使用者相關聯的資源。這些鍵通常用於角色的信任政策。使用 OIDC 提供者 (`token.actions.githubusercontent.com`) 名稱，後面接著宣告 (): 來定義條件索引鍵。`:aud` **`token.actions.githubusercontent.com:aud`**

某些 OIDC 聯合條件金鑰可用於角色工作階段來授權資源存取權。如果 [可在工作階段中使用] 欄中的值為 [是]，您可以在策略中使用這些條件索引鍵來定義允許使用者在其他 AWS 服務中存取的項目。當宣告無法在工作階段中使用時，OIDC 條件內容金鑰只能用於初始AssumeRoleWithWebIdentity驗證的角色信任原則中。

選取您的 IdP 以查看 IdP 中的宣告如何對應至中的 IAM 條件內容金鑰。 AWS

Default

預設會列出標準 OIDC 宣告，以及它們如何對應至中的 AWS STS 條件內容索引鍵。 AWS您可以使用這些金鑰來控制角色的存取。若要這麼做，請將AWS STS 條件索引鍵與 IdP JWT 宣告欄中的值進行比較。如果您的 IdP 未列在索引標籤選項中，請使用此對應。

GitHub 操作工作流程和 Google 是在其 OIDC JWT ID IdPs 令牌中使用默認實現的一些示例。

AWS STS 條件鍵	IdP JWT 索賠	工作階段中可用
amr	amr	是

AWS STS 條件鍵	IdP JWT 索賠	工作階段中可用
aud	阿茲普 如果未為設定任何值azp，則aud條件鍵會對應至aud宣告。	是
email	email	否
oaud	aud	否
sub	sub	是

如需搭配使用 OIDC 條件內容索引鍵的詳細資訊 GitHub，請參閱。設定 GitHub OIDC 身分識別提供者的角色如需有關 Google aud 和 azp 欄位的詳細資訊，請參閱 Google Identity Platform OpenID Connect 指南。

amr

使用字串運算子。鍵是多值的，這表示您使用條件設定運算子在政策中對其進行測試。

範例：token.actions.githubusercontent.com:amr

「驗證方法參照」包含使用者的登入資訊。鍵可以包含以下值：

- 如果使用者未經身分驗證，則鍵只包含 unauthenticated。
- 如果使用者經過驗證，索引鍵會包含呼叫 (accounts.google.com) 中使用的登入提供者的值authenticated和名稱。

aud

使用字串運算子。

範例：

- accounts.google.com:aud
- token.actions.githubusercontent.com:aud

使用aud條件索引鍵來確認對象是否符合您在策略中指定的對象。您可以使用 aud 金鑰搭配相同身分識別提供者的子金鑰。

此條件索引鍵是從下列權杖欄位中設定的：

- 未設定 aud 欄位時，您應用程式的 OAuth 2.0 Google 用戶端 ID 的 azp。設定 azp 欄位時，aud 欄位符合 accounts.google.com:oaud 條件鍵。
- 設定 azp 欄位時為 azp。這可能發生於混合式應用程式，其中，Web 應用程式和 Android 應用程式具有不同的 OAuth 2.0 Google 用戶端 ID，但共用相同的 Google API 專案。

當您使用 accounts.google.com:aud 條件鍵撰寫政策時，您必須知道應用程式是否為設定 azp 欄位的混合式應用程式。

azp Field Not Set

下列範例政策適用於未設定 azp 欄位的非混合式應用程式。在這種情況下，Google ID 權杖 aud 欄位值符合 accounts.google.com:aud 和 accounts.google.com:oaud 條件鍵值。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "accounts.google.com:aud": "aud-value",
                    "accounts.google.com:oaud": "aud-value",
                    "accounts.google.com:sub": "sub-value"
                }
            }
        }
    ]
}
```

azp Field Set

下列範例政策適用於已設定 azp 欄位的混合式應用程式。在這種情況下，Google ID 權杖 aud 欄位值只符合 accounts.google.com:oaud 條件鍵值。azp 欄位值符合 accounts.google.com:aud 條件鍵值。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "accounts.google.com:aud": "azp-value",
                    "accounts.google.com:oaud": "aud-value",
                    "accounts.google.com:sub": "sub-value"
                }
            }
        }
    ]
}
```

email

使用字串運算子。

範例：`accounts.google.com:email`

此條件金鑰會驗證使用者的電子郵件地址。此聲明的值可能不是此帳戶的唯一值，並且可能會隨著時間而改變，因此您不應使用此值作為驗證您的使用者記錄的主要識別碼。

oaud

使用字串運算子。

範例：`accounts.google.com:oaud`

此鍵指定此 ID 令牌旨在用於的其他對象（aud）。必須是您應用程式的其中一個 OAuth 2.0 用戶端 ID。

sub

使用字串運算子。

範例：

- `accounts.google.com:sub`
- 令牌. 動作內容. COM: 子

使用這些金鑰來確認主旨是否符合您在原則中指定的主旨。您可以搭配相同身分提供者的 sub 鍵來使用 aud 鍵。

在下列角色信任原則中，sub條件索引鍵會將角色限制為名為的 GitHib 分支demo。

```
{
   "Version": "2012-10-17",
   "Statement": [
      "Condition": {
         "StringEquals": {
            "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
            "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-
repo:ref:refs/heads/demo"
         }
      }
   ]
}
```

Amazon Cognito

此索引標籤說明 Amazon Cognito 如何將 OIDC 宣告對應至中的內容 AWS STS 金鑰。 AWS您可以使用這些金鑰來控制角色的存取。若要這麼做，請將AWS STS 條件索引鍵與 IdP JWT 宣告欄中的值進行比較。

對於 Amazon Cognito 使用的角色，金鑰的定義方式使用宣告後的 `cognito-identity.amazonaws.com`。

如需身分集區宣告對應的詳細資訊，請參閱 Amazon Cognito 開發人員指南中的預設提供者對應。如需有關使用者集區宣告對應的詳細資訊，請參閱 Amazon Cognito 開發人員指南中的使用 ID 權杖。

AWS STS 條件鍵	IdP JWT 索賠	工作階段中可用
amr	amr	是
aud	aud	是
oaud	aud	否
sub	sub	是

amr

使用字串運算子。鍵是多值的，這表示您使用條件設定運算子在政策中對其進行測試。

範例 — cognito-identity.amazonaws.com:amr

「驗證方法參照」包含使用者的登入資訊。鍵可以包含以下值：

- 如果使用者未經身分驗證，則鍵只包含 unauthenticated。
- 如果使用者經過驗證，索引鍵會包含呼叫 (cognito-identity.amazonaws.com) 中使用的登入提供者的值authenticated和名稱。

例如，Amazon Cognito 角色的信任政策中的下列條件會測試使用者是否未經驗證。

```
"Condition": {
  "StringEquals":
    { "cognito-identity.amazonaws.com:aud": "us-east-2:identity-pool-id" },
  "ForAnyValue:StringLike":
    { "cognito-identity.amazonaws.com:amr": "unauthenticated" }
}
```

aud

使用字串運算子。

範例 — cognito-identity.amazonaws.com:aud

對您的使用者進行身分驗證的使用者集區應用程式用戶端。Amazon Cognito 會在存取權杖 client_id 宣告中呈現相同的值。

oaud

使用字串運算子。

範例 — cognito-identity.amazonaws.com:oaud

對您的使用者進行身分驗證的使用者集區應用程式用戶端。Amazon Cognito 會在存取權杖 client_id 宣告中呈現相同的值。

sub

使用字串運算子。

範例 — cognito-identity.amazonaws.com:sub

已驗證使用者的唯一識別碼 (UUID) 或主體。使用者名稱在您的使用者集區中可能不是唯一的。sub 聲明是識別給定用戶的最佳方法。您可以搭配相同身分提供者的 sub 鍵來使用 aud 鍵。

```
{
   "Version": "2012-10-17",
   "Statement": [
      "Condition": {
         "StringEquals": {
            "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
            "cognito-identity.amazonaws.com:sub": [
               "us-east-1:12345678-1234-1234-1234-123456790ab",
               "us-east-1:98765432-1234-1234-1243-123456790ab"
            ]
         }
      }
   ]
}
```

Login with Amazon

此索引標籤說明如何 Login with Amazon 對應 OIDC 宣告以 AWS STS 調節內容金鑰。 AWS您可以使用這些金鑰來控制角色的存取。若要這麼做，請將AWS STS 條件索引鍵與 IdP JWT 宣告欄中的值進行比較。

AWS STS 條件鍵	IdP JWT 索賠	工作階段中可用
app_id	申請識別碼	是
sub	使用者 ID	是
user_id	使用者 ID	是

應用程式識別碼

使用字串運算子。

範例 — `www.amazon.com:app_id`

此索引鍵會指定與其他身分識別提供者所使用之aud欄位相符的對象內容。

sub

使用字串運算子。

範例 — www.amazon.com:sub

此機碼會驗證使用者識別碼是否符合您在策略中指定的使用者識別碼。您可以搭配相同身分提供者的 sub 鍵來使用 aud 鍵。

使用者識別碼

使用字串運算子。

範例 — www.amazon.com:user_id

此索引鍵會指定與其他身分識別提供者所使用之aud欄位相符的對象內容。您可以將user_id金鑰與相同身分識別提供者的id金鑰搭配使用。

Facebook

此標籤說明 Facebook 如何將 OIDC 宣告對應至中的 AWS STS 條件內容索引鍵。 AWS您可以使用這些金鑰來控制角色的存取。若要這麼做，請將AWS STS 條件索引鍵與 IdP JWT 宣告欄中的值進行比較。

AWS STS 條件鍵	IdP JWT 索賠	工作階段中可用
app_id	申請識別碼	是
id	id	是

應用程式識別碼

使用字串運算子。

範例 — graph.facebook.com:app_id

此金鑰會驗證對象內容是否符合其他身分識別提供者所使用的aud欄位。

id

使用字串運算子。

範例 — `graph.facebook.com:id`

此機碼會驗證應用程式 (或網站) ID 與您在策略中指定的 ID 相符。

有關 OIDC 聯盟的更多資訊

- Amazon Cognito 使用者指南
- OIDC 聯盟

SAML AWS STS 聯合身分的可用鍵

如果您使用 AWS Security Token Service (AWS STS) 使用 SAML 型聯合，您可以在原則中包含其他條件索引鍵。

SAML 角色信任政策

在角色的信任政策中，您可以包含以下鍵，這些鍵可幫助您確定是否允許呼叫者擔任該角色。除了 `saml:doc` 外，所有值均來自 SAML 聲明。當您建立或編輯具有條件的政策時，在 IAM 主控台視覺化編輯器中可使用清單中的所有項目。標示為 `[]` 的項目可以具有一個值，該值是指定類型的清單。

saml:aud

使用字串運算子。

顯示 SAML 聲明的端點 URL。此鍵值來自聲明中的 `SAML Recipient` 欄位，而不是 `Audience` 欄位。

saml:commonName[]

使用字串運算子。

這是 `commonName` 屬性。

saml:cn[]

使用字串運算子。

這是 `eduOrg` 屬性。

saml:doc

使用字串運算子。

這代表用來擔任該角色的主體。格式為*#####*/*provider-friendly-name*，例如。123456789012/SAMLProviderNameaccount-ID 值是指擁有 SAML 提供者的帳戶。

saml:edupersonaffiliation[]

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonassurance[]

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonentitlement[]

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonnickname[]

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonorgdn

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonorgunitdn[]

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonprimaryaffiliation

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonprimaryorgunitdn

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonprincipalname

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersonscopedaffiliation[]

使用字串運算子。

這是 eduPerson 屬性。

saml:edupersontargetedid[]

使用字串運算子。

這是 eduPerson 屬性。

saml:eduorghomepageuri[]

使用字串運算子。

這是 eduOrg 屬性。

saml:eduorgidentityauthnpolicyuri[]

使用字串運算子。

這是 eduOrg 屬性。

saml:eduorglegalname[]

使用字串運算子。

這是 eduOrg 屬性。

saml:eduorgsuperioruri[]

使用字串運算子。

這是 eduOrg 屬性。

saml:eduorgwhitepagesuri[]

使用字串運算子。

這是 eduOrg 屬性。

saml:givenName[]

使用字串運算子。

這是 `givenName` 屬性。

saml:iss

使用字串運算子。

發行者，由 URN 代表。

saml:mail[]

使用字串運算子。

這是 `mail` 屬性。

saml:name[]

使用字串運算子。

這是 `name` 屬性。

saml:namequalifier

使用字串運算子。

以 SAML 提供者易記名稱為基礎的雜湊值。該值是下列值的串連，依照順序且以 '/' 字元區隔：

1. `Issuer` 回應值 (`saml:iss`)
2. `AWS` 帳戶 ID
3. IAM 中 SAML 提供者的易記名稱 (ARN 的最後一個部分)

帳戶 ID 與 SAML 提供者的易記名稱的串聯可作為鍵 `saml:doc` 供 IAM 政策使用。如需詳細資訊，請參閱 單獨辨識以 SAML 為基礎的聯合身分中的使用者。

saml:organizationStatus[]

使用字串運算子。

這是 `organizationStatus` 屬性。

saml:primaryGroupSID[]

使用字串運算子。

這是 primaryGroupSID 屬性。

saml:sub

使用字串運算子。

這是該陳述的主題，其中包含單獨辨識組織中某個使用者的值 (例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3)。

saml:sub_type

使用字串運算子。

此鍵可以具有值 persistent、transient 或者由 SAML 聲明中使用的 Format 與 Subject 元素的完整 NameID URI 組成。persistent 的值表示 saml:sub 中的值對於工作階段之間的使用者是相同的。如果值為 transient，則使用者在每個工作階段中擁有不同的 saml:sub 值。如需 NameID 元素的 Format 屬性的詳細資訊，請參閱 設定驗證回應的 SAML 宣告。

saml:surname[]

使用字串運算子。

這是 surnameuid 屬性。

saml:uid[]

使用字串運算子。

這是 uid 屬性。

樣本：500 倍 UniqueIdentifier []

使用字串運算子。

這是 x500UniqueIdentifier 屬性。

如需 eduPerson 和 eduOrg 屬性的一般資訊，請參閱 REFEDS Wiki 網站。如需 eduPerson 屬性的清單，請參閱 eduPerson 物件類別規格 (201602)。

類型為清單的條件鍵可包括多個值。若要在清單值的政策中建立條件，可以使用設定運算子 (ForAllValues、ForAnyValue)。例如，若要允許關係為「教職員」或「員工」 (但不是「學生」) 的任何使用者，您可以使用如下條件：

```
"Condition": {
```

```
    "ForAllValues:StringLike": {
      "saml:edupersonaffiliation":[ "faculty", "staff"]
    }
}
```

跨服務 SAML 型聯合內容索引鍵 AWS STS

某些 SAML 型聯合條件金鑰可用於後續要求中，以授權其他服務和AssumeRole呼叫中的 AWS 作業。當聯合主參與者擔任其他角色時，可在角色信任原則中使用這些條件金鑰，以及在來自其他 AWS 服務的資源策略中，以授權同盟主參與者存取資源時，這些條件金鑰可用於角色信任原則。如需使用這些金鑰的詳細資訊，請參閱關於以 SAML 2.0 為基礎的聯合。

選取條件鍵以查看描述。

- saml:namequalifier
- saml:sub
- saml:sub_type

> **Note**
>
> 在初始外部身分提供者 (IdP) 驗證回應之後，沒有其他以 SAML 為基礎的聯合條件鍵可供使用。

可用的按鍵 AWS STS

對於假定使用 AWS Security Token Service (AWS STS) 操作的角色，您可以在 IAM 角色信任政策中使用下列條件金鑰。

saml:sub

使用字串運算子。

這是該陳述的主題，其中包含單獨辨識組織中某個使用者的值 (例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3)。

STS: AWSServiceName

使用字串運算子。

使用此鍵來指定可以使用持有人權杖的服務。當您在政策中使用此條件鍵時，請使用服務主體指定服務。服務主體是可以在政策的 `Principal` 元素中指定的服務名稱。例如，`codeartifact.amazonaws.com`是 AWS CodeArtifact 服務主體。

某些 AWS 服務需要您具有獲得 AWS STS 服務承載令牌的權限，然後才能以編程方式訪問其資源。例如， AWS CodeArtifact 需要主體使用持有人權杖來執行某些操作。`aws codeartifact get-authorization-token` 命令會傳回一個持有人權杖。然後，您可以使用承載令牌來執行 AWS CodeArtifact 操作。如需有關持有人權杖的詳細資訊，請參閱使用持有人權杖。

可用性 – 此鍵會呈現在取得持有人權杖的請求中。您不能直接調用以 AWS STS 獲取承載令牌。當您在其他服務中執行某些操作時，服務會代表您請求持有人權杖。

您可以使用此條件鍵來允許主體取得持有人權杖，以與特定服務搭配使用。

STS: DurationSeconds

適用於數字運算子。

使用此索引鍵可指定主體在取得 AWS STS 承載權杖時可使用的持續時間 (以秒為單位)。

某些 AWS 服務需要您具有獲得 AWS STS 服務承載令牌的權限，然後才能以編程方式訪問其資源。例如， AWS CodeArtifact 需要主體使用持有人權杖來執行某些操作。`aws codeartifact get-authorization-token` 命令會傳回一個持有人權杖。然後，您可以使用承載令牌來執行 AWS CodeArtifact 操作。如需有關持有人權杖的詳細資訊，請參閱使用持有人權杖。

可用性 – 此鍵會呈現在取得持有人權杖的請求中。您不能直接調用以 AWS STS 獲取承載令牌。當您在其他服務中執行某些操作時，服務會代表您請求持有人權杖。 AWS STS assume-role 操作不會呈現鍵。

STS: ExternalId

使用字串運算子。

使用此鍵來請求主體在擔任 IAM 角色時提供特定的識別碼。

可用性 — 當主體提供外部 ID，同時使用 AWS CLI 或 AWS API 假定角色時，此金鑰會出現在要求中。

當您在其他帳戶擔任角色時，可能需要此唯一識別符。若帳戶管理員 (該角色所屬的帳戶) 提供給您外部 ID，請將該數值填入 `ExternalId` 參數。該值可為任何字串，例如密碼短語或帳號。外部 ID 的主要功能是解決並防止「混淆代理人」問題。如需有關外部 ID 和混淆代理人問題的詳細資訊，請參閱 將 AWS 資源存取權授予第三方時，如何使用外部 ID。

ExternalId 值必須最少為 2 個字元，最多為 1,224 個字元。該值必須為英數字元，且不包含空格。也可以包含下列符號：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、小老鼠 (@)、冒號 (:)、正斜線 (/) 和連字號 (-)。

sts：RequestContext/上下文鍵

使用字串運算子。

使用此鍵，可將內嵌在請求中所傳遞的受信任權杖發行者簽署之內容聲明的工作階段內容鍵/值對，與角色信任政策中指定的內容鍵/值進行比較。

可用性 — 當在要求參數中提供內容宣告，同時假設使用 AWS STS AssumeRole API 作業的角色時，此金鑰會出現在ProvidedContexts要求中。

此內容鍵會格式化為 "sts:RequestContext/context-key":"context-value"，其中位 context-key 和 context-value 為內容鍵/值對。如果請求中傳遞簽署內容聲明中內嵌了多個內容鍵，則每一組鍵/值對都有一個內容鍵。您必須為角色信任政策中的 sts:SetContext 動作授予許可，才能允許主體在產生的工作階段權杖中設定內容鍵。若要進一步了解可與此金鑰搭配使用的受支援 IAM 身分中心內容金鑰，請參閱使用AWS IAM Identity Center 者指南中的 IAM 身分中心的AWS STS 條件金鑰。

您可以在角色信任政策中使用此鍵，以便在使用者或其屬性擔任角色時，能根據使用者或其屬性實行精細的存取控制。例如，您可以將 Amazon Redshift 設定為 IAM Identity Center 應用程式，以代表您的員工或聯合身分存取 Amazon S3 資源。

下列角色信任政策允許 Amazon Redshift 服務主體擔任帳戶 111122223333 中的角色。只要 identitystore:UserId 內容鍵值集為 1111-22-3333-44-5555，它也會授予許可，允許 Amazon Redshift 服務主體在請求中設定內容鍵。假設角色之後，活動會出現在AdditionalEventData元素內的 AWS CloudTrail 記錄檔中，其中包含由前後關聯提供者在假設角色請求中設定的工作階段前後關聯索引鍵值配對。當不同的主體使用角色時，這可讓系統管理員更容易區分角色工作階段。索引鍵值配對是由指定的內容提供者所設定，而不是由 AWS CloudTrail 或 AWS STS設定。這可讓內容提供者控制 CloudTrail 記錄檔和工作階段資訊中包含的內容。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
```

```
            },
            "Action": [
                "sts:AssumeRole",
                "sts:SetContext"
            ],
            "Condition": {
                "ForAllValues:ArnEquals": {
                    "sts:RequestContextProviders": [
                        "arn:aws:iam::aws:contextProvider/IdentityCenter"
                    ]
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "sts:RequestContext/identitystore:UserId":
 "1111-22-3333-44-5555"
                }
            }
        }
    ]
}
```

STS: RequestContextProviders

使用 ARN 運算子。

使用此鍵，可將請求中的內容提供者 ARN 與角色信任政策中指定的內容提供者 ARN 進行比較。

可用性 — 當在要求參數中提供內容宣告，同時假設使用 AWS STS `AssumeRole` API 作業的角色時，此金鑰會出現在`ProvidedContexts`要求中。

下列範例條件會檢查請求中傳遞的內容提供者 ARN 是否符合角色信任政策條件中指定的 ARN。

```
      "Condition": {
        "ForAllValues:ArnEquals": {
          "sts:RequestContextProviders": [
            "arn:aws:iam::aws:contextProvider/IdentityCenter"
          ]
        }
      }
```

STS: RoleSessionName

使用字串運算子。

使用此鍵，比較使用政策中指定的值擔任角色時，主體指定的工作階段名稱。

可用性 — 當主體使用 AWS Management Console、任何 assume-role CLI 命令或任何 AWS STS `AssumeRole` API 作業擔任角色時，此金鑰會出現在要求中。

您可以在角色信任政策中使用此鍵，請求您的使用者在擔任角色時提供特定的工作階段名稱。例如，您可以請求 IAM 使用者將自己的使用者名稱指定為其工作階段名稱。在 IAM 使用者擔任角色之後，活動會出現在 AWS CloudTrail 日誌中，其工作階段名稱與其使用者名稱相符。當不同的主體使用角色時，這可讓系統管理員更容易區分角色工作階段。

下列角色信任政策會請求 111122223333 帳戶中的 IAM 使用者在擔任角色時，提供其 IAM 使用者名稱作為工作階段名稱。使用條件鍵中的 `aws:username` 條件變數強制執行此需求。此政策允許 IAM 使用者擔任政策所連接的角色。此政策不允許使用暫時憑證的任何人擔任該角色，因為 `username` 變數僅供 IAM 使用者使用。

> **Important**
>
> 您可以使用任何單一值條件鍵作為變數。您無法使用多重值條件鍵做為變數。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RoleTrustPolicyRequireUsernameForSessionName",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
            "Condition": {
                "StringLike": {"sts:RoleSessionName": "${aws:username}"}
            }
        }
    ]
}
```

當管理員檢視動作的 AWS CloudTrail 記錄檔時，他們可以將工作階段名稱與其帳戶中的使用者名稱進行比較。在下列範例中，名為 `matjac` 的使用者使用名為 `MateoRole` 的角色執行作業。然後，管理員可以聯絡 Mateo Jackson，他擁有名為 `matjac` 的使用者。

```
    "assumedRoleUser": {
```

```
        "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:matjac",
        "arn": "arn:aws:sts::111122223333:assumed-role/MateoRole/matjac"
    }
```

如果您允許使用角色進行跨帳戶存取，則某個帳戶中的使用者可以在另一個帳戶中擔任角色。中列出之假定角色使用者的 ARN CloudTrail 包含角色所在的帳戶。它不包括擔任角色的使用者帳戶。使用者僅在帳戶中是唯一的。因此，我們建議您只使用此方法來 CloudTrail 檢查您所管理帳戶中使用者所承擔之角色的記錄。您的使用者可能會在多個帳戶中使用相同的使用者名稱。

STS: SourceIdentity

使用字串運算子。

使用此鍵，比較使用政策中指定的值擔任角色時，主體指定的來源身分。

可用性 — 當主體提供來源識別，同時使用任何 AWS STS assume-role CLI 命令或 AWS STS `AssumeRole` API 作業假設角色時，此金鑰會出現在要求中。

您可以在角色信任政策中使用此鍵，請求您的使用者在擔任角色時設定特定的來源身分。例如，您可以要求人力或聯合身分來指定來源身分的值。您可以將身分提供者 (IdP) 設定為使用與使用者相關聯的其中一個屬性，例如使用者名稱或電子郵件作為來源身分。然後，IdP 會將來源識別做為其傳送至的宣告或宣告中的屬性傳遞。 AWS來源身分屬性的值可識別擔任該角色的使用者或應用程式。

在使用者擔任該角色之後，活動會出現在具有已設定來源身分值的 AWS CloudTrail 日誌中。這可讓系統管理員更容易判斷使用中的角色執行動作的人員或哪些動作 AWS。您必須對 `sts:SetSourceIdentity` 動作授予許可，允許身分以設定來源身分。

與 `sts:RoleSessionName` 不同，在設定來源身分之後，就無法變更值。這會出現在由角色以來源身分所採取的所有動作的請求內容中。當您使用工作階段憑證來擔任另一個角色時，此值仍然存在於後續角色工作階段。從另一個角色取得及擔任角色稱為角色鏈結。

您可以使用`aws:SourceIdentity`全域條件索引鍵，根據後續要求中的來源識別值進一步控制 AWS 資源的存取。

下列角色信任政策允許 IAM 使用者 `AdminUser` 擔任帳戶 `111122223333` 中的角色。它也會對 `AdminUser` 許可授予來設定來源身分，只要來源身分識別設定為 `DiegoRamirez`。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowAdminUserAssumeRole",
            "Effect": "Allow",
            "Principal": {"AWS": " arn:aws:iam::111122223333:user/AdminUser"},
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {"sts:SourceIdentity": "DiegoRamirez"}
            }
        }
    ]
}
```

若要進一步了解有關使用來源身分的詳細資訊，請參閱 監控並控制使用擔任角色所採取的動作。

STS: TransitiveTagKeys

使用字串運算子。

使用此鍵，可將請求中的可轉移工作階段標籤鍵與政策中指定的內容進行比較。

可用性 – 當您使用臨時安全憑證提出請求時，此鍵會呈現在請求中。這些包括使用任何 assume-role 操作或 `GetFederationToken` 操作所建立的憑證。

當您使用暫時安全憑證提出請求時，請求內容會包含 `aws:PrincipalTag` 內容鍵。此鍵包含工作階段標籤的清單、可轉移工作階段標籤和角色標籤。可轉移工作階段標籤是在您使用工作階段憑證擔任另一個角色時，保存到所有後續工作階段的標籤。從另一個角色取得及擔任角色稱為角色鏈結。

您可以在政策中使用此條件鍵，要求在擔任角色或與使用者聯合身分時將特定工作階段標籤設為可轉移。

AWS 服務的動作、資源和條件索引鍵

每個 AWS 服務都可以定義動作、資源和條件內容金鑰，以便在 IAM 政策中使用。如需 AWS 服務及其動作、資源和條件內容索引鍵的清單，請參閱服務授權參考中的動作、資源和條件索引鍵。

進一步了解有關 IAM 的資源

IAM 是一種豐富的產品，您將找到許多資源來幫助您進一步了解 IAM 如何協助您保護 AWS 帳戶 和資源的安全。

主題

身分

請諮詢這些資源以建立、管理和使用身分。

- 管理 AM Identity Center 中的身分 - 有關在 IAM Identity Center 中建立使用者和群組的程序資訊。
- IAM 身分 (使用者、使用者群組和角色) - 深入討論使用者、群組和角色。

憑證 (密碼、存取金鑰和 MFA 裝置)

查看以下指南，以管理您和 IAM 使用者的密碼、存取金鑰 AWS 帳戶 和 MFA 裝置。

- 管理使用者密碼 AWS – 描述管理您的帳戶中 IAM 使用者密碼的選項。
- 管理 IAM 使用者的存取金鑰 - 描述存取金鑰的運作方式，以及您可以如何使用它們對 AWS發出程式化呼叫。我們建議您先考慮其他更安全的方法來替代存取金鑰。如需詳細資訊，請參閱 AWS 一般參考 指南中的長期存取金鑰的考量事項和替代方案。
- 在中使用多因素身份驗證（MFA） AWS – 描述如何設定您的帳戶和 IAM 使用者，要求在登入前允許在裝置上產生密碼和一次性使用程式碼。(有時稱為雙重身分驗證)。

如需有關您用於存取 Amazon Web Services 的憑證類型的一般資訊，請參閱 AWS 一般參考 指南中的 AWS 安全憑證。

許可和政策

了解 IAM 政策的內部工作原理，並找到有關授予許可的最佳方式的秘訣：

- IAM 中的政策和許可 – 介紹用於定義許可的政策語言。說明如何將許可連接到使用者或群組，或者對於某些 AWS 產品，如何連接到資源本身。
- IAM JSON 政策元素參考 – 提供每個政策語言元素的描述和範例。
- 驗證 IAM 政策 – 尋找用於 JSON 政策驗證的資源。
- 以身分為基礎的 IAM 政策範例— 顯示各種 AWS 產品中一般工作的原則範例。
- AWS 政策產生器 – 透過從清單中選擇產品和動作來建立自訂政策。
- IAM 政策模擬器 — 測試政策是否允許或拒絕特定請求 AWS。

聯合與委派

您可以針對其他地方已驗證 (登入) AWS 帳戶 的使用者授予對中資源的存取權。這些使用者可以是另一個 AWS 帳戶 (稱為委派) 中的 IAM 使用者、透過組織登入程序驗證的使用者，或是來自網際網路身分供應商的使用者，例如使用 Amazon、Facebook、Google 或任何其他 OpenID Connect (OIDC) 相容身分提供者登入。在這些情況下，使用者會取得臨時安全登入資料以存取 AWS 資源。

- IAM 教學課程：使用 IAM 角色將存取許可委派給不同 AWS 帳戶 – 指導您透過授予另一個 AWS 帳戶中的 IAM 使用者的跨帳戶存取權。
- 暫時性憑證的常見案例— 描述在外部驗證 AWS 之後，可以聯合使用者的 AWS方式。

IAM 和其他 AWS 產品

大多數 AWS 產品都與 IAM 整合，因此您可以使用 IAM 功能來協助保護這些產品中資源的存取權。以下資源討論一些最熱門 AWS 產品的 IAM 和安全性。如需有關使用 IAM 的產品的完整清單，包括指向每個產品的詳細資訊的連結，請參閱 AWS 與 IAM 搭配使用的服務。

搭配使用 IAM 與 Amazon EC2

- 控制對 Amazon EC2 資源的存取 – 描述如何使用 IAM 功能來允許使用者管理 Amazon EC2 執行個體、磁碟區等。
- 使用執行個體設定檔— 描述如何使用 IAM 角色安全地為在 Amazon EC2 執行個體上執行且需要存取其他 AWS 產品的應用程式提供登入資料。

搭配使用 IAM 與 Amazon S3

- 管理 Amazon S3 資源的存取許可 – 討論儲存貯體和物件的 Amazon S3 安全模型，其中包含 IAM 政策。
- 撰寫 IAM 政策：授予存取 Amazon S3 儲存貯體中的使用者特定資料夾 – 討論如何讓使用者保護 Amazon S3 中自己的資料夾 (有關 Amazon S3 和 IAM 的詳細文章，請選擇部落格文章標題下方的 S3 標籤)。

搭配使用 IAM 與 Amazon RDS

- 使用 AWS Identity and Access Management (IAM) 管理對 Amazon RDS 資源的存取 — 描述如何使用 IAM 控制對資料庫執行個體、資料庫快照等的存取。
- RDS 資源層級許可的入門 – 描述如何使用 IAM 控制存取特定 Amazon RDS 執行個體。

搭配使用 IAM 與 Amazon DynamoDB

- 使用 IAM 控制存取 DynamoDB 資源 – 描述如何使用 IAM 來允許使用者管理 DynamoDB 表格和索引。
- 以下影片 (8:55) 解釋如何為個別 DynamoDB 資料庫項目或屬性 (或兩者) 提供存取控制。

  DynamoDB 的精細存取控制入門

一般安全實務

尋找有關保護和資源安全的最佳方法的專家秘訣 AWS 帳戶 和指引：

- 安全性、身分識別與合規的最佳實務 — 尋找有關如何管理 AWS 帳戶 和產品安全性的資源，包括有關安全性架構、IAM 使用、加密和資料安全等方面的建議。
- Identity and Access Management — AWS Well-Architected 的架構可協助您瞭解在雲端中設計和執行工作負載的關鍵概念、設計原則和架構最佳實務。
- IAM 中的安全最佳實務 – 提供有關如何使用 IAM 協助保護您的 AWS 帳戶 和資源的建議。
- AWS CloudTrail 用戶指南 — 用於 AWS CloudTrail 跟踪對該信息進行的 API 調用的歷史記錄 AWS 並將其存儲在日誌文件中。這有助於您判斷哪些使用者和帳戶存取帳戶中的資源，何時進行呼叫，請求了哪些動作等等。

一般資源

探索下列資源，以進一步了解 IAM 和 AWS.

- IAM 的產品資訊 – 有關 AWS Identity and Access Management 產品的一般資訊。
- AWS re:Post for AWS Identity and Access Management — 造訪 AWS re:Post 以與 AWS 社群討論 IAM 相關的技術問題。

- 課程和工作坊 — 除了可以幫助提高 AWS 技能並獲得實踐經驗的自定進度實驗室之外，還可以鏈接到基於角色和專業課程的鏈接。
- AWS 開發人員中心 — 探索教學課程、下載工具，以及瞭解 AWS 開發人員活動。
- AWS 開發人員工具 — 開發人員工具、SDK、IDE 工具組，以及用於開發和管理 AWS 應用程式的命令列工具的連結。
- 入門資源中心 — 瞭解如何設定 AWS 帳戶、加入 AWS 社群，以及啟動您的第一個應用程式。
- 實作教學課程 — 按照 step-by-step 教學課程啟動您的第一個應用程式 AWS。
- AWS 白皮書 — 完整的技術 AWS 白皮書清單連結，涵蓋架構、安全性和經濟等主題，並由 AWS 解決方案架構師或其他技術專家撰寫。
- AWS Support 中心 — 建立和管理 AWS Support 案例的中心。同時也包含其他實用資源的連結，例如論壇、技術常見問答集、服務健康狀態和 AWS Trusted Advisor.
- AWS Support— 有關資訊的主要網頁 AWS Support one-on-one, 快速回應的支援管道，可協助您在雲端中建置和執行應用程式.
- 聯絡我們 – 查詢有關 AWS 帳單、帳戶、事件、濫用與其他問題的聯絡中心。
- AWS 網站條款 — 有關我們的版權和商標的詳細資訊；您的帳戶、授權和網站存取權限；以及其他主題。

使用 HTTP 查詢請求呼叫 IAM API

目錄

您可以使用查詢 API 以程式設計方式存取 IAM 和 AWS STS 服務。查詢 API 請求為 HTTPS 請求，其一定含有 `Action` 參數，以指示所要執行的動作。IAM 和 AWS STS 支援所有動作的 GET 和 POST 請求。也就是說，API 不會要求您在某些動作上使用 GET，在其他動作上使用 POST。但是，GET 請求受到 URL 的限制大小；雖然此限制取決於瀏覽器，但一般限制為 2048 個位元組。因此，對於需要較大流量的查詢 API 請求，必須使用 POST 請求。

回應為 XML 文件。如需回應的詳細資訊，請參閱 IAM API 參考或 AWS Security Token Service API 參考中的個別動作頁面。

> **Tip**
>
> 您可以使用其中一個 AWS SDK，而不是直接呼叫 IAM 或 AWS STS API 作業。該 AWS 軟件包括庫和示例代碼，用於各種編程語言和平台（Java，紅寶石，網絡，iOS，安卓等）。SDK 提供了一種方便的方式來建立 IAM 和. AWS例如，開發套件會負責的工作諸如以密碼演算法簽署請求 (參閱下面)、管理錯誤以及自動重試請求。如需 AWS SDK 的相關資訊，包括如何下載和安裝這些軟體開發套件，請參閱 Amazon Web Services 工具頁面。

如需 API 動作和錯誤的詳細資訊，請參閱IAM API 參考或AWS Security Token Service API 參考。

端點

IAM 和 AWS STS 每個端點都有一個全球端點：

- (IAM)https://iam.amazonaws.com
- (AWS STS)https://sts.amazonaws.com

> **Note**
>
> AWS STS 除了全域端點之外，還支援將要求傳送至地區端點。在您可以 AWS STS 在某個區域中使用之前，您必須先為您的區域啟動 STS AWS 帳戶。如需啟動其他區域的詳細資訊 AWS STS，請參閱AWS STS 在一個管理 AWS 區域。

如需有關所有服務之 AWS 端點和區域的詳細資訊，請參閱 AWS 一般參考.

必要的 HTTPS

由於查詢 API 會傳回安全憑證等敏感資訊，必須對所有 API 請求使用 HTTPS。

簽署 IAM API 請求

請求必須使用存取金鑰 ID 和私密存取金鑰簽署。強烈建議您不要使用 AWS 帳戶根使用者 登入資料來執行 IAM 的日常作業。您可以使用 IAM 使用者的登入資料，也可以用 AWS STS 來產生臨時安全登入資料。

若要簽署您的 API 要求，我們建議您使用 AWS 簽名版本 4。如需有關使用簽章第 4 版的詳細資訊，請前往 AWS 一般參考 中的 簽章第 4 版簽署程序。

如果需要使用簽章第 2 版，則 AWS 一般參考中提供簽章第 2 版的使用資訊。

如需詳細資訊，請參閱下列內容：

- AWS 安全登入資料。提供有關用於存取之認證類型的一般資訊 AWS。
- IAM 中的安全最佳實務。 提供使用 IAM 服務來協助保護 AWS 資源安全的建議清單。
- IAM 中的暫時安全憑證. 描述如何建立和使用暫時性安全憑證。

IAM 的文件歷史紀錄

下表說明 IAM 的主要文件更新。

變更	描述	日期
AccessAnalyzerServiceRolePolicy：新增許可	IAM Access Analyzer 新增了對權限的支援，可將 IAM 使用者和角色政策相關資訊擷取至AccessAnalyzerServiceRole政策的服務層級許可。	2024年5月30日
AccessAnalyzerServiceRolePolicy：新增許可	IAM 存取分析器新增了許可支援，可將 Amazon EC2 快照的區塊公共存取目前狀態擷取到AccessAnalyzerServiceRole政策的服務層級許可。	2024 年 1 月 23 日
AccessAnalyzerServiceRolePolicy：新增許可	IAM 存取分析器將 DynamoDB 串流和表格新增至政策的服務層級許可。AccessAnalyzerServiceRole	2024年1月11日
AccessAnalyzerServiceRolePolicy：新增許可	IAM 存取分析器將 Amazon S3 目錄儲存貯體新增至AccessAnalyzerServiceRole政策的服務層級許可。	2023 年 12 月 1 日
IAMAccessAnalyzerReadOnlyAccess：新增許可	IAM 存取分析器為 IAM AccessAnalyzer ReadOnly 存取新增許可，讓您檢查政策的更新是否授予其他存取權限。 IAM Access Analyzer 需要此許可，才能對您的政策執行政策檢查。	2023 年 11 月 26 日

IAM Access Analyzer 新增了未使用的存取權分析器	IAM Access Analyzer 簡化了檢查未使用的存取權，引導您達到最低權限。IAM Access Analyzer 會持續分析您的帳戶，以識別未使用的存取權，並建立包含調查結果的集中式儀表板。	2023 年 11 月 26 日
IAM Access Analyzer 新增了自訂政策檢查	IAM Access Analyzer 現在提供自訂政策檢查，可在部署之前驗證 IAM 政策是否符合您的安全標準。	2023 年 11 月 26 日
AccessAnalyzerServiceRolePolicy：新增許可	IAM 存取分析器將 IAM 動作新增至AccessAnalyzerServiceRole政策的服務層級許可，以支援下列動作： • 列出政策的實體 • 產生上次存取的服務詳細資訊 • 列出存取金鑰資訊	2023 年 11 月 26 日
對 60 多項其他服務和動作的最近存取動作資訊和政策產生支援	IAM 現在支援最近存取的動作資訊，並針對 60 多項其他服務產生包含動作層級資訊的政策，以及可取得最近存取資訊的動作清單。	2023 年 11 月 1 日
對 140 多項服務的最近存取的動作資訊支援	IAM 現在會為 140 多項服務提供最近存取的動作資訊，以及可取得最近存取資訊的動作清單。	2023 年 9 月 14 日

支援根使用者和 IAM 使用者使用多台多重要素驗證 (MFA) 裝置	現在，您最多可以為每個使用者新增最多八台 MFA 裝置，包括 FIDO 安全金鑰、使用虛擬驗證器應用程式的以時間為基礎的一次性密碼 (TOTP)，或者硬體 TOTP 權杖。	2022 年 11 月 16 日
IAM Access Analyzer 支援新的資源類型	IAM Access Analyzer 已新增對下列資源類型的支援： • Amazon EBS 磁碟區快照 • Amazon ECR 儲存庫 • Amazon EFS 檔案系統 • Amazon RDS 資料庫快照 • Amazon RDS 資料庫叢集快照 • Amazon SNS 主題	2022 年 10 月 25 日
U2F 棄用和 WebAuthn /FIDO 更新	移除了 U2F 做為 MFA 選項的提及 WebAuthn，並新增了有關 FIDO2 和 FIDO 安全金鑰的資訊。	2022 年 5 月 31 日
IAM 中彈性的更新	新增有關當某個事件中斷 AWS 區域之間的通訊時，維持對 IAM 憑證進行存取的資訊。	2022 年 5 月 16 日
資源的新全域條件金鑰	您現在可以根據帳戶、組織單位 (OU) 或包含您資源的組織來控制 AWS Organizations 對資源的存取。您可以在 IAM 政策中使用 `aws:ResourceAccount` 、`aws:ResourceOrgID` 以及 `aws:ResourceOrgPaths` 全域條件金鑰。	2022 年 4 月 27 日

使用 AWS 軟體開發套件的 IAM 程式碼範例	已新增程式碼範例，說明如何搭配 AWS 軟體開發套件 (SDK) 使用 IAM。這些範例分為程式碼摘錄 (示範如何呼叫個別服務函數) 和範例 (示範如何透過呼叫相同服務中的多個函數來完成特定任務)。	2022 年 4 月 7 日
更新政策評估邏輯流程圖	更新政策評估邏輯流程圖，以及決定是否允許或拒絕帳戶中的請求區段中的相關文字。	2021 年 11 月 17 日
更新安全最佳實務	新增了有關建立系統管理使用者而非使用根使用者憑證的資訊，移除了使用使用者群組指派許可給 IAM 使用者的最佳實務，以及澄清何時使用受管政策而非內嵌政策。	2021 年 10 月 5 日
更新資源型政策的政策評估邏輯主題	新增有關資源型政策和相同帳戶中不同主體類型影響的資訊。	2021 年 10 月 5 日
更新單一值和多重值條件金鑰	現在會更詳細地說明單一值和多重值條件金鑰之間的差異。值類型新增至每個 AWS 全域條件內容金鑰。	2021 年 9 月 30 日
IAM Access Analyzer 支援 Amazon S3 多區域存取點	IAM Access Analyzer 可識別允許公有和跨帳戶存取的 Amazon S3 儲存貯體，包括使用 Amazon S3 多區域存取點。	2021 年 9 月 2 日
AWS 受管策略更新-現有策略的更新	IAM 存取分析器更新了現有的 AWS 受管政策。	2021 年 9 月 2 日

支援更多服務用於動作層級政策產生	IAM 存取分析器可以產生 IAM 政策，其中包含其他服務的動作層級存取活動資 AWS 訊。	2021 年 8 月 24 日
產生跨帳戶追蹤的 IAM 政策	您現在可以使用 IAM Access Analyzer，透過不同帳戶中的追蹤 (例如集中式 AWS CloudTrail AWS Organizations 追蹤)，根據存取活動產生精細的政策。	2021 年 8 月 18 日
其他 IAM Access Analyzer 政策檢查	IAM Access Analyzer 透過新增對 IAM 政策中包含的條件進行驗證的新政策檢查，來延伸政策驗證。這些檢查會分析政策陳述式中的條件區塊，並報告安全性警告、錯誤和建議，以及可採取行動的建議。 IAM Access Analyzer新增了下列政策檢查： • 錯誤 — 無效的服務主體格式 • 錯誤 – 條件中缺少標籤鍵 • 安全性警告 — 拒絕服務 NotAction 的不支援標籤條件金鑰 • 安全性警告 - 使用服務不支援的標籤條件金鑰拒絕 • 安全性警告 - 遺失配對的條件金鑰 • 建議 — 允許服務 NotAction 使用不支援的標籤條件索引鍵 • 建議 - 使用服務不支援的標籤條件金鑰允許	2021 年 6 月 29 日

對更多服務提供最近存取動作支援	您現在可以在 IAM 主控台中檢視關於 IAM 主體上次針對下列服務使用動作的最近存取動作資訊：Amazon EC2、IAM、Lambda 和 Amazon S3 管理動作。您也可以使用 AWS CLI 或 AWS API 擷取資料報告。您可以使用此資訊來辨別不必要的許可，以便您能夠微調您的 IAM 政策以更完善地遵循最低權限的原則。	2021 年 4 月 19 日
監控並控制使用擔任角色所採取的動作	管理員可以將 IAM 角色設定為要求身分傳遞來源身分，其在 AWS CloudTrail中登入。檢閱來源身分資訊可協助管理員判斷曾使用擔任角色工作階段執行動作者。	2021 年 4 月 13 日
根據存取活動產生 IAM 政策	您現在可以使用 IAM Access Analyzer 根據您在 AWS CloudTrail中的存取活動產生精細政策。	2021 年 4 月 7 日
IAM Access Analyzer 政策檢查	IAM Access Analyzer 現在會在政策編寫期間提供超過 100 項政策檢查及可採取行動的建議。	2021 年 3 月 16 日
擴充的政策驗證選項	IAM 主控台、 AWS API 提供擴充的政策驗證，並 AWS CLI 使用 IAM 存取分析器中的政策檢查來協助您編寫安全且功能強大的 JSON 政策。	2021 年 3 月 15 日
標記 IAM 資源	現在，您可以使用標籤鍵值對標籤其他 IAM 資源。	2021 年 2 月 11 日

IAM 使用者的預設密碼政策	如果您未為自己設定自訂密碼政策 AWS 帳戶，IAM 使用者密碼現在必須符合預設 AWS 密碼政策。	2020 年 11 月 18 日
AWS 服務的動作、資源和條件索引鍵頁面已移動	每個 AWS 服務都可以定義動作、資源和條件內容金鑰，以便在 IAM 政策中使用。您現在可以在「AWS 服務授權參考」中找到服務及其動作、資源和條件內容索引鍵的清單。	2020 年 11 月 16 日
IAM 使用者的較長角色工作階段持續時間	IAM 使用者現在可以在中切換角色時擁有較長的角色工作階段持續時間 AWS Management Console，從而減少工作階段到期造成的中斷。使用者會被授予針對角色設定的工作階段持續時間上限，或 IAM 使用者工作階段中的剩餘時間，以較少者為準。	2020 年 7 月 24 日
使用服務配額 (Service Quotas) 來請求 IAM 實體的快速增加	您可以使用 Service Quotas 主控台，為可調整的 IAM 配額請求增加配額。現在，在 Service Quotas 中有些增加會自動核准，且在幾分鐘內您的帳戶便可使用。較大的要求會提交給 AWS Support。	2020 年 6 月 25 日

IAM 中的最近存取資訊現在包括 Amazon S3 管理動作	除了服務最近存取資訊外，您現在還可以在 IAM 主控台中檢視 IAM 主體上次使用 Amazon S3 動作的相關資訊。您也可以使用 AWS CLI 或 AWS API 擷取資料報表。此報告包含主體上次嘗試存取的允許服務和動作，以及何時存取的相關資訊。您可以使用此資訊來辨別不必要的許可，以便您能夠微調您的 IAM 政策以更完善地遵循最低權限的原則。	2020 年 6 月 3 日
新增安全性章節	安全章節可協助您瞭解如何設定 IAM， AWS STS 以及符合安全與合規目標。您也會了解如何使用其他 AWS 服務來協助監控並保護 IAM 資源。	2020 年 4 月 29 日
目錄:RoleSession名稱	您現在可以撰寫根據主體擔任角色時所指定的工作階段名稱而授予許可的政策。	2020 年 4 月 21 日
AWS 登錄頁面更新	在主 AWS 登入頁面登入時，您無法選擇以 AWS 帳戶根使用者 或 IAM 使用者身分登入。當您這麼做時，頁面上的標籤會指出您是否應該提供根使用者電子郵件地址或您的 IAM 使用者資訊。本文件包含更新的螢幕擷取畫面，以協助您了解 AWS 登入頁面。	2020 年 3 月 4 日

AWS：通過AWSService 和 aws：CalledVia 條件鍵	您現在可以寫入政策，限制服務是否能代表 IAM 主體 (使用者或角色) 提出請求。主體向 AWS 服務提出請求時，該服務可能會使用主體的憑證，向其他服務提出後續請求。若任何服務使用主體憑證提出請求，請使用 `aws:ViaAWSService` 條件金鑰來匹配。若特定服務使用主體憑證提出請求，請使用 `aws:CalledVia` 條件金鑰來匹配。	2020 年 2 月 20 日
政策模擬器新增了許可界限的支援	您現在可以使用 IAM 政策模擬器，測試許可界限對 IAM 實體的效果。	2020 年 1 月 23 日
跨帳戶政策評估	您現在可以了解如何 AWS 評估跨帳戶存取的策略。當信任帳戶中的資源包含以資源為基礎的政策，而允許其他帳戶中的主體存取該資源時，就會發生這種情況。兩個帳戶中都必須允許此請求。	2020 年 1 月 2 日
工作階段標籤	現在當您在 AWS STS中擔任角色或聯合使用者時，可以包含標籤。當您執行 `AssumeRole` 或 `GetFederationToken` 操作時，您可以將工作階段標籤傳遞為屬性。當您執行`AssumeRoleWithSAML` 或`AssumeRoleWithWebIdentity` 作業時，您可以將公司識別中的屬性傳遞給 AWS。	2019 年 11 月 22 日

控制 AWS 帳戶 中的群組的存取 AWS Organizations	您現在可以參考 IAM 政策 AWS Organizations 中的組織單位 (OU)。如果您使用 Organizations 將帳戶組織成 OU，即可在授與資源存取權之前，要求主體從屬於特定 OU。主體包括 AWS 帳戶根使用者、IAM 使用者和 IAM 角色。若要執行此作業，請在政策的 `aws:PrincipalOrgPaths` 條件金鑰中指定 OU 路徑。	2019 年 11 月 20 日
上次使用的角色	您現在可以檢視上次使用角色的日期、時間和區域。此資訊也可協助您識別帳戶中未使用的角色。您可以使用 AWS Management Console、 AWS CLI 和 AWS API 來檢視上次使用角色時間的相關資訊。	2019 年 11 月 19 日
更新 Global condition context keys (全域條件內容索引鍵) 頁面	您現在可以了解每個全域條件金鑰包含在請求內容中的時機。您也可以使用頁面目錄 (TOC) 更輕易地導覽至每個金鑰。頁面上的資訊可協助您撰寫更精確的政策。例如，若您的員工搭配 IAM 角色使用聯合，建議您使用 `aws:userId` 金鑰而非 `aws:userName` 金鑰。`aws:userName` 金鑰僅適用於 IAM 使用者而非角色。	2019 年 10 月 6 日

阿巴克 AWS	了解基於屬性的訪問控制（ABAC）如何在 AWS 使用標籤時工作，以及它與傳統 AWS 授權模型的比較。使用 ABAC 教學來了解如何建立和測試政策，允許具備主體標籤的 IAM 角色存取具備相符標籤的資源。此策略可讓個人只檢視或編輯其工作所需的 AWS 資源。	2019 年 10 月 3 日
AWS STS GetAccessKeyInfo 操作	您可以檢閱程式碼中的 AWS 存取金鑰，以判斷金鑰是否來自您擁有的帳戶。您可以使用`aws sts get-access-key-info` AWS CLI 命令或`GetAccessKeyInfo` AWS API 操作傳遞訪問密鑰 ID。	2019 年 7 月 24 日
檢視 IAM 中的 Organizations 服務最近存取的資訊	您現在可以在 IAM 主控台的AWS Organizations區段中檢視 AWS Organizations 實體或政策上次存取的服務資訊。您也可以使用 AWS CLI 或 AWS API 擷取資料報表。這些資料包含 Organizations 帳戶中的主體最後一次嘗試存取哪些允許的服務以及何時存取的相關資訊。您可以使用此資訊來找出不必要的許可，以便您能夠微調 Organizations 政策以更完善地遵循最低權限的原則。	2019 年 6 月 20 日
使用受管政策做為工作階段政策	當您擔任角色時，您現在可以傳遞最多 10 個受管政策 ARN。這可讓您限制該角色臨時憑證的許可。	2019 年 5 月 7 日

AWS STS 全局端點的會話令牌的區域兼容性	您現在可以選擇是否使用版本 1 或版本 2 的全域端點權杖。版本 1 令牌僅在默認情況下可用的 AWS 區域中有效。這些權杖不適用於手動啟用的區域，例如亞太區域 (香港)。版本 2 權杖在所有區域都有效。不過，版本 2 權杖較長且可能會影響暫時存放權杖的系統。	2019 年 4 月 26 日
允許啟用和停用 AWS 區域	您現在可以建立可讓管理員啟用和停用亞太區域 (香港) (ap-east-1)的政策。	2019 年 4 月 24 日
IAM 使用者的「我的安全憑證」頁面	IAM 使用者現在可以在 My Security Credentials (我的安全憑證) 頁面上管理其所有憑證。此 AWS Management Console 頁面顯示帳戶資訊，例如帳號 ID 和規範使用者 ID。使用者也可以檢視和編輯其密碼、存取金鑰、X.509 憑證、SSH 金鑰和 Git 憑證。	2019 年 1 月 24 日
存取 Advisor API	您現在可以使用 AWS CLI 和 AWS API 來檢視上次存取的服務資訊。	2018 年 12 月 7 日
標記 IAM 使用者和角色	您現在可以使用 IAM 標籤，以使用標籤鍵值組將自訂屬性新增到身分 (IAM 使用者或角色)。您也可以使用標籤來控制身分對資源的存取權或控制哪些標籤可以連接到身分。	2018 年 11 月 14 日

U2F 安全金鑰	您現在可以在登入 AWS Management Console時使用 U2F 安全金鑰的多重要素驗證 (MFA) 選項。	2018 年 9 月 25 日
支援 Amazon VPC 端點	您現在可以在 VPC 和 AWS STS 美國西部 (奧勒岡) 區域之間建立私人連線。	2018 年 7 月 31 日
許可界限	新功能可讓您更輕鬆地授與信任的員工管理 IAM 許可的能力，無需授與完整的 IAM 管理存取許可。	2018 年 7 月 12 日
AWS：PrincipalOrg身份證	透過指定 IAM 主體的 AWS 組織，新的條件金鑰可讓您更輕鬆地控制 AWS 資源的存取。	2018 年 5 月 17 日
AWS：RequestedRegion	新的條件金鑰可讓您更輕鬆地使用 IAM 政策來控制對 AWS 區域的存取。	2018 年 4 月 25 日
增加 IAM 角色的工作階段持續時間	IAM 角色現在可以有 12 小時的工作階段持續時間。	2018 年 3 月 28 日
更新角色建立工作流程	新工作流程改善建立信任關係及連接許可至角色的處理程序。	2017 年 9 月 8 日
AWS 帳戶 登入程序	更新的 AWS 登入體驗允許根使用者和 IAM 使用者使用首頁上的 「登入 AWS Management Console主控台」連結。	2017 年 8 月 25 日
範例 IAM 政策	文件更新推出超過 30 個範例政策。	2017 年 8 月 2 日

IAM 最佳實務	資訊新增到 IAM 主控台的Users (使用者) 區段，可更輕鬆遵循 IAM 最佳實務。	2017 年 7 月 5 日
Auto Scaling 資源	資源層級許可能夠控制 Auto Scaling 資源的存取和許可。	2017 年 5 月 16 日
Amazon RDS for MySQL 和 Amazon Aurora 資料庫	資料庫管理員可以將資料庫使用者與 IAM 使用者和角色建立關聯，從而管理使用者從單一位置存取所有 AWS 資源。	2017 年 4 月 24 日
服務連結角色	服務連結角色可讓您更輕鬆、更安全地將權限委派給 AWS 服務。	2017 年 4 月 19 日
政策摘要	新的政策摘要讓您更輕鬆了解 IAM 政策中的許可。	2017 年 3 月 23 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。